Contents

List of Tables 3

Abstract 7

1 Introduction 9
  1.1 Motivation .................................. 9
  1.2 Sports Scheduling Terminology .................. 10
  1.3 Competition systems .............................. 11
  1.4 Scheduling competition in NBA ..................... 13
  1.5 Present schedule Euroleague Basketball ........... 15
  1.6 Objectives .................................... 16
  1.7 Antecedents .................................... 16

2 Mathematical concepts used in the project 19
  2.1 Graph Theory .................................. 19
     2.1.1 Graphs and Digraphs ......................... 19
     2.1.2 Paths and Cycles ............................. 22
     2.1.3 Hamiltonian Paths and Cycles ................. 23
     2.1.4 Edge coloring ................................ 24
     2.1.5 Tournaments .................................. 24
  2.2 Spherical Distance ............................... 25

3 Proposal of competition schedule 27
  3.1 Teams taking part on the competition ............... 27
     3.1.1 Requeriments ................................ 27
     3.1.2 Participant teams .............................. 27
  3.2 Requeriments of the competition ................... 28
  3.3 Proposed competition schedule ..................... 28
3.4 Allocation of the teams ................................................. 29
3.5 Scheduling the competition ........................................... 30
  3.5.1 Scheduling divisions ............................................. 30
  3.5.2 Scheduling non-conference ...................................... 34
  3.5.3 Scheduling conference (non-division) ......................... 36
  3.5.4 Block games ..................................................... 37
3.6 Sequence of games blocks ........................................... 38

4 Implementation ......................................................... 41
  4.1 Used software .................................................... 41
  4.2 Description of the programs ...................................... 41
  4.2.1 Program kmeans ............................................... 41
  4.2.2 Program modifica .............................................. 42
  4.2.3 Program camihamiltonia ....................................... 42
  4.2.4 Program kmeassubgrups ........................................ 42
  4.2.5 Program modificasubgrups ...................................... 43
  4.2.6 Program calendari .............................................. 44
  4.2.7 Program desplaçament .......................................... 44
  4.2.8 Program desplaçamentcurt .................................... 45
  4.2.9 Program calendarifinal ......................................... 45

5 Conclusions .......................................................... 47
  5.1 Final schedule .................................................... 47
  5.2 Possible improvements ............................................. 56

Bibliography ............................................................ 57

A Annex ................................................................. 61
  A.1 Program kmeans .................................................. 61
  A.2 Program modifica .................................................. 65
  A.3 Program camihamiltonia .......................................... 69
  A.4 Program kmeassubgrups .......................................... 72
  A.5 Program modificasubgrups ........................................ 76
  A.6 Program calendari .................................................. 78
  A.7 Program desplaçament ............................................. 84
  A.8 Program desplaçamentcurt ....................................... 89
  A.9 Program calendarifinal ............................................ 96
List of Tables

1.1 Distribution of the NBA teams ......................... 14
3.1 Cities ................................................. 28
3.2 Allocation of the teams ............................... 30
3.3 Initial Division Schedule ......................... 32
3.4 Home-and-Away Pattern ............................ 32
3.5 Final Division Schedule ......................... 33
3.6 Blocs de partits .................................... 37
# List of Figures

1.1 Localization of the NBA teams ........................................... 14  
1.2 NBA Schedule ................................................................. 15  

2.1 Graph of order 6 and size 7 ............................................... 20  
2.2 Regular bipartite graph $K_{3,3}$ ....................................... 20  
2.3 Complete digraph $K_6$ ..................................................... 21  
2.4 Weighted digraph ........................................................... 23  
2.5 Hamiltonian Cycle ........................................................... 23  
2.6 Example of edge coloring .................................................. 24  
2.7 Lines of latitude and longitude in Earth ............................. 25  
2.8 Spherical triangle ............................................................ 26  
2.9 Example of spheric distance calculation ............................. 26  

3.1 Map of the cities ............................................................. 29  
3.2 Distribution of the teams .................................................. 30  
3.3 Scheduling Division A ...................................................... 33  
3.4 Scheduling Division B ...................................................... 33  
3.5 Scheduling Division C ...................................................... 34  
3.6 Scheduling Division D ...................................................... 34  
3.7 Bipartite tournament $K_{6,6}$ ............................................. 35  
3.8 Hamiltonian cycle of minimum distance A ......................... 36  
3.9 Hamiltonian cycle of minimum distance B ......................... 36  
3.10 Hamiltonian cycle of minimum distance C ......................... 37  
3.11 Hamiltonian cycle of minimum distance D ......................... 37  
3.12 Road trip Division A to Division D ................................... 38  
3.13 Bipartite tournament $K_{3,3}$ ........................................... 39  
3.14 Subdivision A1 move to B1 .............................................. 39  
3.15 Subdivision A2 move to B2 .............................................. 39
LIST OF FIGURES

4.1 Programs kmeans, modifica and camihamiltonia . . . . . . . . . 43
4.2 Programs kmeanssubgrups and modificasubgrups . . . . . . . . 44
4.3 Program calendari . . . . . . . . . . . . . . . . . . . . . . . 45
4.4 Program desplaçament . . . . . . . . . . . . . . . . . . . . . . 45
4.5 Program desplaçamentcurt . . . . . . . . . . . . . . . . . . . 46
4.6 Program calendarifinal . . . . . . . . . . . . . . . . . . . . . 46
Acknowledgements

I would like to sincerely thank the teachers Dr. Josep M Miret, Dr. Josep Conde and Dr. Nacho Lopez for their valuable help when I was making this project. They have always been willing to cooperate and their contributions have been vital to achieve the objective.
Abstract

The object of this project is to schedule a fictitious European basketball competition with many teams situated a long distances. The schedule must be fair, feasible and economical, which means that the total distance traveled by every team must be the minimal possible. First, we define the sport competition terminology and study different competition systems, focusing on the NBA and the Euroleague systems. Then we define concepts of graph theory and spherical distance that will be needed. Next we propose a competition system, explaining where will be allocated the teams and how will be the scheduling. Then there is a description of the programs that have been implemented, and, finally, the complete schedule is displayed, and some possible improvements are mentioned.
Chapter 1

Introduction

1.1 Motivation

Professional sports are nowadays a big business with many involved agents: TV networks offer large amount of money in order to broadcast the most attractive games of every competition, players and trainers are very well payed to do their job and thousands of fans want to see their favourite team playing, both home and away. Furthermore, there are many other people involved: mass media, security forces, airlines and other displacing means, organizers and many others. Therefore, it is a very important economic activity which needs an appropriate organization to optimize its performance.

One of the most important factors of this organization is the schedule of the games of the competition, that is, how to determine the date and the venue in which every game will be played. A good schedule is vital for a competition’s success in order to maximize revenues, ensure the attractiveness of the games and keep the interest of both the media and the fans. But finding the best schedule of the games is a difficult task with multiple constraints and objectives. It must be fair (granting the same conditions to all teams), feasible (fulfilling every logistic and organizational requirement) and economical (minimizing the total distance traveled by the teams).

This is specially important in competitions with many teams which have to travel long distances. An example is the National Basketball Association (N.B.A.), the American professional basketball competition, with thirty
teams that stretch from the American east coast to the west coast and from
the Canadian border to Florida. In order to reduce the amount of travel,
teams sometimes do not return home after every away game. Instead, they
visit some opposing teams before going back home. On the other hand,
teams do not like to be too much time away or too much time playing home.
So, there is a conflict between travel distances and the home/away pattern
which difficults the task of making good schedules.

There are important mathematical concepts underlying the scheduling
problem, most of all related with graph theory. Of course, we have the help
of computers which allow us to make programs to do the long required cal-
culations.

The main objective of the present project is to schedule a fictional eu-
ropen basketball competition similar to the N.B.A., using concepts of graph
theory and making the computer programs that will be needed.

1.2 Sports Scheduling Terminology

We will use the terminology proposed by [3] and [15]. In a sport competi-
tion, \( n \) teams play games between them according to a given timetable. Each
game consists of an ordered pair of teams \((i, j)\). The first team, \( i \) plays at
home (it uses its own venue) and the second team, \( j \), plays away. Games
are scheduled in rounds. Each round is played in a given day. A schedule
consists of games assigned to rounds. A schedule is compact if each team
plays exactly one game in each round, otherwise it is relaxed.

If a team plays two home or two away games in two consecutive rounds,
it is said to have a break. A series of consecutives away games is called an
away tour or a road trip. Consecutive home games are a home stand. The
length of a road trip or home stand is the number or opponents played.
1.3 Competition systems

- **Round robin competition** Every team plays against every other team a fixed number of times. Most sport leagues play a double round robin tournament where the teams meet twice (once at home, once away). The number of rounds in a compact single round robin tournament is $n - 1$ and the number of games is $\frac{n(n - 1)}{2}$. A *mirrored* double round robin tournament is a tournament where every team plays against every other team once in the first $n - 1$ rounds, followed by the same games with reversed venues in the last $n - 1$ rounds.

There is a table where the teams add points depending on its result in every game. Usually they add three points if they have won, one in case of draw and none if they have lost. Obviously, the champion is the team that has more points at the end of the competition, and sometimes the poorest performers are subject to relegation.

In theory this is the fairest way to determine a champion, because each competitor has an equal chance against all other participants and the element of luck is seen to be reduced.

The primary disadvantage is the long time needed to complete it. Another problem can be in later matches of the competition that sometimes pair one competitor with options to be champion against another without, who maybe would play softer.

- **Elimination or knockout tournament** Every participant plays with other participants and the loser is eliminated. The number of participants reduces every round until there is only two survivors, which play the final match, whose winner will be the champion. The round can consist of a single game, two games (usually, each participant plays once at home and once away and the two scores are added), or several games (also known as playoff where the winner is the first competitor to win a fixed number of matches).

This format has some advantages: it enables a relatively large number
of competitors to participate. There are no matches where one competitor has more to play than the other.

On the other hand, it has also disadvantages: most participants are eliminated after few games. It is less suited to games where draws are frequent. Many times, luck has an important influence and the champion is not the best participant.

- **Swiss-system tournament** There are several rounds of competition and the winner is the player or team with the highest aggregate of points earned from each round. In contrast to the round robin competition not every player is matched with all other players during the tournament, and opposed to knockout tournament, the loser of any round is not eliminated. It consists of a predeterminate number of rounds. In the first round, players are paired either according to some pattern or randomly. In subsequent rounds, players are sorted according to their cumulative scores and they are assigned opponents with the same or similar score up to that point. Two players never have to oppose each other twice in the same tournament.

It is used when it is not feasible to play so many matches as in a round robin tournament. It would require to determine a clear winner the same number of rounds as a knockout tournament, but it has the advantage of not eliminating anyone. The final ranking gives some information of relative strenght for all contestants. The main disadvantage is that as competition goes on, there are a number of inconsequential games.

It is commonly used in chess, bridge and other cards competitions.

- **Mixed systems** Some of the previous systems are used in the same competition. Usually there is a first phase with a round robin competition with one or several groups. Then, the best classified teams play the second phase, that can be smaller groups with also round robin tournament, or knockout system.
1.4 Scheduling competition in NBA

The information of this section has been taken from [14]. It is a mixed tournament, with a first phase (called regular season) played with a special round robin system and a second phase (play-off) played with the knockout system.

To play the regular season there are 30 teams distributed into two conferences (West and East) each one divided into three divisions with five teams each, as it is shown in table 1.1. In figure 1.1 we can see the localization of these teams. An special feature of the regular season is that the teams don’t play the same number of times against each other. Due to geographical reasons, teams play more games against their conference opponents that against their opposite conference adversaries.

Each team have to play:

- 4 games against the other 4 teams of the same division (2 home and 2 away), \[4 \times 4 = 16 \text{ games}\]
- 4 games against 6 out-of-division conference opponents (2 home and 2 away), \[4 \times 6 = 24 \text{ games}\]
- 3 games against the remaining 4 conference teams (2 home and 1 away or 1 home and 2 away), \[3 \times 4 = 12 \text{ games}\]
- 2 games against the 15 teams in the opposite conference (1 home and 1 away), \[2 \times 15 = 30 \text{ games}\]

Another important aspect about scheduling the regular season is the building availability. The arenas where the teams play are not always free to use. So any team before the end of a season have to submit to the NBA office a list of at least 50 dates on which their home court will be available next season. With these dates and with the help of a computer programm, Mr Matt Winick, the senior vice president of scheduling and game operations for the NBA, prepares every year the schedule. The most important part to prepare for every team are the games against the other conference. As they normally involve long road trips, he schedules them during the time that the team can’t play home. Once this is done, he schedules the games against their conference opponents.
The result is a highly relaxed schedule without the concept of rounds. Games are played every day, except Christmas eve and during the all-star game. There is no regularity involving the number the matches to play every day or the amount of consecutive home or away games played for every team. We can observe this in the figure 1.2.

Once the regular season is finished, the 16 best classified (8 of each conference) will play the play-offs.

Table 1.1: Distribution of the NBA teams

<table>
<thead>
<tr>
<th>Northwest</th>
<th>Pacific</th>
<th>Southwest</th>
<th>Atlantic</th>
<th>Central</th>
<th>Southeast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denver</td>
<td>Oakland</td>
<td>Dallas</td>
<td>Boston</td>
<td>Cleveland</td>
<td>Atlanta</td>
</tr>
<tr>
<td>Minneapolis</td>
<td>Los Angeles</td>
<td>Houston</td>
<td>New York</td>
<td>Chicago</td>
<td>Charlotte</td>
</tr>
<tr>
<td>Oklahoma C.</td>
<td>Los Angeles</td>
<td>Memphis</td>
<td>New York</td>
<td>Detroit</td>
<td>Miami</td>
</tr>
<tr>
<td>Portland</td>
<td>Phoenix</td>
<td>New Orleans</td>
<td>Philadelphia</td>
<td>Indiana</td>
<td>Orlando</td>
</tr>
<tr>
<td>Salt Lake C.</td>
<td>Sacramento</td>
<td>San Antonio</td>
<td>Toronto</td>
<td>Milwaukee</td>
<td>Washington</td>
</tr>
</tbody>
</table>

Figure 1.1: Localization of the NBA teams
1.5 Present schedule Euroleague Basketball

We have consulted the internet page [4] to redact these section. There are 24 participant teams, chosen according to its ranking and previous results in national leagues. It is a mixed system with several phases, in which the teams are not distributed by geographical reasons, but for their ranking or results.

- Regular season.
  The 24 teams are divided into 4 groups of six teams. This part is played under a mirrored round-robin format, for a total of 10 games played by each team. The best 4 teams for each group advance to the next phase.

- Top 16.
  The previously classified 16 teams are divided into 2 groups of eight teams. It is also played as a mirrored round-robin format, for a total of 14 games played by each team. The best 4 teams for each group advance.
• Play-off.
The eight classified teams play best-of-five series. The first classified of
each group plays against the 4th classified of the other group, and the
second against the 3rd. Winners advance to final four.

• Final four.
It consists of single elimination matchups. There are 2 semifinals and
the winners play the final.

1.6 Objectives
The main objective of the present project is to imagine a fictitious european
basketball competition and plan how could be the schedule with the mini-
mum travel cost, and also being fair with all the teams and attractive to the
fans and media. In order to achieve this, other intermediate objectives are:

• Determine the main restrictions and problems that have to be taken
into consideration when scheduling a sports competition.

• Examine which concepts of graph theory or other mathematical fields
can help to make a good schedule.

• Create an informatic application that can be used to schedule this
competition while conforming to all objectives and constraints.

1.7 Antecedents
Sports scheduling is now an important research area, with numerous papers
recently published. There are some very useful collection of these papers:
[11] is a webpage where all the related published papers are classified accord-
ing to three aspects: model, method and sports discipline. [8] also presents a
similar list giving a brief description of every one of these papers. [15] designs
a framework for a sports scheduling problem. It defines a problem, details
its constraints and introduces the terminology. Then it proposes solutions to
some professional league sports.
Some papers refer to graph theoretical models for scheduling sports, for example, [18] uses graph theory to deal with geographical constraints for teams located in different cities. [19] and [20] use models of graphs to minimize breaks when scheduling. There are some works which use graph theory to schedule specific sports competitions, like [17] where the Dutch Professional Football League is scheduled, [5] with the basketball UMAC (Upper Midwest Athletic Conference) conference, [22] with the Major League Baseball (MLB) or [13] with the Spanish Football League. [9] uses graph theory to make a new schedule of N.B.A. trying to minimize the travel costs, and with similar methods to this project.

An important issue related to sports scheduling is the traveling tournament problem, which consists of minimize the total distance traveled by the teams, but maintaining the length of every home stand and road trip into some established limits. [3] defines it and proposes some methods to solve it, while [7] uses some algorithms to solve it and applies it in the Japanese baseball competition.
Chapter 2

Mathematical concepts used in the project

2.1 Graph Theory

Most of the definitions of this chapter are taken from [2] and [6].

2.1.1 Graphs and Digraphs

Definition 2.1 A graph $G$ is a finite nonempty set of objects called vertices together with a (possibly empty) set of unordered pairs of distincts vertices of $G$ called edges. The vertex set of $G$ is denoted by $V(G)$, while the edge set is denoted by $E(G)$. Graphs are represented graphically by drawing a dot for every vertex and drawing an arc between two vertices if they are connected by an edge.

The edge $e = (u, v)$ is said to join the vertices $u$ and $v$, which are adjacent vertices, while $u$ and $e$ are incident, as are $v$ and $e$.

Definition 2.2 Order of $G$ is the cardinality of $V(G)$, that is, the number of vertices of $G$, while size of $G$ is the cardinality of $E(G)$, that is, the number of edges of $G$. In figure 2.1 we can see a graph of order 6 and size 7.

Definition 2.3 The degree of a vertex $v$ in a graph $G$ is the number of edges of $G$ incident with $v$. A vertex is called odd or even depending on whether its degree is odd or even.
Definition 2.4 A complete graph $K_n$ of order $n$ is a graph with $n$ vertices in which every vertex is adjacent to the others. It has $\frac{n(n-1)}{2}$ edges.

Definition 2.5 A regular graph is a graph in which every vertex has the same degree. It is $k$-regular if every vertex has degree $k$.

Definition 2.6 A bipartite graph is a graph that can be decomposed into two partite sets. For example, in the figure 2.2 there is a regular bipartite graph.

Definition 2.7 A directed graph or digraph $D$ is a finite nonempty set of objects called vertices together with a (possibly empty) set of ordered pairs of distinct vertices of $D$ called arcs or directed edges. The vertex set of $D$ is denoted by $V(D)$, while the arc set is denoted by $E(D)$. Graphically, the direction is indicated by drawing an arrow on the edge.

The concepts of order and size of $D$ are analogue to that used for graphs. So, order is the number of vertices and size is the number of arcs of $D$.

If $a = (u, v)$ is an arc of $D$, then $a$ is said to join $u$ and $v$. It is also said that $a$ is incident from $u$ and incident to $v$, while $u$ is incident to $a$ and $v$ is incident from $a$. Furthermore, $a$ is said to be adjacent to $v$ and $v$ is adjacent from $u$.

Definition 2.8 The outdegree, $od(v)$, of a vertex $v$ in a digraph $D$ is the number of arcs of $D$ incident from $v$. The indegree, $in(v)$, of a vertex $v$ in a digraph $D$ is the number of arcs of $D$ incident to $v$. The degree, $deg(v)$, of a vertex $v$ is the sum of the outdegree and the indegree.
Theorem 2.9 If $D$ is a digraph of order $p$ and size $q$ where $V(G) = \{v_1, \ldots, v_p\}$, then,

$$\sum_{i=1}^{p} \text{od } v_i = \sum_{i=1}^{p} \text{id } v_i = q \quad (2.1)$$

Definition 2.10 A symmetric digraph is a digraph $D$ in which whenever $(u, v)$ is an arc of $D$, then $(v, u)$ is as well.

Definition 2.11 An asymmetric digraph or oriented graph is a digraph $D$ in which whenever $(u,v)$ is an arc of $D$, then $(v, u)$ is not an arc of $D$.

Definition 2.12 A complete digraph is a digraph $D$ in which for every two distinct vertices $u$ and $v$ of $D$, at least one of the arcs $(u, v)$ or $(v, u)$ is present in $D$. The complete digraph $K_6$, with 6 edges is shown in the figure 2.3

Figure 2.3: Complete digraph $K_6$

Definition 2.13 The complete symmetric digraph of order $p$ has both arcs $(u, v)$ and $(v, u)$ for every two distinct vertices $u$ and $v$ and is denoted by $K_p^*$.

Definition 2.14 A regular digraph of degree $r$ or $r$-regular digraph is a digraph in which $\text{od } v = \text{id } v = r$ for every vertex $v$ of it.
2.1.2 Paths and Cycles

Definition 2.15 A walk is an alternative sequence of vertices and edges, beginning and ending with a vertex, where each vertex is incident to both the edge that precedes it and the edge that follows it in the sequence. A walk is closed if its first and last vertices are the same and open if they are different.

Definition 2.16 The length $l$ of a walk is the number of edges that it uses. For an open walk $l = n - 1$, where $n$ is the number of vertices visited. For a close walk, $l = n$ (the start/end vertex is listed twice, but is not counted twice).

Definition 2.17 A trail is a walk in which all the edges are distinct. A close trail has been called a tour or circuit.

Definition 2.18 A path is a walk in which no vertices (and thus no edges) are repeated.

Definition 2.19 A cycle is a closed walk with no repeated vertices or edges.

Definition 2.20 A connected graph is a graph in which it is possible to establish a path from any vertex to any other. Otherwise, the graph is disconnected.

Definition 2.21 Distance $d(u, v)$ between two vertices $u$ and $v$ of a connected graph $G$ is the minimum of the lengths of the $u - v$ paths of $G$.

Definition 2.22 A weighted graph is a graph in which each edge $e$ is assigned a positive number, called the weight of $e$ and denoted by $w(e)$. Figure 2.4 shows an example of a weighted graph.

Definition 2.23 Length of a path $P$ in a weighted graph $G$ is the sum of the weights of the edges of $P$. 
2.1.3 Hamiltonian Paths and Cycles

**Definition 2.24** A path or cycle is hamiltonian if it uses all vertices exactly once. A graph that contains a hamiltonian path is traceable and one that contains a hamiltonian cycle is a hamiltonian graph. For example in 2.5 we find a hamiltonian cycle: ABCDENMLKJIHGQRSTPOFA.
Definition 2.25  Traveling Salesman Problem is the determination of a minimum hamiltonian cycle in a weighted complete graph $G$ of order $p$. This can be determined by inspecting $(p - 1)!$ hamiltonian cycles and looking for the one with minimum weight. Any algorithm that does it is clearly not efficient, unless $p$ is a small number. There are approximate methods that are more efficient, but they do not guarantee the optimal solution.

2.1.4  Edge coloring

Definition 2.26  An edge coloring of a graph is an assignment of colors to its edges so that no two adjacent edges have the same color, as we see in the figure 2.6.

![Figure 2.6: Example of edge coloring](image)

Definition 2.27  The edge-chromatic number of a graph is the least number of colors it takes to color its edges so that adjacent edges have different colors.

2.1.5  Tournaments

A tournament is a digraph in which each pair of vertices is connected by exactly one arc. In other words, it is an oriented complete graph or a complete asymmetric digraph.
2.2 Spherical Distance

In this section we will mostly follow the explanations of [1]. The Earth is approximately an sphere and, thus, we can apply the concepts of spheric trigonometry in order to calculate the distance between two points in its surface.

Every point in the Earth is located by its geographical coordinates: longitude and latitude.

Definition 2.28 Longitude of a point on the Earth’s surface is the angle east (+) or west (-) from a reference meridian (usually the Greenwich meridian) to another meridian that passes through that point.

Definition 2.29 Latitude of a point on the Earth’s surface is the angle between the equatorial plane and a line that passes through that point and is normal to the surface of a reference ellipsoid which approximates the shape of the Earth.

The units of both longitude and latitude are degrees, minutes and seconds. In the figure 2.7, the longitude of the point M is the angle OAB, and the latitude is the angle OBM.

Figure 2.7: Lines of latitude and longitude in Earth

Definition 2.30 A spherical triangle is a figure formed on the surface of a sphere by three great circular arcs intersecting pairwise in three vertices. We can see an example in the figure 2.8.

Bessel’s formula: \[ \cos a = \cos b \cos c + \sin b \sin c \cos A \]

Definition 2.31 Spherical distance between two points on the surface of one sphere is the length of the shorter arc between them.
We will see with an example how we can calculate the spherical distance. Supposing the geographical coordinates of cities A and B from figure 2.9 are:

A(longitude; latitude) = A(55° 45’ 13" E; 55° 48’ 10" N)
B(longitude; latitude) = A(48° 50’ 2" E; 20° 30’ 40" N)

We can see the spherical triangle PBA, where we have:

\[ a = PB = 90° - 48° 50’ 2" = 41° 9’ 58" \]
\[ b = PA = 90° - 55° 45’ 13" = 34° 14’ 47" \]
\[ P = 55° 48’10" - 20° 30’ 40" = 35° 17’ 30" \]

Applying Bessel’s formula: \( \cos p = \cos a \cos b + \sin a \sin b \cos P = 0.925 \)

Thus, we have: \( p = 22.386° \)

Since 360 correspond to 40000 km (the length of a maximum circular arc), the spherical distance AB is equal to 2487.33 km
Chapter 3

Proposal of competition schedule

3.1 Teams taking part on the competition

3.1.1 Requirements

Every team which want to participate in the competition must accomplish some requirements:

- The club must have an available venue in its city to play its home games.
- The team must confirm the existence of an international airport at a maximum distance of 100 km by road from the arena, with enough daily flights to allow the visiting team to have access to the city under the right conditions.
- The team must resign to play its national competitions, as it will not be enough days to participate in both.

3.1.2 Participant teams

The participant teams have been selected according to their ranking over the last five years in the previous Euroleague competition, with the addition of Paris and London, which are expected to became important centers of basket. In the table 3.1 there are the cities with a team participating in the
competition with its geographical coordinates (there are two teams in some
cities), and in the figure 3.1 there is a map of Europe where they are located.

Table 3.1: Cities

<table>
<thead>
<tr>
<th>City</th>
<th>Latitude</th>
<th>Longitude</th>
<th>City</th>
<th>Latitude</th>
<th>Longitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barcelona</td>
<td>41°23'0&quot; N</td>
<td>2°11'0&quot; E</td>
<td>Bologna</td>
<td>44°30'27&quot; N</td>
<td>11°21'5&quot; E</td>
</tr>
<tr>
<td>Madrid</td>
<td>40°23'0&quot; N</td>
<td>3°43'0&quot; W</td>
<td>Milan</td>
<td>45°27'51&quot; N</td>
<td>9°11'25&quot; E</td>
</tr>
<tr>
<td>Vitoria</td>
<td>42°51'0&quot; N</td>
<td>2°41'0&quot; W</td>
<td>Siena</td>
<td>43°19'7&quot; N</td>
<td>11°19'50&quot; E</td>
</tr>
<tr>
<td>Lyon</td>
<td>45°45'35&quot; N</td>
<td>4°50'32&quot; E</td>
<td>Ljubljana</td>
<td>46°3'20&quot; N</td>
<td>14°30'30&quot; E</td>
</tr>
<tr>
<td>Paris</td>
<td>48°51'24&quot; N</td>
<td>2°21'3&quot; E</td>
<td>Zagreb</td>
<td>45°49'0&quot; N</td>
<td>15°59'0&quot; E</td>
</tr>
<tr>
<td>London</td>
<td>51°30'20&quot; N</td>
<td>0°7'39&quot; W</td>
<td>Bamberg</td>
<td>49°53'30&quot; N</td>
<td>10°53'30&quot; E</td>
</tr>
<tr>
<td>Belgrade</td>
<td>44°49'0&quot; N</td>
<td>20°28'0&quot; E</td>
<td>Warsaw</td>
<td>52°13'56&quot; N</td>
<td>21°0'30&quot; E</td>
</tr>
<tr>
<td>Athens (2)</td>
<td>37°58'0&quot; N</td>
<td>23°43'0&quot; E</td>
<td>Berlin</td>
<td>52°31'0&quot; N</td>
<td>13°23'0&quot; E</td>
</tr>
<tr>
<td>Tel Aviv</td>
<td>32°4'0&quot; N</td>
<td>34°47'0&quot; E</td>
<td>Kaunas</td>
<td>54°54'0&quot; N</td>
<td>23°56'0&quot; E</td>
</tr>
<tr>
<td>Istanbul (2)</td>
<td>41°0'49&quot; N</td>
<td>28°57'18&quot; E</td>
<td>Vilnius</td>
<td>54°41'0&quot; N</td>
<td>25°17'0&quot; E</td>
</tr>
<tr>
<td>Moscow (2)</td>
<td>55°45'0&quot; N</td>
<td>37°37'0&quot; E</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.2 Requeriments of the competition

- Each team will play every other team at least once home and once away, but not necessarily the same number of times.
- Esch team will play the same number of games home than away.
- The total distance traveled by the teams should be minimized.
- Every round consist of the same number of games.
- The lenght of a road trip or home stand should not exceed 6 matches.

3.3 Proposed competition schedule

The teams are arranged into two conferences: West and East. Each conference consits of two divisions: A and B (West) and C and D (Est), every one
3.4 Allocation of the teams

of which is also divided into two subdivisions. There will be six teams in every division and three teams in every subdivision.

Each team will play every team in their conference four times: two of which must be away games and the other two must be home games.

Each team will play every team that is not in their conference two times: one away game and one home game.

To summarize, every team will play:

- 44 Conference Games (4 x 11). 22 home games and 22 away games
- 24 Non-Conference Games (2 x 12). 12 home games and 12 away games

Therefore, there will be 68 rounds, each one with 12 games, which makes a total of 816 games to be scheduled.

In order to achieve homogeneous rounds, they will be organized in blocks.

3.4 Allocation of the teams

Once applied the informatic programmes that will be explained in 4, the participant teams are allocated the way we see in table 3.2 and in figure 3.2:
Table 3.2: Allocation of the teams

<table>
<thead>
<tr>
<th>Conference</th>
<th>West</th>
<th>East</th>
</tr>
</thead>
<tbody>
<tr>
<td>Division</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Subdivision</td>
<td>A1</td>
<td>B1</td>
</tr>
<tr>
<td></td>
<td>Barcelona</td>
<td>Bologna</td>
</tr>
<tr>
<td></td>
<td>Madrid</td>
<td>Milan</td>
</tr>
<tr>
<td></td>
<td>Vitoria</td>
<td>Siena</td>
</tr>
<tr>
<td>Subdivision</td>
<td>A2</td>
<td>B2</td>
</tr>
<tr>
<td></td>
<td>Lyon</td>
<td>Ljubljana</td>
</tr>
<tr>
<td></td>
<td>Paris</td>
<td>Zagreb</td>
</tr>
<tr>
<td></td>
<td>London</td>
<td>Bamberg</td>
</tr>
</tbody>
</table>

Figure 3.2: Distribution of the teams

3.5 Scheduling the competition

There will be three different types of competition:

3.5.1 Scheduling divisions

In terms of graph theory, each division is a complete graph with 6 vertices: $K_6$, and the schedule is a tournament with an edge-coloring.
3.5 Scheduling the competition

The teams are represented by points and for each pair of points an arc is drawn from the visiting team to the home team. If a game between the teams $i$ and $j$ is played in the home-city of team $i$, it is a home-game for $i$ and an away game for $j$, which can be represented by an arc $(j, i)$.

The in-degree of a node would refer to the number of home games the represented team would play, as well as the out-degree would refer to the number of the away games.

An oriented coloring in tournaments is obtained by partitioning the arcs into $n$ color classes such that no two adjacents edges have the same color. Each color corresponds to a specific round. Then, the schedule is defined this way: if arc $(i, j)$ has color $p$, it means that teams $i$ and $j$ play against each other in the home city of team $j$ on the round $p$.

When a team plays an away game, it is assumed to travel from its home site to the away venue and then to return home the same day, as the distances would not be too long. They are the blocks (A) and (B) in 3.6.

To schedule 6 teams in a round robin tournament we will follow the explanations of [12]. We will construct an initial table with the first half of the teams listed consecutively in the first row and the last half of the teams listed in reverse order in the next row. The teams that line up in the table will play in the first round.
Round 1:

```
1 2 3
6 5 4
```

To get to the pairings for the next round we will fix the first team in the cell in the first row and first column but we will move all the other teams in the clockwise direction.
Round 2:

```
1 6 2
5 4 3
```
We repeat this rotation operation until we exhaust the new pairings and return to the start.

Round 3: \[
\begin{array}{ccc}
1 & 5 & 6 \\
4 & 3 & 2 \\
\end{array}
\]
Round 4: \[
\begin{array}{ccc}
1 & 4 & 5 \\
3 & 2 & 6 \\
\end{array}
\]
Round 5: \[
\begin{array}{ccc}
1 & 3 & 4 \\
2 & 6 & 5 \\
\end{array}
\]

With this, the resultant schedule is shown in the table 3.3

Table 3.3: Initial Division Schedule

<table>
<thead>
<tr>
<th>R.1:</th>
<th>(6, 1)</th>
<th>(5, 2)</th>
<th>(4, 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R.2:</td>
<td>(5, 1)</td>
<td>(4, 6)</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>R.3:</td>
<td>(4, 1)</td>
<td>(3, 5)</td>
<td>(2, 6)</td>
</tr>
<tr>
<td>R.4:</td>
<td>(3, 1)</td>
<td>(2, 4)</td>
<td>(6, 5)</td>
</tr>
<tr>
<td>R.5:</td>
<td>(2, 1)</td>
<td>(6, 5)</td>
<td>(5, 4)</td>
</tr>
</tbody>
</table>

But this schedule does not take into account the distribution of home-games and away-games, since it is desired that each team alternates home and away games. To prevent this, [18] propose to associate a Home-and-Away Pattern (HAP) to the schedule. It is an array where entry \((i, d)\) is \(H\) if the team \(i\) has a home-game on day \(d\) or \(A\) otherwise. The row are the different teams and the columns are the rounds. To minimize the number of breaks, this is the proposed HAP:

Table 3.4: Home-and-Away Pattern

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>H</td>
<td>A</td>
<td>A</td>
<td>H</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>H</td>
<td>A</td>
<td>H</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>H</td>
<td>A</td>
<td>A</td>
<td>H</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>A</td>
<td>H</td>
<td>H</td>
<td>A</td>
<td>H</td>
</tr>
<tr>
<td>5</td>
<td>H</td>
<td>A</td>
<td>H</td>
<td>H</td>
<td>A</td>
</tr>
<tr>
<td>6</td>
<td>H</td>
<td>A</td>
<td>H</td>
<td>A</td>
<td>H</td>
</tr>
</tbody>
</table>

With all this, we can see the final schedule in the table 3.5
3.5 Scheduling the competition

Table 3.5: Final Division Schedule

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R.1:</td>
<td>(1, 6)</td>
<td>(2, 5)</td>
<td>(4, 3)</td>
</tr>
<tr>
<td>R.2:</td>
<td>(5, 1)</td>
<td>(6, 4)</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>R.3:</td>
<td>(1, 4)</td>
<td>(3, 5)</td>
<td>(2, 6)</td>
</tr>
<tr>
<td>R.4:</td>
<td>(1, 3)</td>
<td>(4, 2)</td>
<td>(6, 5)</td>
</tr>
<tr>
<td>R.5:</td>
<td>(2, 1)</td>
<td>(3, 6)</td>
<td>(5, 4)</td>
</tr>
</tbody>
</table>

In the figures 3.3, 3.4, 3.5 and 3.6 we can see the graph representation of the schedule of every division.
3.5.2 Scheduling non-conference.

The six teams belonging to a division will move simultaneously and they will play against the six teams of another division. It can also be represented by a bipartite tournament, in this case, the $K_{6,6}$, that we can see in the figure 3.7. The teams that move are the away teams and the ones that stay are the home teams. Every team begins the tournament in a different place and after having played its first game, it goes to the site of its second opponent instead of going back home. It returns home after its last away game. There are some points to be aware of:

- As the away teams will play with all the home teams, it is necessary to find the route which would minimize the distance they have to do. It is, the hamiltonian cycle with minimum distance. That will be the cycle that all the away teams will follow. The figures 3.8, 3.9, 3.10 and 3.11 show the hamiltonian cycles of every division, with the distances in km.
• The away teams will not have to do the complete cycle, as they will enter on one place, follow the cycle and finish on the previous place, from where they will return home.

• In order to minimize distance, the further away team will be assigned the shortest go and return route.

We can see an example in 3.12 where it is shown the road trip of the teams from division A to division D. The red arrows indicates the entrance of every away team in to the cycle and the blue arrows signals the exit point of the cycle. The team of division A which is farthest to division D is Madrid. Then, it will be assigned the shortest go and back route, which is go to Berlin and return from Warsaw. Thus, Madrid goes to Berlin, then it follows the cycle playing all the games until Warsaw, where it plays the last game and comes back home.

The resulted schedule is stored in the blocks (G), (H), (I) i (J).

Figure 3.7: Bipartite tournament $K_{6,6}$
3.5.3 Scheduling conference (non-division)

The three teams belonging to a subdivision will move simultaneously and they will play against the three teams of another subdivision. It can also be represented by a bipartite tournament, that is, an orientations of a complete bipartite graph, in this case, the $K_{3,3}$ displayed in 3.13. The results will be stored in the blocks (C), (D), (E) and (F).
3.5 Scheduling the competition

Figure 3.10: Hamiltonian cycle of minimum distance C

Figure 3.11: Hamiltonian cycle of minimum distance D

3.5.4 Block games.

The block games shown in table 3.6 are defined:

<table>
<thead>
<tr>
<th>Block</th>
<th>Games</th>
<th>Displacements</th>
<th>Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A)</td>
<td>5</td>
<td>A ; B ; C ; D</td>
<td>2</td>
</tr>
<tr>
<td>(B)</td>
<td>5</td>
<td>A ; B ; C ; D (Reversed venues)</td>
<td>2</td>
</tr>
<tr>
<td>(C)</td>
<td>6</td>
<td>A → C ; B → D</td>
<td>1</td>
</tr>
<tr>
<td>(D)</td>
<td>6</td>
<td>C → A ; D → B</td>
<td>1</td>
</tr>
<tr>
<td>(E)</td>
<td>6</td>
<td>A → D ; B → C</td>
<td>1</td>
</tr>
<tr>
<td>(F)</td>
<td>6</td>
<td>D → A ; C → B</td>
<td>1</td>
</tr>
<tr>
<td>(G)</td>
<td>3</td>
<td>A1 → B1 ; A2 → B2 ; C1 → D1 ; C2 → D2</td>
<td>2</td>
</tr>
<tr>
<td>(H)</td>
<td>3</td>
<td>B2 → A1 ; B1 → A2 ; D2 → C1 ; D1 → C2</td>
<td>2</td>
</tr>
<tr>
<td>(I)</td>
<td>3</td>
<td>B1 → A1 ; B2 → A2 ; D1 → C1 ; D2 → C2</td>
<td>2</td>
</tr>
<tr>
<td>(J)</td>
<td>3</td>
<td>A1 → B2 ; A2 → B1 ; C1 → D2 ; C2 → D1</td>
<td>2</td>
</tr>
</tbody>
</table>
3.6 Sequence of games blocks

We have the intention to obtain an equitable schedule, that is, to alternate as perfectly as possible the home-games and the away-games for each one of the teams. Another purpose would be to try to space as much as possible the games between the same two opponents. In order to achieve these objectives, we propose the following sequence of blocs:

\[(G); (H); (A); (C); (F), (B); (I); (J); (G); (H); (A); (D); (E), (B); (I); (J)\]

With this sequence, there will be no team which have two consecutive
3.6 Sequence of games blocks

Figure 3.13: Bipartite tournament $K_{3,3}$

Figure 3.14: Subdivision A1 move to B1

Figure 3.15: Subdivision A2 move to B2

road trips or home stands.
Chapter 4

Implementation

In order to decide the most suitable schedule, an informatic application has been created. It consists of several programs, each one with an specific task to do. The information is stored in files that are consulted and updated by the programs.

4.1 Used software

We have used the program *Python*. It has a clear syntax and large standard library. It is freely available from the Python Web site. We have also used the environment *Spyder* in the program *camihamiltonia*, to use graph functions.

4.2 Description of the programs

4.2.1 Program *kmeans*

It looks for the geographic coordinates of the cities, which are stored in the file Ciutats.txt and put the teams in 4 groups, based in geographical nearness. This 4 groups can have a different number of teams. The results are stored in the files GrupA.txt, GrupB.txt, GrupC.txt and GrupD.txt. This program has been based on the K-Means Clustering procedure, which is composed of the following steps:

- Place k points (in our case, ther will be 4) into the space represented by the objects that are being clustered. These points represent initial group centroids..
• 2 Assign each given point (that would represent a city with its geographical coordinates) to the group that has the closest centroid.

• 3 When all the points have been assigned, recalculate the positions of the k centroids as the average of the points assignes to its group.

• 4 Repeat steps 2 and 3 until the centroids no longer move. These will be the final groups produced.

The algorithm is significantly sensitive to the initial selected cluster centres, and it can produce different groups depending on the initial clusters. In order to avoid this problem, we have decided to place the initial centroids not randomly but in the bisector of the rectangle formed by the coordinates of the points. Another problem it has is that the resulted groups can not have the same number of points.

4.2.2 Program modifica

This program consults the resultant files of the previous program and checks which files have less than 6 teams and which ones have more than 6 teams. In the groups with less than 6 teams, it calculates the average of its geographical coordinates. Then, it calculates the distances between the cities of the group with more than 6 teams and the previous averages and observes which teams is nearer. Next, it erases this team of the group with more than 6 teams and adds it to the group which had less teams. Finally, it records the results in the same files, which now have 6 teams each one.

4.2.3 Program camihamiltonia

This program consults the 4 previous files and it calculates in everyone the hamiltonian cycle with minimum length. Then, it records the results in the same files, every one of which have now 6 teams arranged by the hamiltonian cycle with minimum length.

In figure 4.1 there is a diagrama showing how these 3 programs work

4.2.4 Program kmeanssubgrups

This program accesses the data in the files GrupA.txt, GrupB.txt, GrupC.txt and GrupD.txt and distributes the teams in every file in two file, which can
4.2 Description of the programs

Figure 4.1: Programs kmeans, modifica and camihamiltonia

24 teams have different number of teams. It saves the results in the files GrupA1.txt, GrupA2.txt, GrupB1.txt, GrupB2.txt, GrupC1.txt, GrupC2.txt, GrupD1.txt and GrupD2.txt.

4.2.5 Program *modificasubgrups*

It performs in the previous subgroups files the same action als the program modifica. In the end, every subgroup will have 3 teams.

There is a schema of these two programms procedure in the figure 4.2
4.2.6 Program calendar

It accesses to the files GrupA.txt, GrupB.txt, GrupC.txt and GrupD.txt. Then, it organises in everyone a round-robin tournament. The results are recorded in the files BlocA.txt and BlocB.txt. We see it in the figure 4.3.

4.2.7 Program desplaçament

The figure 4.4 shows how this program works. It also uses the data of the files GrupA.txt, GrupB.txt, GrupC.txt and GrupD.txt, but, in this case, it decides how will be played the matches when the 6 teams of one division move to another division of the other conference. It calculates the distances from every origin city to the destination division and assigns the nearest displacement to the farthest city. The results are recorded in the files BlocC.txt, BlocD.txt, BlocE.txt and BlocF.txt,
4.2 Description of the programs

Figure 4.3: Program calendari

Figure 4.4: Program desplaçament

4.2.8 Program desplaçament curt

With the data of the files the files GrupA1.txt, GrupA2.txt, GrupB1.txt, GrupB2.txt, GrupC1.txt, GrupC2.txt, GrupD1.txt and GrupD2.txt it calculates the displacement of the teams belonging to any subgroup to another subgroup of the other division of the same conference. The results are recorded in the files BlocG.txt, BlocH.txt, BlocI.txt and BlocJ.txt, as we can see in the figure 4.5.

4.2.9 Program calendarifinal

It accesses to the data of the files BlocA.txt, BlocB.txt, BlocC.txt,BlocD.txt, BlocE.txt, BlocF.txt, BlocG.txt, BlocH.txt, BlocI.txt and BlocJ.txt, end organizes the schedule of the whole competition with the desired sequence of blocks. Finally, it prints this schedules. A diagram of this is showed in the figure 4.6.
Implementation

Figure 4.5: Program desplacament curt

Grup A1 -> Grup B1 -> Bloc G
Grup A2 -> Grup B2 -> Bloc H
Grup C1 -> Grup D1 -> Bloc I
Grup C2 -> Grup D2 -> Bloc J

Figure 4.6: Program calendarifinal

Bloc A -> Calendarifinal
Bloc B
Bloc C
Bloc D
Bloc E
Bloc F
Bloc G
Bloc H
Bloc I
Bloc J

Calendari
Competici
Chapter 5

Conclusions

5.1 Final schedule

Finally once used all the previously explained programs, we obtain the schedule of the competition with 68 rounds, each one consisting of 12 games.
<table>
<thead>
<tr>
<th>Jornada 1</th>
<th>Jornada 2</th>
<th>Jornada 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bologna - Vitoria</td>
<td>Bologna - Madrid</td>
<td>Bologna - Barcelona</td>
</tr>
<tr>
<td>Milan - Madrid</td>
<td>Milan - Barcelona</td>
<td>Milan - Vitoria</td>
</tr>
<tr>
<td>Siena - Barcelona</td>
<td>Siena - Vitoria</td>
<td>Siena - Madrid</td>
</tr>
<tr>
<td>Warsaw - Tel Aviv</td>
<td>Warsaw - Athens 1</td>
<td>Warsaw - Athens 2</td>
</tr>
<tr>
<td>Kaunas - Athens 1</td>
<td>Kaunas - Athens 2</td>
<td>Kaunas - Tel Aviv</td>
</tr>
<tr>
<td>Berlin - Athens 2</td>
<td>Berlin - Tel Aviv</td>
<td>Berlin - Athens 1</td>
</tr>
<tr>
<td>Ljubljana - Lyon</td>
<td>Ljubljana - Paris</td>
<td>Ljubljana - London</td>
</tr>
<tr>
<td>Zagreb - Paris</td>
<td>Zagreb - London</td>
<td>Zagreb - Lyon</td>
</tr>
<tr>
<td>Bamberg - London</td>
<td>Bamberg - Lyon</td>
<td>Bamberg - Paris</td>
</tr>
<tr>
<td>Moscow 1 - Belgrade</td>
<td>Moscow 1 - Istanbul 2</td>
<td>Moscow 1 - Istanbul 1</td>
</tr>
<tr>
<td>Moscow 2 - Istanbul 2</td>
<td>Moscow 2 - Istanbul 1</td>
<td>Moscow 2 - Belgrade</td>
</tr>
<tr>
<td>Vilnius - Istanbul 1</td>
<td>Vilnius - Belgrade</td>
<td>Vilnius - Istanbul 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Jornada 4</th>
<th>Jornada 5</th>
<th>Jornada 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lyon - Siena</td>
<td>Lyon - Milan</td>
<td>Lyon - Bologna</td>
</tr>
<tr>
<td>Paris - Milan</td>
<td>Paris - Bologna</td>
<td>Paris - Siena</td>
</tr>
<tr>
<td>London - Bologna</td>
<td>London - Siena</td>
<td>London - Milan</td>
</tr>
<tr>
<td>Istanbul 1 - Warsaw</td>
<td>Istanbul 1 - Berlin</td>
<td>Istanbul 1 - Kaunas</td>
</tr>
<tr>
<td>Istanbul 2 - Berlin</td>
<td>Istanbul 2 - Kaunas</td>
<td>Istanbul 2 - Warsaw</td>
</tr>
<tr>
<td>Belgrade - Kaunas</td>
<td>Belgrade - Warsaw</td>
<td>Belgrade - Berlin</td>
</tr>
<tr>
<td>Barcelona - Ljubljana</td>
<td>Barcelona - Bamberg</td>
<td>Barcelona - Zagreb</td>
</tr>
<tr>
<td>Madrid - Bamberg</td>
<td>Madrid - Zagreb</td>
<td>Madrid - Ljubljana</td>
</tr>
<tr>
<td>Vitoria - Zagreb</td>
<td>Vitoria - Ljubljana</td>
<td>Vitoria - Bamberg</td>
</tr>
<tr>
<td>Athens 1 - Moscow 2</td>
<td>Athens 1 - Moscow 1</td>
<td>Athens 1 - Vilnius</td>
</tr>
<tr>
<td>Athens 2 - Moscow 1</td>
<td>Athens 2 - Vilnius</td>
<td>Athens 2 - Moscow 2</td>
</tr>
<tr>
<td>Tel Aviv - Vilnius</td>
<td>Tel Aviv - Moscow 2</td>
<td>Tel Aviv - Moscow 1</td>
</tr>
</tbody>
</table>
### Jornada 7
- Vitoria - London
- Madrid - Paris
- Lyon - Barcelona
- Zagreb - Ljubljana
- Siena - Bamberg
- Milan - Bologna
- Belgrade - Istanbul 2
- Athens 2 - Istanbul 1
- Tel Aviv - Athens 1
- Kaunas - Berlin
- Vilnius - Warsaw
- Moscow 2 - Moscow 1

### Jornada 8
- London - Madrid
- Barcelona - Vitoria
- Paris - Lyon
- Ljubljana - Siena
- Bologna - Zagreb
- Bamberg - Milan
- Istanbul 2 - Athens 2
- Athens 1 - Belgrade
- Istanbul 1 - Tel Aviv
- Berlin - Vilnius
- Moscow 1 - Kaunas
- Warsaw - Moscow 2

### Jornada 9
- Barcelona - London
- Madrid - Lyon
- Vitoria - Paris
- Bologna - Ljubljana
- Siena - Milan
- Zagreb - Bamberg
- Athens 1 - Istanbul 2
- Athens 2 - Tel Aviv
- Belgrade - Istanbul 1
- Moscow 1 - Berlin
- Vilnius - Moscow 2
- Kaunas - Warsaw

### Jornada 10
- Lyon - London
- Paris - Barcelona
- Madrid - Vitoria
- Milan - Ljubljana
- Bamberg - Bologna
- Siena - Zagreb
- Tel Aviv - Istanbul 2
- Istanbul 1 - Athens 1
- Athens 2 - Belgrade
- Moscow 2 - Berlin
- Warsaw - Moscow 1
- Vilnius - Kaunas

### Jornada 11
- London - Paris
- Vitoria - Lyon
- Barcelona - Madrid
- Ljubljana - Bamberg
- Zagreb - Milan
- Bologna - Siena
- Istanbul 2 - Istanbul 1
- Belgrade - Tel Aviv
- Athens 1 - Athens 2
- Berlin - Warsaw
- Kaunas - Moscow 2
- Moscow 1 - Vilnius

### Jornada 12
- Istanbul 2 - Paris
- Istanbul 1 - Lyon
- Tel Aviv - Barcelona
- Athens 1 - London
- Athens 2 - Madrid
- Belgrade - Vitoria
- Berlin - Siena
- Warsaw - Ljubljana
- Moscow 2 - Bamberg
- Moscow 1 - Zagreb
- Vilnius - Bologna
- Kaunas - Milan

### Jornada 13
- Istanbul 2 - Lyon
- Istanbul 1 - Barcelona
- Tel Aviv - London
- Athens 1 - Madrid
- Athens 2 - Vitoria
- Belgrade - Paris
- Berlin - Ljubljana
- Warsaw - Bamberg
- Moscow 2 - Zagreb
- Moscow 1 - Bologna
- Vilnius - Milan
- Kaunas - Siena

### Jornada 14
- Istanbul 2 - Barcelona
- Istanbul 1 - London
- Tel Aviv - Madrid
- Athens 1 - Vitoria
- Athens 2 - Paris
- Belgrade - Lyon
- Berlin - Bamberg
- Warsaw - Zagreb
- Moscow 2 - Bologna
- Moscow 1 - Milan
- Vilnius - Siena
- Kaunas - Ljubljana

### Jornada 15
- Istanbul 2 - London
- Istanbul 1 - Madrid
- Tel Aviv - Vitoria
- Athens 1 - Paris
- Athens 2 - Lyon
- Belgrade - Barcelona
- Berlin - Zagreb
- Warsaw - Bologna
- Moscow 2 - Milan
- Moscow 1 - Siena
- Vilnius - Ljubljana
- Kaunas - Bamberg
<table>
<thead>
<tr>
<th>Jornada 16</th>
<th>Jornada 17</th>
<th>Jornada 18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Istanbul 2 - Madrid</td>
<td>Istanbul 2 - Vitoria</td>
<td>London - Moscow 1</td>
</tr>
<tr>
<td>Istanbul 1 - Vitoria</td>
<td>Istanbul 1 - Paris</td>
<td>Paris - Moscow 2</td>
</tr>
<tr>
<td>Tel Aviv - Paris</td>
<td>Tel Aviv - Lyon</td>
<td>Lyon - Vilnius</td>
</tr>
<tr>
<td>Athens 1 - Lyon</td>
<td>Athens 1 - Barcelona</td>
<td>Barcelona - Warsaw</td>
</tr>
<tr>
<td>Athens 2 - Barcelona</td>
<td>Athens 2 - London</td>
<td>Madrid - Berlin</td>
</tr>
<tr>
<td>Belgrade - London</td>
<td>Belgrade - Madrid</td>
<td>Vitoria - Kaunas</td>
</tr>
<tr>
<td>Berlin - Bologna</td>
<td>Berlin - Milan</td>
<td>Ljubljana - Athens 2</td>
</tr>
<tr>
<td>Warsaw - Milan</td>
<td>Warsaw - Siena</td>
<td>Bamberg - Belgrade</td>
</tr>
<tr>
<td>Moscow 2 - Siena</td>
<td>Moscow 2 - Ljubljana</td>
<td>Milan - Athens 1</td>
</tr>
<tr>
<td>Moscow 1 - Ljubljana</td>
<td>Moscow 1 - Bamberg</td>
<td>Bologna - Istanbul 2</td>
</tr>
<tr>
<td>Vilnius - Bamberg</td>
<td>Vilnius - Zagreb</td>
<td>Siena - Istanbul 1</td>
</tr>
<tr>
<td>Kaunas - Zagreb</td>
<td>Kaunas - Bologna</td>
<td>Zagreb - Tel Aviv</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Jornada 19</th>
<th>Jornada 20</th>
<th>Jornada 21</th>
</tr>
</thead>
<tbody>
<tr>
<td>London - Moscow 2</td>
<td>London - Vilnius</td>
<td>London - Warsaw</td>
</tr>
<tr>
<td>Paris - Vilnius</td>
<td>Paris - Warsaw</td>
<td>Paris - Berlin</td>
</tr>
<tr>
<td>Lyon - Warsaw</td>
<td>Lyon - Berlin</td>
<td>Lyon - Kaunas</td>
</tr>
<tr>
<td>Barcelona - Berlin</td>
<td>Barcelona - Kaunas</td>
<td>Barcelona - Moscow 1</td>
</tr>
<tr>
<td>Madrid - Kaunas</td>
<td>Madrid - Moscow 1</td>
<td>Madrid - Moscow 2</td>
</tr>
<tr>
<td>Vitoria - Moscow 1</td>
<td>Vitoria - Moz 2</td>
<td>Vitoria - Vilnius</td>
</tr>
<tr>
<td>Ljubljana - Belgrade</td>
<td>Ljubljana - Athens 1</td>
<td>Ljubljana - Istanbul 2</td>
</tr>
<tr>
<td>Bamberg - Athens 1</td>
<td>Bamberg - Istanbul 2</td>
<td>Bamberg - Istanbul 1</td>
</tr>
<tr>
<td>Milan - Istanbul 2</td>
<td>Milan - Istanbul 1</td>
<td>Milan - Tel Aviv</td>
</tr>
<tr>
<td>Bologna - Istanbul 1</td>
<td>Bologna - Tel Aviv</td>
<td>Bologna - Athens 2</td>
</tr>
<tr>
<td>Siena - Tel Aviv</td>
<td>Siena - Athens 2</td>
<td>Siena - Belgrade</td>
</tr>
<tr>
<td>Zagreb - Athens 2</td>
<td>Zagreb - Belgrade</td>
<td>Zagreb - Athens 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Jornada 22</th>
<th>Jornada 23</th>
<th>Jornada 24</th>
</tr>
</thead>
<tbody>
<tr>
<td>London - Berlin</td>
<td>London - Kaunas</td>
<td>London - Vitoria</td>
</tr>
<tr>
<td>Paris - Kaunas</td>
<td>Paris - Moscow 1</td>
<td>Paris - Madrid</td>
</tr>
<tr>
<td>Lyon - Moscow 1</td>
<td>Lyon - Moscow 2</td>
<td>Barcelona - Lyon</td>
</tr>
<tr>
<td>Barcelona - Moscow 2</td>
<td>Barcelona - Vilnius</td>
<td>Ljubljana - Zagreb</td>
</tr>
<tr>
<td>Madrid - Vilnius</td>
<td>Madrid - Warsaw</td>
<td>Bamberg - Siena</td>
</tr>
<tr>
<td>Vitoria - Warsaw</td>
<td>Vitoria - Berlin</td>
<td>Bologna - Milan</td>
</tr>
<tr>
<td>Ljubljana - Istanbul 1</td>
<td>Ljubljana - Tel Aviv</td>
<td>Istanbul 2 - Belgrade</td>
</tr>
<tr>
<td>Bamberg - Tel Aviv</td>
<td>Bamberg - Athens 2</td>
<td>Istanbul 1 - Athens 2</td>
</tr>
<tr>
<td>Milan - Athens 2</td>
<td>Milan - Belgrade</td>
<td>Athens 1 - Tel Aviv</td>
</tr>
<tr>
<td>Bologna - Belgrade</td>
<td>Bologna - Athens 1</td>
<td>Berlin - Kaunas</td>
</tr>
<tr>
<td>Siena - Athens 1</td>
<td>Siena - Istanbul 2</td>
<td>Warsaw - Vilnius</td>
</tr>
<tr>
<td>Zagreb - Istanbul 2</td>
<td>Zagreb - Istanbul 1</td>
<td>Moscow 1 - Moscow 2</td>
</tr>
<tr>
<td>Jornada 25</td>
<td>Jornada 26</td>
<td>Jornada 27</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>Vitoria - Barcelona</td>
<td>Lyon - Madrid</td>
<td>Barcelona - Paris</td>
</tr>
<tr>
<td>Lyon - Paris</td>
<td>Paris - Vitoria</td>
<td>Vitoria - Madrid</td>
</tr>
<tr>
<td>Siena - Ljubljana</td>
<td>Ljubljana - Bologna</td>
<td>Ljubljana - Milan</td>
</tr>
<tr>
<td>Zagreb - Bologna</td>
<td>Milan - Siena</td>
<td>Bologna - Milan</td>
</tr>
<tr>
<td>Milan - Bamberg</td>
<td>Bamberg - Zagreb</td>
<td>Zagreb - Siena</td>
</tr>
<tr>
<td>Athens 2 - Istanbul 2</td>
<td>Istanbul 2 - Athens 1</td>
<td>Istanbul 2 - Tel Aviv</td>
</tr>
<tr>
<td>Belgrade - Athens 1</td>
<td>Tel Aviv - Athens 2</td>
<td>Athens 1 - Istanbul 1</td>
</tr>
<tr>
<td>Tel Aviv - Istanbul 1</td>
<td>Istanbul 1 - Belgrade</td>
<td>Belgrade - Athens 2</td>
</tr>
<tr>
<td>Vilnius - Berlin</td>
<td>Berlin - Moscow 1</td>
<td>Berlin - Moscow 2</td>
</tr>
<tr>
<td>Kaunas - Moscow 1</td>
<td>Moscow 2 - Vilnius</td>
<td>Moscow 1 - Warsaw</td>
</tr>
<tr>
<td>Moscow 2 - Warsaw</td>
<td>Warsaw - Kaunas</td>
<td>Kaunas - Vilnius</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Jornada 28</th>
<th>Jornada 29</th>
<th>Jornada 30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paris - London</td>
<td>Barcelona - Siena</td>
<td>Barcelona - Milan</td>
</tr>
<tr>
<td>Lyon - Vitoria</td>
<td>Madrid - Milan</td>
<td>Madrid - Bologna</td>
</tr>
<tr>
<td>Madrid - Barcelona</td>
<td>Vitoria - Bologna</td>
<td>Vitoria - Siena</td>
</tr>
<tr>
<td>Bamberg - Ljubljana</td>
<td>Athens 1 - Berlin</td>
<td>Athens 1 - Kaunas</td>
</tr>
<tr>
<td>Milan - Zagreb</td>
<td>Athens 2 - Kaunas</td>
<td>Athens 2 - Warsaw</td>
</tr>
<tr>
<td>Siena - Bologna</td>
<td>Tel Aviv - Warsaw</td>
<td>Tel Aviv - Berlin</td>
</tr>
<tr>
<td>Istanbul 1 - Istanbul 2</td>
<td>Lyon - Zagreb</td>
<td>Lyon - Bamberg</td>
</tr>
<tr>
<td>Tel Aviv - Belgrade</td>
<td>Paris - Bamberg</td>
<td>Paris - Ljubljana</td>
</tr>
<tr>
<td>Athens 2 - Athens 1</td>
<td>London - Ljubljana</td>
<td>London - Zagreb</td>
</tr>
<tr>
<td>Warsaw - Berlin</td>
<td>Istanbul 1 - Vilnius</td>
<td>Istanbul 1 - Moscow 2</td>
</tr>
<tr>
<td>Moscow 2 - Kaunas</td>
<td>Istanbul 2 - Moscow 2</td>
<td>Istanbul 2 - Moscow 1</td>
</tr>
<tr>
<td>Vilnius - Moscow 1</td>
<td>Belgrade - Moscow 1</td>
<td>Belgrade - Vilnius</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Jornada 31</th>
<th>Jornada 32</th>
<th>Jornada 33</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barcelona - Bologna</td>
<td>Ljubljana - Barcelona</td>
<td>Ljubljana - Vitoria</td>
</tr>
<tr>
<td>Madrid - Siena</td>
<td>Zagreb - Vitoria</td>
<td>Zagreb - Madrid</td>
</tr>
<tr>
<td>Vitoria - Milan</td>
<td>Bamberg - Madrid</td>
<td>Bamberg - Barcelona</td>
</tr>
<tr>
<td>Athens 1 - Warsaw</td>
<td>Moscow 1 - Athens 1</td>
<td>Moscow 1 - Tel Aviv</td>
</tr>
<tr>
<td>Athens 2 - Berlin</td>
<td>Moscow 2 - Tel Aviv</td>
<td>Moscow 2 - Athens 2</td>
</tr>
<tr>
<td>Tel Aviv - Kaunas</td>
<td>Vilnius - Athens 2</td>
<td>Vilnius - Athens 1</td>
</tr>
<tr>
<td>Lyon - Ljubljana</td>
<td>Bologna - London</td>
<td>Bologna - Paris</td>
</tr>
<tr>
<td>Paris - Zagreb</td>
<td>Milan - Paris</td>
<td>Milan - Lyon</td>
</tr>
<tr>
<td>London - Bamberg</td>
<td>Siena - Lyon</td>
<td>Siena - London</td>
</tr>
<tr>
<td>Istanbul 1 - Moscow 1</td>
<td>Warsaw - Istanbul 2</td>
<td>Warsaw - Belgrade</td>
</tr>
<tr>
<td>Istanbul 2 - Vilnius</td>
<td>Kaunas - Belgrade</td>
<td>Kaunas - Istanbul 1</td>
</tr>
<tr>
<td>Belgrade - Moscow 2</td>
<td>Berlin - Istanbul 1</td>
<td>Belgrade - Istanbul 2</td>
</tr>
</tbody>
</table>
Conclusions

<table>
<thead>
<tr>
<th>Jornada 34</th>
<th>Jornada 35</th>
<th>Jornada 36</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ljubljana - Madrid</td>
<td>Bologna - Vitoria</td>
<td>Bologna - Madrid</td>
</tr>
<tr>
<td>Zagreb - Barcelona</td>
<td>Milan - Madrid</td>
<td>Milan - Barcelona</td>
</tr>
<tr>
<td>Bamberg - Vitoria</td>
<td>Siena - Barcelona</td>
<td>Siena - Vitoria</td>
</tr>
<tr>
<td>Moscow 1 - Athens 2</td>
<td>Warsaw - Tel Aviv</td>
<td>Warsaw - Athens 1</td>
</tr>
<tr>
<td>Moscow 2 - Athens 1</td>
<td>Kaunas - Athens 1</td>
<td>Kaunas - Athens 2</td>
</tr>
<tr>
<td>Vilnius - Tel Aviv</td>
<td>Berlin - Athens 2</td>
<td>Berlin - Tel Aviv</td>
</tr>
<tr>
<td>Bologna - Lyon</td>
<td>Ljubljana - Lyon</td>
<td>Ljubljana - Paris</td>
</tr>
<tr>
<td>Milan - London</td>
<td>Zagreb - Paris</td>
<td>Zagreb - London</td>
</tr>
<tr>
<td>Siena - Paris</td>
<td>Bamberg - London</td>
<td>Bamberg - Lyon</td>
</tr>
<tr>
<td>Warsaw - Istanbul 1</td>
<td>Moscow 1 - Belgrade</td>
<td>Moscow 1 - Istanbul 2</td>
</tr>
<tr>
<td>Kaunas - Istanbul 1</td>
<td>Moscow 2 - Istanbul 2</td>
<td>Moscow 2 - Istanbul 1</td>
</tr>
<tr>
<td>Berlin - Belgrade</td>
<td>Vilnius - Istanbul 1</td>
<td>Vilnius - Belgrade</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Jornada 37</th>
<th>Jornada 38</th>
<th>Jornada 39</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bologna - Barcelona</td>
<td>Lyon - Siena</td>
<td>Lyon - Milan</td>
</tr>
<tr>
<td>Milan - Vitoria</td>
<td>Paris - Milan</td>
<td>Paris - Bologna</td>
</tr>
<tr>
<td>Siena - Madrid</td>
<td>London - Bologna</td>
<td>London - Siena</td>
</tr>
<tr>
<td>Warsaw - Athens 2</td>
<td>Istanbul 1 - Warsaw</td>
<td>Istanbul 1 - Berlin</td>
</tr>
<tr>
<td>Kaunas - Tel Aviv</td>
<td>Istanbul 2 - Berlin</td>
<td>Istanbul 2 - Kaunas</td>
</tr>
<tr>
<td>Berlin - Athens 1</td>
<td>Belgrade - Kaunas</td>
<td>Belgrade - Warsaw</td>
</tr>
<tr>
<td>Ljubljana - London</td>
<td>Barcelona - Ljubljana</td>
<td>Barcelona - Bamberg</td>
</tr>
<tr>
<td>Zagreb - Lyon</td>
<td>Madrid - Bamberg</td>
<td>Madrid - Zagreb</td>
</tr>
<tr>
<td>Bamberg - Paris</td>
<td>Vitoria - Zagreb</td>
<td>Vitoria - Ljubljana</td>
</tr>
<tr>
<td>Moscow 1 - Istanbul 1</td>
<td>Athens 1 - Moscow 2</td>
<td>Athens 1 - Moscow 1</td>
</tr>
<tr>
<td>Moscow 2 - Belgrade</td>
<td>Athens 2 - Moscow 1</td>
<td>Athens 2 - Vilnius</td>
</tr>
<tr>
<td>Vilnius - Istanbul 2</td>
<td>Tel Aviv - Vilnius</td>
<td>Tel Aviv - Moscow 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Jornada 40</th>
<th>Jornada 41</th>
<th>Jornada 42</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lyon - Bologna</td>
<td>Vitoria - London</td>
<td>London - Madrid</td>
</tr>
<tr>
<td>Paris - Siena</td>
<td>Madrid - Paris</td>
<td>Barcelona - Vitoria</td>
</tr>
<tr>
<td>London - Milan</td>
<td>Lyon - Barcelona</td>
<td>Paris - Lyon</td>
</tr>
<tr>
<td>Istanbul 1 - Kaunas</td>
<td>Zagreb - Ljubljana</td>
<td>Ljubljana - Siena</td>
</tr>
<tr>
<td>Istanbul 2 - Warsaw</td>
<td>Siena - Bamberg</td>
<td>Bologna - Zagreb</td>
</tr>
<tr>
<td>Belgrade - Berlin</td>
<td>Milan - Bologna</td>
<td>Bamberg - Milan</td>
</tr>
<tr>
<td>Barcelona - Zagreb</td>
<td>Belgrade - Istanbul 2</td>
<td>Istanbul 2 - Athens 2</td>
</tr>
<tr>
<td>Madrid - Ljubljana</td>
<td>Athens 2 - Istanbul 1</td>
<td>Athens 1 - Belgrade</td>
</tr>
<tr>
<td>Vitoria - Bamberg</td>
<td>Tel Aviv - Athens 1</td>
<td>Istanbul 1 - Tel Aviv</td>
</tr>
<tr>
<td>Athens 1 - Vilnius</td>
<td>Kaunas - Berlin</td>
<td>Berlin - Vilnius</td>
</tr>
<tr>
<td>Athens 2 - Moscow 2</td>
<td>Vilnius - Warsaw</td>
<td>Moscow 1 - Kaunas</td>
</tr>
<tr>
<td>Tel Aviv - Moscow 1</td>
<td>Moscow 2 - Moscow 1</td>
<td>Warsaw - Moscow 2</td>
</tr>
</tbody>
</table>
### Jornada 43
- Barcelona - London
- Madrid - Lyon
- Vitoria - Paris
- Bologna - Ljubljana
- Siena - Milan
- Zagreb - Bamberg
- Athens 1 - Istanbul 2
- Athens 2 - Tel Aviv
- Belgrade - Istanbul 1
- Moscow 1 - Berlin
- Vilnius - Moscow 2
- Kaunas - Warsaw

### Jornada 44
- Lyon - London
- Paris - Barcelona
- Madrid - Vitoria
- Milan - Ljubljana
- Bamberg - Bologna
- Siena - Zagreb
- Tel Aviv - Istanbul 2
- Istanbul 1 - Athens 1
- Athens 2 - Belgrade
- Moscow 2 - Berlin
- Warsaw - Moscow 1
- Vilnius - Kaunas

### Jornada 45
- London - Paris
- Vitoria - Lyon
- Barcelona - Madrid
- Ljubljana - Bamberg
- Zagreb - Milan
- Bologna - Siena
- Istanbul 2 - Istanbul 1
- Belgrade - Tel Aviv
- Athens 1 - Athens 2
- Berlin - Warsaw
- Kaunas - Moscow 2
- Moscow 1 - Vilnius

### Jornada 46
- London - Istanbul 2
- Paris - Istanbul 1
- Lyon - Tel Aviv
- Barcelona - Athens 2
- Madrid - Belgrade
- Vitoria - Athens 1
- Ljubljana - Moscow 1
- Bamberg - Vilnius
- Milan - Warsaw
- Bologna - Berlin
- Siena - Kaunas
- Zagreb - Moscow 2

### Jornada 47
- London - Istanbul 1
- Paris - Tel Aviv
- Lyon - Athens 2
- Barcelona - Belgrade
- Madrid - Athens 1
- Vitoria - Istanbul 2
- Ljubljana - Vilnius
- Bamberg - Warsaw
- Milan - Berlin
- Bologna - Kaunas
- Siena - Moscow 2
- Zagreb - Mosc 1

### Jornada 48
- London - Tel Aviv
- Paris - Athens 2
- Lyon - Belgrade
- Barcelona - Athens 1
- Madrid - Istanbul 2
- Vitoria - Istanbul 1
- Ljubljana - Warsaw
- Bamberg - Berlin
- Milan - Kaunas
- Bologna - Moscow 2
- Siena - Moscow 1
- Zagreb - Vilnius

### Jornada 49
- London - Athens 2
- Paris - Belgrade
- Lyon - Athens 1
- Barcelona - Istanbul 2
- Madrid - Istanbul 1
- Vitoria - Tel Aviv
- Ljubljana - Berlin
- Bamberg - Kaunas
- Milan - Moscow 2
- Bologna - Moscow 1
- Siena - Vilnius
- Zagreb - Warsaw

### Jornada 50
- London - Belgrade
- Paris - Athens 1
- Lyon - Istanbul 2
- Barcelona - Istanbul 1
- Madrid - Tel Aviv
- Vitoria - Athens 2
- Ljubljana - Kaunas
- Bamberg - Moscow 2
- Milan - Moscow 1
- Bologna - Vilnius
- Siena - Warsaw
- Zagreb - Berlin

### Jornada 51
- London - Athens 1
- Paris - Istanbul 2
- Lyon - Istanbul 1
- Barcelona - Tel Aviv
- Madrid - Athens 2
- Vitoria - Belgrade
- Ljubljana - Moscow 2
- Bamberg - Moscow 1
- Milan - Vilnius
- Bologna - Warsaw
- Siena - Berlin
- Zagreb - Kaunas
<table>
<thead>
<tr>
<th>Jornada 52</th>
<th>Jornada 53</th>
<th>Jornada 54</th>
</tr>
</thead>
<tbody>
<tr>
<td>Istanbul 2 - Siena</td>
<td>Istanbul 2 - Zagreb</td>
<td>Istanbul 2 - Ljubljana</td>
</tr>
<tr>
<td>Istanbul 1 - Zagreb</td>
<td>Istanbul 1 - Ljubljana</td>
<td>Istanbul 1 - Bologna</td>
</tr>
<tr>
<td>Tel Aviv - Ljubljana</td>
<td>Tel Aviv - Bologna</td>
<td>Tel Aviv - Bamberg</td>
</tr>
<tr>
<td>Athens 1 - Bologna</td>
<td>Athens 1 - Bamberg</td>
<td>Athens 1 - Milan</td>
</tr>
<tr>
<td>Athens 2 - Bamberg</td>
<td>Athens 2 - Milan</td>
<td>Athens 2 - Siena</td>
</tr>
<tr>
<td>Belgrade - Milan</td>
<td>Belgrade - Siena</td>
<td>Belgrade - Zagreb</td>
</tr>
<tr>
<td>Berlin - Madrid</td>
<td>Berlin - Lyon</td>
<td>Berlin - Paris</td>
</tr>
<tr>
<td>Moscow 2 - Paris</td>
<td>Moscow 2 - London</td>
<td>Moscow 2 - Barcelona</td>
</tr>
<tr>
<td>Moscow 1 - London</td>
<td>Moscow 1 - Barcelone</td>
<td>Moscow 1 - Vitoria</td>
</tr>
<tr>
<td>Vilnius - Barcelone</td>
<td>Vilnius - Vitoria</td>
<td>Vilnius - Madrid</td>
</tr>
<tr>
<td>Kaunas - Vitoria</td>
<td>Kaunas - Madrid</td>
<td>Kaunas - Lyon</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Jornada 55</th>
<th>Jornada 56</th>
<th>Jornada 57</th>
</tr>
</thead>
<tbody>
<tr>
<td>Istanbul 2 - Bologna</td>
<td>Istanbul 2 - Bamberg</td>
<td>Istanbul 2 - Milan</td>
</tr>
<tr>
<td>Istanbul 1 - Bamberg</td>
<td>Istanbul 1 - Milan</td>
<td>Istanbul 1 - Siena</td>
</tr>
<tr>
<td>Tel Aviv - Milan</td>
<td>Tel Aviv - Siena</td>
<td>Tel Aviv - Zagreb</td>
</tr>
<tr>
<td>Athens 1 - Siena</td>
<td>Athens 1 - Zagreb</td>
<td>Athens 1 - Ljubljana</td>
</tr>
<tr>
<td>Athens 2 - Zagreb</td>
<td>Athens 2 - Ljubljana</td>
<td>Athens 2 - Bologna</td>
</tr>
<tr>
<td>Belgrade - Ljubljana</td>
<td>Belgrade - Bologna</td>
<td>Belgrade - Bamberg</td>
</tr>
<tr>
<td>Berlin - London</td>
<td>Berlin - Barcelona</td>
<td>Berlin - Vitoria</td>
</tr>
<tr>
<td>Warsaw - Barcelona</td>
<td>Warsaw - Vitoria</td>
<td>Warsaw - Madrid</td>
</tr>
<tr>
<td>Moscow 2 - Vitoria</td>
<td>Moscow 2 - Madrid</td>
<td>Moscow 2 - Lyon</td>
</tr>
<tr>
<td>Moscow 1 - Madrid</td>
<td>Moscow 1 - Lyon</td>
<td>Moscow 1 - Paris</td>
</tr>
<tr>
<td>Vilnius - Lyon</td>
<td>Vilnius - Paris</td>
<td>Vilnius - London</td>
</tr>
<tr>
<td>Kaunas - Paris</td>
<td>Kaunas - London</td>
<td>Kaunas - Barcelone</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Jornada 58</th>
<th>Jornada 59</th>
<th>Jornada 60</th>
</tr>
</thead>
<tbody>
<tr>
<td>London - Vitoria</td>
<td>Madrid - London</td>
<td>London - Barcelona</td>
</tr>
<tr>
<td>Paris - Madrid</td>
<td>Vitoria - Barcelona</td>
<td>Lyon - Madrid</td>
</tr>
<tr>
<td>Barcelone - Lyon</td>
<td>Lyon - Paris</td>
<td>Paris - Vitoria</td>
</tr>
<tr>
<td>Ljubljana - Zagreb</td>
<td>Siena - Ljubljana</td>
<td>Ljubljana - Bologna</td>
</tr>
<tr>
<td>Bamberg - Siena</td>
<td>Zagreb - Bologna</td>
<td>Milan - Siena</td>
</tr>
<tr>
<td>Bologna - Milan</td>
<td>Milan - Bamberg</td>
<td>Bamberg - Zagreb</td>
</tr>
<tr>
<td>Istanbul 2 - Belgrade</td>
<td>Athens 2 - Istanbul 2</td>
<td>Istanbul 2 - Athens 1</td>
</tr>
<tr>
<td>Istanbul 1 - Athens 2</td>
<td>Belgrade - Athens 1</td>
<td>Tel Aviv - Athens 2</td>
</tr>
<tr>
<td>Athens 1 - Tel Aviv</td>
<td>Tel Aviv - Istanbul 1</td>
<td>Istanbul 1 - Belgrade</td>
</tr>
<tr>
<td>Berlin - Kaunas</td>
<td>Vilnius - Berlin</td>
<td>Berlin - Moscow 1</td>
</tr>
<tr>
<td>Warsaw - Vilnius</td>
<td>Kaunas - Moscow 1</td>
<td>Moscow 2 - Vilnius</td>
</tr>
<tr>
<td>Moscow 1 - Moscow 2</td>
<td>Moscow 2 - Warsaw</td>
<td>Warsaw - Kaunas</td>
</tr>
<tr>
<td>Jornada 61</td>
<td>Jornada 62</td>
<td>Jornada 63</td>
</tr>
<tr>
<td>------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>London - Lyon</td>
<td>Paris - London</td>
<td>Barcelona - Siena</td>
</tr>
<tr>
<td>Barcelona - Paris</td>
<td>Lyon - Vitoria</td>
<td>Madrid - Milan</td>
</tr>
<tr>
<td>Vitoria - Madrid</td>
<td>Madrid - Barcelona</td>
<td>Vitoria - Bologna</td>
</tr>
<tr>
<td>Ljubljana - Milan</td>
<td>Bamberg - Ljubljana</td>
<td>Athens 1 - Berlin</td>
</tr>
<tr>
<td>Bologna - Bamberg</td>
<td>Milan - Zagreb</td>
<td>Athens 2 - Kaunas</td>
</tr>
<tr>
<td>Zagreb - Siena</td>
<td>Siena - Bologna</td>
<td>Tel Aviv - Warsaw</td>
</tr>
<tr>
<td>Istanbul 2 - Tel Aviv</td>
<td>Istanbul 1 - Istanbul 2</td>
<td>Lyon - Zagreb</td>
</tr>
<tr>
<td>Athens 1 - Istanbul 1</td>
<td>Tel Aviv - Belgrade</td>
<td>Paris - Bamberg</td>
</tr>
<tr>
<td>Belgrade - Athens 2</td>
<td>Athens 2 - Athens 1</td>
<td>London - Ljubljana</td>
</tr>
<tr>
<td>Berlin - Moscow 2</td>
<td>Warsaw - Berlin</td>
<td>Istanbul 1 - Vilnius</td>
</tr>
<tr>
<td>Moscow 1 - Warsaw</td>
<td>Moscow 2 - Kaunas</td>
<td>Istanbul 2 - Moscow 2</td>
</tr>
<tr>
<td>Kaunas - Vilnius</td>
<td>Vilnius - Moscow 1</td>
<td>Belgrade - Moscow 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Jornada 64</th>
<th>Jornada 65</th>
<th>Jornada 66</th>
</tr>
</thead>
<tbody>
<tr>
<td>Barcelona - Milan</td>
<td>Barcelona - Bologna</td>
<td>Ljubljana - Barcelona</td>
</tr>
<tr>
<td>Madrid - Bologna</td>
<td>Madrid - Siena</td>
<td>Zagreb - Vitoria</td>
</tr>
<tr>
<td>Vitoria - Siena</td>
<td>Vitoria - Milan</td>
<td>Bamberg - Madrid</td>
</tr>
<tr>
<td>Athens 1 - Kaunas</td>
<td>Athens 1 - Warsaw</td>
<td>Moscow 1 - Athens 1</td>
</tr>
<tr>
<td>Athens 2 - Warsaw</td>
<td>Athens 2 - Berlin</td>
<td>Moscow 2 - Tel Aviv</td>
</tr>
<tr>
<td>Tel Aviv - Berlin</td>
<td>Tel Aviv - Kaunas</td>
<td>Vilnius - Athens 2</td>
</tr>
<tr>
<td>Lyon - Bamberg</td>
<td>Lyon - Ljubljana</td>
<td>Bologna - London</td>
</tr>
<tr>
<td>Paris - Ljubljana</td>
<td>Paris - Zagreb</td>
<td>Milan - Paris</td>
</tr>
<tr>
<td>London - Zagreb</td>
<td>London - Bamberg</td>
<td>Siena - Lyon</td>
</tr>
<tr>
<td>Istanbul 1 - Moscow 2</td>
<td>Istanbul 1 - Moscow 1</td>
<td>Warsaw - Istanbul 2</td>
</tr>
<tr>
<td>Istanbul 2 - Moscow 1</td>
<td>Istanbul 2 - Vilnius</td>
<td>Kaunas - Belgrade</td>
</tr>
<tr>
<td>Belgrade - Vilnius</td>
<td>Belgrade - Moscow 2</td>
<td>Berlin - Istanbul 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Jornada 67</th>
<th>Jornada 68</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ljubljana - Vitoria</td>
<td>Ljubljana - Madrid</td>
</tr>
<tr>
<td>Zagreb - Madrid</td>
<td>Zagreb - Barcelona</td>
</tr>
<tr>
<td>Bamberg - Barcelona</td>
<td>Bamberg - Vitoria</td>
</tr>
<tr>
<td>Moscow 1 - Tel Aviv</td>
<td>Moscow 1 - Athens 2</td>
</tr>
<tr>
<td>Moscow 2 - Athens 2</td>
<td>Moscow 2 - Athens 1</td>
</tr>
<tr>
<td>Vilnius - Athens 1</td>
<td>Vilnius - Tel Aviv</td>
</tr>
<tr>
<td>Bologna - Paris</td>
<td>Bologna - Lyon</td>
</tr>
<tr>
<td>Milan - Lyon</td>
<td>Milan - London</td>
</tr>
<tr>
<td>Siena - London</td>
<td>Siena - Paris</td>
</tr>
<tr>
<td>Warsaw - Belgrade</td>
<td>Warsaw - Istanbul 1</td>
</tr>
<tr>
<td>Kaunas - Istanbul 1</td>
<td>Kaunas - Istanbul 2</td>
</tr>
<tr>
<td>Berlin - Istanbul 2</td>
<td>Berlin - Belgrade</td>
</tr>
</tbody>
</table>
5.2 Possible improvements

We believe that the obtained schedule accomplishes its objectives, but there is some improvements to be considered:

- Explore what would happen if one division had more teams than others.

- Probably, a disadvantage of the final schedule is that road trip and home stands are too long (6 matches). One option would be to have 6 divisions each one of 4 teams. Then, they would consist only of 4 matches.

- Think about possible options for the second phase:
  - Would it be played with the play-off system?
  - If there is a play-off, how many matches would have every serie? Would the teams be arranged geographically or according to their previous classification?
  - Would it be a final match, a final four, or just play-off until the end?

- Decide if it would be a closed competition (always the same teams, without relegations and upgrades) or an open one (with relegations and upgrades) and, if so, how many teams would be affected and which method would be used.
Bibliography


[9] K. Kinder, Graph Theory as it relates to Sport Scheduling., <kkinder22.files.wordpress.com/2007/06/graph-theory-as-it-relates-to-scheduling>


Appendix A

Annex

A.1 Program *kmeans*

```python
import random
from math import *

def disesf(v1,v2):
    a=90-v1[0]
    b=90-v2[0]
    P=v1[1]-v2[1]
    ad=radians(a)
    bd=radians(b)
    Pd=radians(P)
    cosdis=cos(ad)*cos(bd)+sin(ad)*sin(bd)*cos(Pd)
    dis=degrees(acos(cosdis))
    distancia=dis*4000/36
    return distancia

c = open("ciutats.txt","r")
lines=[line for line in c]
c.close()
pos=[]
for line in lines:
    col=line.split(',

```
x = int(col[1]) + int(col[2]) / 60 + int(col[3]) / 3600
y = int(col[4]) + int(col[5]) / 60 + int(col[6]) / 3600
pos.append([col[0], x, y])

# Busquem el màxim i el mínim de cada coordenada
minx = 1000; miny = 1000
maxx = 0; maxy = 0
for i in range(len(pos)):
    if pos[i][1] < minx:
        minx = (pos[i][1])
    if pos[i][1] > maxx:
        maxx = (pos[i][1])
    if pos[i][2] < miny:
        miny = (pos[i][2])
    if pos[i][2] > maxy:
        maxy = (pos[i][2])
ranges = [minx, maxx, miny, maxy]
print(ranges)

# Creem 4 centroids aleatoris dintre del rang
clusters = []
k = 4
for i in range(k):
    # clx = ranges[0] + random.random() * (ranges[1] - ranges[0])
    if i == 0 or i == 2:
        # clx = ranges[0] + (ranges[1] - ranges[0]) * (i+1)/4
    if i == 1 or i == 3:
        clx = ranges[0] + (ranges[1] - ranges[0]) * i/4
clx = ranges[0] + (ranges[1] - ranges[0]) * (i+1)/4
clusters.append([clx, cly])
print(clusters)
# Program kmeans

```python
lastmatches=None
for t in range(5):
    print('Iteracio %d' % t)
    bestmatches=[[0] for i in range(k)]
    for j in range(len(pos)):
        row=pos[j]
        bestmatch=0
        v2=[row[1],row[2]]
        for i in range(k):
            v1=clusters[i]
            d=disesf(v1,v2)
            if d<disesf(clusters[bestmatch],v2):
                bestmatch=i
        bestmatches[bestmatch].append(j)

    # If the results are the same as last time, this is complete
    if bestmatches==lastmatches:
        break
    lastmatches=bestmatches

    # Move the centroids to the average of their members
    ca=[]
    for i in range(k):
        avgs=[0.0]*2
        mx=0
        my=0
        print(bestmatches[i])
        if len(bestmatches[i])>0:
            Lp=len(bestmatches[i])
            for rowid in bestmatches[i]:
                print(rowid)
                print(pos[rowid])
                ca.append(pos[rowid])
                avgs[0]+=int(pos[rowid][1])
                avgs[1]+=int(pos[rowid][2])
                mx=avgs[0]/len(bestmatches[i])
                my=avgs[1]/len(bestmatches[i])
```

```python
clusters[i][0]=mx
clusters[i][1]=my

print(clusters[i])
fca = open("ciutatsA.txt","w+")
fcb = open("ciutatsB.txt","w+")
fcc = open("ciutatsC.txt","w+")
fcd = open("ciutatsD.txt","w+")
L0=len(bestmatches[0])
L1=len(bestmatches[1])
L2=len(bestmatches[2])
L3=len(bestmatches[3])

for q in range(L0):
    ciutat=ca[q][0]+','+str(ca[q][1])+','+str(ca[q][2])
    fca.write(ciutat)
    fca.write("\n")

for q in range(L0,L0+L1):
    ciutat=ca[q][0]+','+str(ca[q][1])+','+str(ca[q][2])
    fcb.write(ciutat)
    fcb.write("\n")

for q in range(L0+L1,L0+L1+L2):
    ciutat=ca[q][0]+','+str(ca[q][1])+','+str(ca[q][2])
    fcc.write(ciutat)
    fcc.write("\n")

for q in range(L0+L1+L2,L0+L1+L2+L3):
    ciutat=ca[q][0]+','+str(ca[q][1])+','+str(ca[q][2])
    fcd.write(ciutat)
    fcd.write("\n")

fca.close()
fcb.close()
fcc.close()
fcd.close()
```
A.2 Program *modifica*

```python
from math import *

def disesf(v1,v2):
a=90-v1[0]
b=90-v2[0]
P=v1[1]-v2[1]
ad=radians(a)
bd=radians(b)
Pd=radians(P)

    cosdis=cos(ad)*cos(bd)+sin(ad)*sin(bd)*cos(Pd)
    dis=degrees(acos(cosdis))
    distancia=dis*1000/9
    return distancia

def falta(i,n,posi,mxf,h):
avgs=[0.0]*2
mxf=0
myf=0
for j in range(n):
    avgs[h]+=int(posi[j][h+1])
    avgs[h+1]+=int(posi[j][h+2])

mxf=avgs[h]/n
myf=avgs[h+1]/n
return mxf,myf

def sobra(i,n,posi):
d=[]
for h in range(len(mx)):
v2=(mx[h][1][0],mx[h][1][1])
for j in range(n):
v1=(posi[j][1],posi[j][2])
    dis=disesf(v1,v2)
    d.append(dis)
ng=int(len(d)/n)
```
for p in range(ng):
    dmin=7000
    cmin=[]

for q in range(0,8):
    if d[p*8+q]<dmin and posi[q]!=[]:
        dmin=d[p*8+q]
        c= posi[q][0]
        cx= posi[q][1]
        cy= posi[q][2]
        cm=q
        if mx[p][0]==0:
            posA.append([c,cx,cy])
        if mx[p][0]==1:
            posB.append([c,cx,cy])
        if mx[p][0]==2:
            posC.append([c,cx,cy])
        if mx[p][0]==3:
            posD.append([c,cx,cy])

if i==0:
    del posA[cm][0:3]
if i==1:
    del posB[cm][0:3]
if i==2:
    del posC[cm][0:3]
if i==3:
    del posD[cm][0:3]
    del posi[cm][0:3]
return posi

L=[]
fa = open("ciutatsA.txt","r")
lines=[line for line in fa]
fa.close()
posA=[]
for line in lines:
col=line.split(',,')
x=float(col[1])
y=float(col[2])
    posA.append([col[0],x,y])
L.append(len(posA))
fb = open("ciutatsB.txt","r")
lines=[line for line in fb]
fb.close()
posB=[]
for line in lines:
    col=line.split(',,')
x=float(col[1])
y=float(col[2])
    posB.append([col[0],x,y])
L.append(len(posB))
fcc = open("ciutatsC.txt","r")
lines=[line for line in fc]
fcc.close()
posC=[]
for line in lines:
    col=line.split(',,')
x=float(col[1])
y=float(col[2])
    posC.append([col[0],x,y])
L.append(len(posC))
fd = open("ciutatsD.txt","r")
lines=[line for line in fd]
fd.close()
posD=[]
for line in lines:
    col=line.split(',,')
x=float(col[1])
y=float(col[2])
    posD.append([col[0],x,y])
L.append(len(posD))

mx=[]
my=[]
for i in range(len(L)):
    posi=[]
    n=L[i]
    mxf=0
    if i==0:
        posi=posA
    if i==1:
        posi=posB
    if i==2:
        posi=posC
    if i==3:
        posi=posD
    if L[i]<6:
        mf=falta(i,n,posi,mxf,0)
        mx.append([i,mf])
for i in range(len(L)):
    posi=[]
    n=L[i]
    if i==0:
        posi=posA
    if i==1:
        posi=posB
    if i==2:
        posi=posC
    if i==3:
        posi=posD
    if L[i]>6:
        mf=sobra(i,n,posi)

fca = open("ciutatsA.txt","w+")
for q in range(len(posA)):
    ciutat=posA[q][0]+','+str(posA[q][1])+','+str(posA[q][2])
    fca.write(ciutat)
    fca.write("\n")
fca.close()
A.3 Program camihamiltonia

from math import *
import networkx as nx

def disesf(v1,v2):
    a=90-v1[0]
    b=90-v2[0]
    P=v1[1]-v2[1]
    ad=radians(a)
    bd=radians(b)
    Pd=radians(P)
    cosdis=cos(ad)*cos(bd)+sin(ad)*sin(bd)*cos(Pd)
dis=degrees(acos(cosdis))
distancia=dis*1000/9
return distancia

def distgrup(grup):
    G=nx.Graph()
    if grup=="A":
        fitxer="ciutatsA.txt"
    if grup=="B":
        fitxer="ciutatsB.txt"
    if grup=="C":
        fitxer="ciutatsC.txt"
    if grup=="D":
        fitxer="ciutatsD.txt"

    fc = open(fitxer,"r")
    lines=[line for line in fc]
    fc.close()
    pos=[]
    for line in lines:
        col=line.split(′,′)
        x=float(col[1])
        y=float(col[2])
        pos.append([col[0],x,y])
    L=len(pos)
    for j in range(L-1):
        for h in range(j+1,L):
            v2=(pos[j][1],pos[j][2])
            v1=(pos[h][1],pos[h][2])
            d=disesf(v1,v2)
            ciutat1=pos[j][0]
            ciutat2=pos[h][0]
            G.add_edge(ciutat1,ciutat2,weight=d)

    L=len(G.nodes())
    distances=[]
    for i in range(0,6):
        for path in nx.all_simple_paths(G, source=G.nodes()[i], target=G.nodes())
LP = len(path)
if LP == 7:
    distance = 0
    for i in range(0, 6):
        distance = distance + G[path[i]][path[i+1]]['weight']
    distances.append((distance, path))
min_dist = distances[0][0];
min_path = []

# find the min distance
for dist in distances:
    if dist[0] <= min_dist:
        min_dist = dist[0]
        min_path = dist[1]

print(min_path)
print("Distancia total")
print(min_dist)

poss = []
for line in lines:
    col = line.split(', ,
    nom = col[0]
    lat = str(col[1])
    lon = str(col[2])
    poss.append([nom, lat, lon])
    Ls = len(poss)

# Ordenem les ciutats de sortida per la distancia total
aux = []
ordre = []
fca = open(fitxer, "w+")

for i in range(0, Ls):
    seg = min_path[i]

Annex

for k in range(0,Ls):
    if poss[k][0]==seg:
        ordre.append(poss[k])
        ciutat=poss[k][0]+','+poss[k][1]+','+poss[k][2]
        fca.write(ciutat)

fca.close()

gp=distgrup("A")
gp=distgrup("B")
gp=distgrup("C")
gp=distgrup("D")

A.4 Program kmeanssubgrups

import random
from math import *

def disesf(v1,v2):
    a=90-v1[0]
    b=90-v2[0]
    P=v1[1]-v2[1]
    ad=radians(a)
    bd=radians(b)
    Pd=radians(P)

    cosdis=cos(ad)*cos(bd)+sin(ad)*sin(bd)*cos(Pd)
    dis=degrees(acos(cosdis))
    distancia=dis*4000/36
    return distancia

def subgrup(grup):
    if grup=="A":
        fitxer="ciutatsA.txt"
    if grup=="B":
        fitxer="ciutatsB.txt"
    if grup=="C":
        fitxer="ciutatsC.txt"
fitxer="ciutatsC.txt"
if grup="D":
    fitxer="ciutatsD.txt"

fc = open(fitxer,"r")
lines=[line for line in fc]
fc.close()
pos=[]
for line in lines:
    col=line.split(',
    x=float(col[1])
y=float(col[2])
pos.append([col[0],x,y])

#Busquem el mxim i el mnim de cada coordenada
minx=1000; miny=1000
maxx=0; maxy=0
for i in range(len(pos)):
    if pos[i][1]<minx:
        minx=(pos[i][1])
    if pos[i][1]>maxx:
        maxx=(pos[i][1])
    if pos[i][2]<miny:
        miny=(pos[i][2])
    if pos[i][2]>maxy:
        maxy=(pos[i][2])
ranges=[minx,maxx,miny,maxy]
print(ranges)

#Creem 2 centroids aleatoris dintre del rang
clusters=[]
k=2
for i in range(k):
    # clx=ranges[0]+random.random()*(ranges[1]-ranges[0])
    # cly=ranges[2]+random.random()*(ranges[3]-ranges[2])
    #if i==0 or i==2:
    #clx=ranges[0]+(ranges[1]-ranges[0])*i+1/4
    #cly=ranges[2]+(ranges[3]-ranges[2])*i+1/4
#if i==1 or i==3:
  
  #clx=ranges[0]+(ranges[1]-ranges[0])*i/4
  #cly=ranges[3]+(ranges[2]-ranges[3])*i/4
  clx=ranges[0]+(ranges[1]-ranges[0])*(i+1)/3
  cly=ranges[2]+(ranges[3]-ranges[2])*(i+1)/3

  clusters.append([clx,cly])

print (clusters)

lastmatches=None
for t in range(5):
  print('Iteracio %d' % t)
  bestmatches=[[0 for i in range(k)] for j in range(len(pos))]
  for j in range(len(pos)):
    row=pos[j]
    bestmatch=0
    v2=[row[1],row[2]]
    for i in range(k):
      v1=clusters[i]
      d=disesf(v1,v2)
      if d<disesf(clusters[bestmatch],v2):
        bestmatch=i
      bestmatches[bestmatch].append(j)

  # If the results are the same as last time, this is complete
  if bestmatches==lastmatches:
    break
  lastmatches=bestmatches

  # Move the centroids to the average of their members
  ca=[]
  for i in range(k):
    avgs=[0.0]*2
    mx=0
    my=0
    print(bestmatches[i])
if len(bestmatches[i])>0:
    for rowid in bestmatches[i]:
        print(rowid)
        print(pos[rowid])
        ca.append(pos[rowid])

        avgs[0]+=int(pos[rowid][1])
        avgs[1]+=int(pos[rowid][2])
        mx=avgs[0]/len(bestmatches[i])
        my=avgs[1]/len(bestmatches[i])

        clusters[i][0]=mx
        clusters[i][1]=my
        print(clusters[i])

if grup=="A":
    fitxer1="ciutatsA1.txt"
    fitxer2="ciutatsA2.txt"
if grup=="B":
    fitxer1="ciutatsB1.txt"
    fitxer2="ciutatsB2.txt"
if grup=="C":
    fitxer1="ciutatsC1.txt"
    fitxer2="ciutatsC2.txt"
if grup=="D":
    fitxer1="ciutatsD1.txt"
    fitxer2="ciutatsD2.txt"

fc1 = open(fitxer1,"w+")
fc2 = open(fitxer2,"w+")
L0=len(bestmatches[0])
L1=len(bestmatches[1])
for q in range(L0):
    ciutat=ca[q][0]+',',+str(ca[q][1])+',',+str(ca[q][2])
    fc1.write(ciutat)
    fc1.write("\n")

for q in range(L0,L0+L1):
ciutat=ca[q][0]+','+str(ca[q][1])+','+str(ca[q][2])
fc2.write(ciutat)
fc2.write("\n")

sg=subgrup("A")
sg=subgrup("B")
sg=subgrup("C")
sg=subgrup("D")

A.5 Program modifikasubgrups

from math import *

def disesf(v1,v2):
a=90-v1[0]
b=90-v2[0]
P=v1[1]-v2[1]
ad=radians(a)
bd=radians(b)
Pd=radians(P)

cosdis=cos(ad)*cos(bd)+sin(ad)*sin(bd)*cos(Pd)
dis=degrees(acos(cosdis))
distancia=dis*1000/9
return distancia

def modify(fitxer1,fitxer2):
f1 = open(fitxer1,"r")
lines=[line for line in f1]
f1.close()
pos1=[]
for line in lines:
    col=line.split(',',')
x=float(col[1])
y=float(col[2])
pos1.append([col[0],x,y])
L1=len(pos1)
if L1==3 or L1>3:
    return
if L1<3:
    avgs=[0.0]*2
    mx=0
    my=0
    for i in range(L1):
        avgs[0]+=int(pos1[i][1])
        avgs[1]+=int(pos1[i][2])
    mx=avgs[0]/L1
    my=avgs[1]/L1

f2 = open(fitxer2,"r")
lines=[line for line in f2]
f2.close()
pos2=[]
for line in lines:
    col=line.split(’,’,)
    x=float(col[1])
    y=float(col[2])
    pos2.append([col[0],x,y])
L2=len(pos2)

if L2>3:
    dmin=5000
    cmin=[]

    for j in range(L2):
        v1=(pos2[j][1],pos2[j][2])
        v2=(mx,my)
        d=disesf(v1,v2)
        if d<dmin:
            dmin=d
            cmin=[]
            cmin.append([pos2[j][0],pos2[j][1],pos2[j][2]])
    c=cmin[0][0]
    cx=cmin[0][1]
cy=cmin[0][2]

pos1.append([c,cx,ry])

for h in range(len(pos2)):
    c1=pos2[h][0]
    c2=cmin[0][0]
    if c1==c2:
        del pos2[h][0:3]

f1 = open(fitxer1,"w+")
for q in range(len(pos1)):
    ciutat=pos1[q][0]+','+str(pos1[q][1])+','+str(pos1[q][2])
    f1.write(ciutat)
    f1.write("\n")
f1.close()

f2 = open(fitxer2,"w+")
for q in range(len(pos2)):
    if pos2[q]!=[]:
        ciutat=pos2[q][0]+','+str(pos2[q][1])+','+str(pos2[q][2])
        f2.write(ciutat)
        f2.write("\n")
f2.close()

mdf=modify("ciutatsA1.txt","ciutatsA2.txt")
mdf=modify("ciutatsA2.txt","ciutatsA1.txt")
mdf=modify("ciutatsB1.txt","ciutatsB2.txt")
mdf=modify("ciutatsB2.txt","ciutatsB1.txt")
mdf=modify("ciutatsC1.txt","ciutatsC2.txt")
mdf=modify("ciutatsC2.txt","ciutatsC1.txt")
mdf=modify("ciutatsD1.txt","ciutatsD2.txt")
mdf=modify("ciutatsD2.txt","ciutatsD1.txt")

A.6 Program calendari

from math import *
def calgrup(grup,sentit):
    if grup=='A':
        fitxer="ciutatsA.txt"
    if grup=='B':
        fitxer="ciutatsB.txt"
    if grup=='C':
        fitxer="ciutatsC.txt"
    if grup=='D':
        fitxer="ciutatsD.txt"
    fc = open(fitxer,"r")
    lines=[line for line in fc]
    fc.close()
    pos=
    for line in lines:
        col=line.split(',',)
        x=float(col[1])
        y=float(col[2])
        pos.append([col[0],x,y])
    L=len(pos)

    # Calendari de les 5 jornades:
    llista=[
        if sentit=="D":
            if mf=="N":
                fcal = open("BlocA.txt","w+")
            if mf=="S":
                fcal = open("BlocA.txt","a+")
        if sentit=="I":
            if mf=="N":
                fcal = open("BlocB.txt","w+")
            if mf=="S":
                fcal = open("BlocB.txt","a+")
        for p in range(L):
            ciutat=pos[p][0]
            llista.append(ciutat)
        for q in range(5):
llistaa=[]
print('Jornada ',q+1)
if sentit=="D":
    if q==0:
        print(llista[5],'-',llista[0])
        print(llista[4],'-',llista[1])
        print(llista[2],'-',llista[3])
        partit1=llista[5]+' - '+llista[0]
        fcal.write(partit1)
        fcal.write("\n")
        fcal.write(partit2)
        fcal.write("\n")
        fcal.write(partit3)
        fcal.write("\n")
    if q==1:
        print(llista[0],'-',llista[4])
        print(llista[3],'-',llista[5])
        print(llista[1],'-',llista[2])
        partit1=llista[0]+' - '+llista[4]
        fcal.write(partit1)
        fcal.write("\n")
        fcal.write(partit2)
        fcal.write("\n")
        fcal.write(partit3)
        fcal.write("\n")
    if q==2:
        print(llista[3],'-',llista[0])
        print(llista[4],'-',llista[2])
        print(llista[5],'-',llista[1])
        partit1=llista[3]+' - '+llista[0]
        fcal.write(partit1)
        fcal.write("\n")
        fcal.write(partit2)
fcal.writeln("n")
fcal.writeln(partit3)
fcal.writeln("n")

if q==3:
    print(llista[2], ' - ', llista[0])
    print(llista[1], ' - ', llista[3])
    print(llista[4], ' - ', llista[5])
    partit1=llista[2] + ' + llista[0]
fcal.writeln(partit1)
fcal.writeln("n")
fcal.writeln(partit2)
fcal.writeln("n")
fcal.writeln(partit3)
fcal.writeln("n")

if q==4:
    print(llista[0], ' - ', llista[1])
    print(llista[5], ' - ', llista[2])
    print(llista[3], ' - ', llista[4])
    partit1=llista[0] + ' + llista[1]
fcal.writeln(partit1)
fcal.writeln("n")
fcal.writeln(partit2)
fcal.writeln("n")
fcal.writeln(partit3)
fcal.writeln("n")

if sentit=="I":
    if q==0:
        print(llista[0], ' - ', llista[5])
        print(llista[1], ' - ', llista[4])
        print(llista[3], ' - ', llista[2])
        partit1=llista[0] + ' + llista[5]
fcal.writeln(partit1)
```python
fcal.write("\n")
fcal.write(partit2)
fcal.write("\n")
fcal.write(partit3)
fcal.write("\n")
if q==1:
    print(llista[4], ' - ', llista[0])
    print(llista[5], ' - ', llista[3])
    print(llista[2], ' - ', llista[1])
    partit1=llista[4]+' - '+llista[0]
fcal.write(partit1)
fcal.write("\n")
fcal.write(partit2)
fcal.write("\n")
fcal.write(partit3)
fcal.write("\n")
if q==2:
    print(llista[0], ' - ', llista[3])
    print(llista[2], ' - ', llista[4])
    print(llista[1], ' - ', llista[5])
    partit1=llista[0]+' - '+llista[3]
fcal.write(partit1)
fcal.write("\n")
fcal.write(partit2)
fcal.write("\n")
fcal.write(partit3)
fcal.write("\n")
if q==3:
    print(llista[0], ' - ', llista[2])
    print(llista[3], ' - ', llista[1])
    print(llista[5], ' - ', llista[4])
    partit1=llista[0]+' - '+llista[2]
```
A.6 Program calendari

```python
fcal.write(partit1)
fcal.write("\n")
partit2=llista[3]+'
'+llista[1]
fcal.write(partit2)
fcal.write("\n")
partit3=llista[5]+'
'+llista[4]
fcal.write(partit3)
fcal.write("\n")

if q==4:
    print(llista[1],', - ',llista[0])
    print(llista[2],', - ',llista[5])
    print(llista[4],', - ',llista[3])
    partit1=llista[1]+'
'+llista[0]
fcal.write(partit1)
fcal.write("\n")
    partit2=llista[2]+'
'+llista[5]
fcal.write(partit2)
fcal.write("\n")
    partit3=llista[4]+'
'+llista[3]
fcal.write(partit3)
fcal.write("\n")

fcal.close()

mf="N"
cg=calgrup("A","D")
cg=calgrup("A","I")

mf="S"

cg=calgrup("B","D")
cg=calgrup("C","D")
cg=calgrup("D","D")
cg=calgrup("B","I")
cg=calgrup("C","I")
cg=calgrup("D","I")
```
A.7 Program desplaçament

from math import *

def disesf(v1,v2):
    a=90-v1[0]
    b=90-v2[0]
    P=v1[1]-v2[1]
    ad=radians(a)
    bd=radians(b)
    Pd=radians(P)

    cosdis=cos(ad)*cos(bd)+sin(ad)*sin(bd)*cos(Pd)
    dis=degrees(acos(cosdis))
    distancia=dis*1000/9
    return distancia

def calgrup(gcasa,gfora):

    if gcasa==’A’:
        fitxer="ciutatsA.txt"
        if gfora==’C’:
            fitxere="BlocD.txt"
        if gfora==’D’:
            fitxere="BlocF.txt"
    if gcasa==’B’:
        fitxer="ciutatsB.txt"
        if gfora==’C’:
            fitxere="BlocF.txt"
        if gfora==’D’:
            fitxere="BlocD.txt"
    if gcasa==’C’:
        fitxer="ciutatsC.txt"
        if gfora==’A’:
            fitxere="BlocC.txt"
        if gfora==’B’:
            fitxere="BlocE.txt"
    if gcasa==’D’:
```python
fitxer="ciutatsD.txt"
if gfora=='A':
    fitxere="BlocE.txt"
if gfora=='B':
    fitxere="BlocC.txt"
if gfora=='A':
    fitxers="ciutatsA.txt"
if gfora=='B':
    fitxers="ciutatsB.txt"
if gfora=='C':
    fitxers="ciutatsC.txt"
if gfora=='D':
    fitxers="ciutatsD.txt"

fcs = open(fitxers,"r")
liness=[line for line in fcs]
fcs.close()
poss=[]
for line in liness:
    col=line.split(',')
    d1x=float(col[1])
    d1y=float(col[2])
    poss.append([col[0],d1x,d1y])
Ls=len(poss)

fc = open(fitxer,"r")
lines=[line for line in fc]
f.close()
pos=[]
for line in lines:
    col=line.split(',')
    x=float(col[1])
    y=float(col[2])
    pos.append([col[0],x,y])
L=len(pos)
matdis=[]
for j in range(Ls):
    for i in range(L):
```
v2=(poss[j][1], poss[j][2])
v1=(pos[i][1], pos[i][2])
d=disesf(v1, v2)
matdis.append([pos[i][0], poss[j][0], d])
# print(matdis)

suma=0
sumadespa=0
sumadespb=0
sumadespc=0
sumadespd=0
sumadespe=0
sumadespf=0
matsum=[]
matdesp=[]

for j in range(Ls):
    sumtot=0
    for i in range(L):
        if i==L-1:
            # print(poss[j][0], ' - ', pos[i][0])
            # print(pos[0][0], ' - ', poss[j][0])
            suma=matdis[j*6+i][2]
            suma+=matdis[j*6+1][2]
            sumadespf+=suma
            tornada5=pos[0][0]
            anada5=pos[i][0]
            if j==5:
                matdesp.append([anada5, tornada5, sumadespf])
            else:
                # print(poss[j][0], ' - ', pos[i][0])
                # print(pos[i+1][0], ' - ', poss[j][0])
                suma=matdis[j*6+i][2]
                suma+=matdis[j*6+i+1][2]
                if i==0:
                    sumadespa+=suma
                    tornada0=pos[i+1][0]
                    anada0=pos[i][0]
if j==5:
    matdesp.append([anada0, tornada0, sumadespa])
if i==1:
    sumadespb+=suma
    tornada1=pos[i+1][0]
    anada1=pos[i][0]
    if j==5:
        matdesp.append([anada1, tornada1, sumadespb])
if i==2:
    sumadespc+=suma
    tornada2=pos[i+1][0]
    anada2=pos[i][0]
    if j==5:
        matdesp.append([anada2, tornada2, sumadespc])
if i==3:
    sumadespd+=suma
    tornada3=pos[i+1][0]
    anada3=pos[i][0]
    if j==5:
        matdesp.append([anada3, tornada3, sumadespd])
if i==4:
    sumadespe+=suma
    tornada4=pos[i+1][0]
    anada4=pos[i][0]
    if j==5:
        matdesp.append([anada4, tornada4, sumadespe])
#
    print(suma)
    sumtot+=suma
#
    print(sumtot)
    matsum.append([sumtot, poss[j][0]])

# Ordenem les ciutats de sortida per la distancia total
aux=[]
for k in range(1,Ls):
    for l in range(0,Ls-k):
        if matsum[l][0]>matsum[l+1][0]:
            aux=matsum[l]
            matsum[l]=matsum[l+1]
matsum[l+1]=aux

# Ordenem els desplaaments per la distància total
auxd=[]
for kd in range(1,Ls):
    for ld in range(0,Ls-kd):
        if matdesp[ld][2]>matdesp[ld+1][2]:
            auxd=matdesp[ld]
            matdesp[ld]=matdesp[ld+1]
            matdesp[ld+1]=auxd

print(matsum)
print(matdesp)

# Assignem el desplaament més llarg a la ciutat a menys distància
n=5
viatge=[]
for m in range(n+1):
    sort=matsum[m][1]
    arrib=matdesp[n-m][0]
    torn=matdesp[n-m][1]
    viatge.append([sort,arrib,torn])
print(viatge)

# Calendari de les 6 jornades:
jornades=[]
for p in range(6):
    local=pos[p][0]
    for v in range(6):
        if viatge[v][1]==local:
            visitant=viatge[v][0]
            jornades.append([local,visitant])
if mf=="N":
    fcal = open(fitxere,"w+")
if mf=="S":
    fcal = open(fitxere,"a+")
for q in range (6):
print('Jornada ',q+1)
if q==0:
    for qe in range(6):
        print(jornades[qe][0],'-',jornades[qe][1])
        partit=jornades[qe][0]+' - '+jornades[qe][1]
        fcal.write(partit)
        fcal.write("\n")
else:
    for qe in range(6):
        print(jornades[qe][0],'-',jornades[(qe+q)%6][1])
        partit=jornades[qe][0]+' - '+jornades[(qe+q)%6][1]
        fcal.write(partit)
        fcal.write("\n")

fcal.close()
mf="N"
gcf=calgrup("A","C")
gcf=calgrup("C","A")
gcf=calgrup("A","D")
gcf=calgrup("C","B")
mf="S"
gcf=calgrup("B","C")
gcf=calgrup("B","D")
gcf=calgrup("D","A")
gcf=calgrup("D","B")

A.8 Program desplaçamentc Curt

from math import *

def diseuf(v1,v2):
a=90-v1[0]
b=90-v2[0]
P=v1[1]-v2[1]
ad=radians(a)
bd=radians(b)
Annex

Pd=radians(P)

\[
\cos \text{dis} = \cos(\text{ad}) \cdot \cos(\text{bd}) + \sin(\text{ad}) \cdot \sin(\text{bd}) \cdot \cos(Pd)
\]

dis=degrees(acos(cosdis))

distancia=dis*1000/9

return distancia

def calgrup(gcasa,gfora):
    if gcasa=='A1':
        fitxer="ciutatsA1.txt"
    if gfora=='B1':
        fitxere="BlocG.txt"
    if gfora=='B2':
        fitxere="BlocJ.txt"
    if gcasa=='A2':
        fitxer="ciutatsA2.txt"
    if gfora=='B1':
        fitxere="BlocJ.txt"
    if gfora=='B2':
        fitxere="BlocG.txt"
    if gcasa=='B1':
        fitxer="ciutatsB1.txt"
    if gfora=='A1':
        fitxere="BlocI.txt"
    if gfora=='A2':
        fitxere="BlocH.txt"
    if gcasa=='B2':
        fitxer="ciutatsB2.txt"
    if gfora=='A1':
        fitxere="BlocH.txt"
    if gfora=='A2':
        fitxere="BlocI.txt"
    if gcasa=='C1':
        fitxer="ciutatsC1.txt"
    if gfora=='D1':
        fitxere="BlocG.txt"
    if gfora=='D2':
        fitxere="BlocJ.txt"
if gcasa==’C2’:
    fitxer="ciutatsC2.txt"
if gfora==’D1’:
    fitxere="BlocJ.txt"
if gfora==’D2’:
    fitxere="BlocG.txt"
if gcasa==’D1’:
    fitxer="ciutatsD1.txt"
if gfora==’C1’:
    fitxere="BlocI.txt"
if gfora==’C2’:
    fitxere="BlocH.txt"
if gcasa==’D2’:
    fitxer="ciutatsD2.txt"
if gfora==’C1’:
    fitxere="BlocH.txt"
if gfora==’C2’:
    fitxere="BlocI.txt"

if gfora==’A1’:
    fitxers="ciutatsA1.txt"
if gfora==’A2’:
    fitxers="ciutatsA2.txt"
if gfora==’B1’:
    fitxers="ciutatsB1.txt"
if gfora==’B2’:
    fitxers="ciutatsB2.txt"
if gfora==’C1’:
    fitxers="ciutatsC1.txt"
if gfora==’C2’:
    fitxers="ciutatsC2.txt"
if gfora==’D1’:
    fitxers="ciutatsD1.txt"
if gfora==’D2’:
    fitxers="ciutatsD2.txt"

fcs = open(fitxer,"r")
liness=[line for line in fcs]
fcs.close()
poss=[]
for line in liness:
    col=line.split(',')
    d1x=float(col[1])
    d1y=float(col[2])
    poss.append([col[0],d1x,d1y])
Ls=len(poss)

fc = open(fitxers,"r")
lines=[line for line in fc]
fcs.close()
pos=[]
for line in lines:
    col=line.split(',')
    x=float(col[1])
    y=float(col[2])
    pos.append([col[0],x,y])
L=len(pos)
matdis=[]
for j in range(Ls):
    for i in range(L):
        v2=(poss[j][1],poss[j][2])
        v1=(pos[i][1],pos[i][2])
        d=distev(v1,v2)
        matdis.append([pos[i][0],poss[j][0],d])
#print(matdis)

suma=0
sumadespa=0
sumadespb=0
sumadespf=0
matsum=[]
matdesp=[]

for j in range(Ls):
    sumtot=0
    for i in range(L):
if i==L-1:
    print(poss[j][0],’ -‘,pos[i][0])
    print(pos[0][0],’ -‘,poss[j][0])
suma=matdis[j*3+i][2]
suma+=matdis[j*3][2]
sumadespf+=suma
tornada5=pos[0][0]
anada5=pos[i][0]
if j==2:
    matdesp.append([anada5,tornada5,sumadespf])
else:
    print(poss[j][0],’ -‘,pos[i][0])
    print(pos[i+1][0],’ -‘,poss[j][0])
suma=matdis[j*3+i][2]
suma+=matdis[j*3+i+1][2]
if i==0:
    sumadespa+=suma
tornada0=pos[i+1][0]
anada0=pos[i][0]
if j==2:
    matdesp.append([anada0,tornada0,sumadespa])
if i==1:
    sumadespb+=suma
tornada1=pos[i+1][0]
anada1=pos[i][0]
if j==2:
    matdesp.append([anada1,tornada1,sumadespb])

# print(suma)
sumtot+=suma
# print(sumtot)
matsum.append([sumtot,poss[j][0]])

# Ordenem les ciutats de sortida per la distància total
aux=[]
for k in range(1,Ls):
    for l in range(0,Ls-k):
        if matsum[l][0]>matsum[l+1][0]:
            aux=matsum[l]
matsum[1]=matsum[1+1]
matsum[1+1]=aux

# Ordenem els desplaaments per la distància total
auxd=[]
for kd in range(1,Ls):
    for ld in range(0,Ls-kd):
        if matdesp[ld][2]>matdesp[ld+1][2]:
            auxd=matdesp[ld]
            matdesp[ld]=matdesp[ld+1]
            matdesp[ld+1]=auxd

print(matsum)
print(matdesp)

# Assignem el desplaament més llarg a la ciutat a menor distància
n=2
viatge=[]
for m in range(n+1):
    sort=matsum[m][1]
    arrib=matdesp[n-m][0]
    torn=matdesp[n-m][1]
    viatge.append([sort,arrib,torn])
print(viatge)

# Calendari de les 3 jornades:
jornades=[]
for p in range(3):
    local=pos[p][0]
    for v in range(3):
        if viatge[v][1]==local:
            visitant=viatge[v][0]
            jornades.append([local,visitant])

if mf=="N":
    fcal = open(fitxere,"w+")
if mf=="S":
    fcal = open(fitxere,"a+")
for q in range (3):
print('Jornada ',q+1)
if q==0:
    for qe in range(3):
        print(jornades[qe][0],'-',jornades[qe][1])
        partit=jornades[qe][0]+' - '+jornades[qe][1]
        fcal.write(partit)
        fcal.write("\n")
else:
    for qe in range(3):
        print(jornades[qe][0],'-',jornades[(qe+q)%3][1])
        partit=jornades[qe][0]+' - '+jornades[(qe+q)%3][1]
        fcal.write(partit)
        fcal.write("\n")

fcal.close()
mf="N"
gcf=calgrup("A1","B1")
gcf=calgrup("A1","B2")
gcf=calgrup("B1","A1")
gcf=calgrup("B1","A2")
mf="S"

gcf=calgrup("C1","D1")
gcf=calgrup("C1","D2")
gcf=calgrup("D1","C1")
gcf=calgrup("D1","C2")
gcf=calgrup("A2","B1")
gcf=calgrup("A2","B2")
gcf=calgrup("B2","A1")
gcf=calgrup("B2","A2")
gcf=calgrup("C2","D1")
gcf=calgrup("C2","D2")
gcf=calgrup("D2","C1")
gcf=calgrup("D2","C2")
A.9 Program *calendarifinal*

```python
from math import *

# Bloc G, H, I, J:
def blocghij(fitxer,j):
    f = open(fitxer,"r")
    lines=[line for line in f]
    f.close()
    calendari=[]
    for line in lines:
        col=line.split('\n')
        calendari.append(col)
    L=len(calendari)
    partit1=[]
    partit2=[]
    partit3=[]
    # Calendari de les 3 jornades dels blocs G, H, I, J:
    for p in range(L):
        if p%9==int() or (p-1)%9==int() or (p-2)%9==int():
            partit=calendari[p]
            partit1.append(partit)
        if (p-3)%9==int() or (p-4)%9==int() or (p-5)%9==int():
            partit=calendari[p]
            partit2.append(partit)
        if (p-6)%9==int() or (p-7)%9==int() or (p-8)%9==int():
            partit=calendari[p]
            partit3.append(partit)
    Lp=len(partit1)
    print("\n")
    print('Jornada ',j+1)
    for i in range(Lp):
        print(partit1[i][0])
    print("\n")
    print('Jornada ',j+2)
    for i in range(Lp):
        print(partit2[i][0])
```

A.9 Program calendarifinal

```python
print("\n")
print('Jornada ',j+3)
for i in range(Lp):
    print(partit3[i][0])

# Bloc A, B:
def blocab(fitxer,j):
    f = open(fitxer,"r")
    lines=[line for line in f]
    f.close()
    calendari=[]
    for line in lines:
        col=line.split('\n')
        calendari.append(col)
    L=len(calendari)

    partit1=[]
    partit2=[]
    partit3=[]
    partit4=[]
    partit5=[]

    # Calendari de les 5 jornades dels blocs A, B:
    for p in range(L):
        if p%15==int() or (p-1)%15==int() or (p-2)%15==int():
            partit=calendari[p]
            partit1.append(partit)
        if (p-3)%15==int() or (p-4)%15==int() or (p-5)%15==int():
            partit=calendari[p]
            partit2.append(partit)
        if (p-6)%15==int() or (p-7)%15==int() or (p-8)%15==int():
            partit=calendari[p]
            partit3.append(partit)
        if (p-9)%15==int() or (p-10)%15==int() or (p-11)%15==int():
            partit=calendari[p]
            partit4.append(partit)
        if (p-12)%15==int() or (p-13)%15==int() or (p-14)%15==int():
            partit=calendari[p]
            partit5.append(partit)
```
Lp=len(partit1)
print("\n")
print(‘Jornada ’,j+1)
for i in range(Lp):
    print(partit1[i][0])
print("\n")
print(‘Jornada ’,j+2)
for i in range(Lp):
    print(partit2[i][0])
print("\n")
print(‘Jornada ’,j+3)
for i in range(Lp):
    print(partit3[i][0])
print("\n")
print(‘Jornada ’,j+4)
for i in range(Lp):
    print(partit4[i][0])
print("\n")
print(‘Jornada ’,j+5)
for i in range(Lp):
    print(partit5[i][0])

# Calendari de les 6 jornades dels blocs C, D, E, F:
def bloccdef(fitxer,j):
f = open(fitxer,"r")
lines=[line for line in f]
f.close()
calendari=[]
for line in lines:
    col=line.split(‘\n’)  
    calendari.append(col)
L=len(calendari)
partit1=[]
partit2=[]
partit3=[]
partit4=[]
partit5=[]
partit6=[]
for p in range(L):
    if p < 6 or (p > 35 and p < 42):
        partit = calendari[p]
        partit1.append(partit)
    if (p > 5 and p < 12) or (p > 41 and p < 48):
        partit = calendari[p]
        partit2.append(partit)
    if (p > 11 and p < 18) or (p > 47 and p < 54):
        partit = calendari[p]
        partit3.append(partit)
    if (p > 17 and p < 24) or (p > 53 and p < 60):
        partit = calendari[p]
        partit4.append(partit)
    if (p > 23 and p < 30) or (p > 59 and p < 66):
        partit = calendari[p]
        partit5.append(partit)
    if (p > 29 and p < 36) or p > 65:
        partit = calendari[p]
        partit6.append(partit)

Lp = len(partit1)
print("\n")
print("Jornada ", j+1)
for i in range(Lp):
    print(partit1[i][0])
print("\n")
print("Jornada ", j+2)
for i in range(Lp):
    print(partit2[i][0])
print("\n")
print("Jornada ", j+3)
for i in range(Lp):
    print(partit3[i][0])
print("\n")
print("Jornada ", j+4)
for i in range(Lp):
    print(partit4[i][0])
print("\n")
print('Jornada ',j+5)

for i in range(Lp):
    print(partit5[i][0])
print("\n")
print('Jornada ',j+6)
for i in range(Lp):
    print(partit6[i][0])

cal=blocghij("BlocG.txt",0)
cal=blocghij("BlocH.txt",3)
cal=blocab("BlocA.txt",6)
cal=bloccdef("BlocC.txt",11)
cal=bloccdef("BlocF.txt",17)
cal=blocab("BlocB.txt",23)
cal=blocghij("BlocI.txt",28)
cal=blocghij("BlocJ.txt",31)
cal=blocghij("BlocG.txt",34)
cal=blocghij("BlocH.txt",37)
cal=blocab("BlocA.txt",40)
cal=bloccdef("BlocD.txt",45)
cal=bloccdef("BlocE.txt",51)
cal=blocab("BlocB.txt",57)
cal=blocghij("BlocI.txt",62)
cal=blocghij("BlocJ.txt",65)