Universitat de Lleida

Algorithms

Jordi Planes

Escola Politècnica Superior
Universitat de Lleida

2016

Opinion Poll

The Art of War

What is the best way to attack a bigger army?


孫子
兵法

Syllabus

What's been done

- Formal specification
- Cost
- Transformation recursion $\rightarrow$ iteration
- Divide and conquer

Syllabus

What we'll do today

- Formal specification
- Cost
- Transformation recursion $\rightarrow$ iteration
- Divide and conquer

Preliminaries

Sequences

Example

Demonstrate $T(n) = T(n-1) + 1$

1. Base case: $T(1) = 1$
2. Recursive case:
   - Assume $T(n-1)$ is correct: $T(n-1) = n - 1$

Sequences

Example

Demonstrate $T(n) = T(n-1) + 1$

1. Base case: $T(1) = 1$
2. Recursive case:
   - Assume $T(n-1)$ is correct: $T(n-1) = n - 1$
   - $T(n) = (n-1) + 1 \Rightarrow T(n) = n$

Sequences

Exercises

1. $T(n) = T(n-1) + n$

Sequences

Let us solve the recurrence.

$$C_N = C_{N/2} + 1$$

Assume $N = 2^n$ (note $n = \log N$). Let us expand the recurrence:

$$\begin{aligned}
C_N &= C_{N/2} + 1 \\
C_{2^n} &= C_{2^{n-1}} + 1 \\
&= C_{2^{n-2}} + 2 \\
&= C_{2^{n-3}} + 3 \\
&\vdots \\
&= C_{2^0} + n \\
&= 1 + n.
\end{aligned}$$

C_N is about $\log N$.

Sequences

Solve the following recurrences for $N \geq 2$ and $C_1 = 1$:

$$C_N = C_{N/2} + N \tag{4}$$
$$C_N = 2C_{N/2} + 1 \tag{5}$$
$$C_N = 2C_{N/2} + N \tag{6}$$

Sequences

Master theorem

$$T(n) = aT(n/b) + f(n^c), a \geq 1, b > 1$$

where:

n	size of the problem
a	number of subproblems
n/b	size of each subproblem
$f(n^c)$	cost of the work out of recursive calls

$$T(n) = \begin{cases} n^{\log_b a} & ifc < \log_b a \\ n^c log^{k+1} n & ifc = \log_b a \\ f(n) & ifc > \log_b a \end{cases}$$

About computational cost

- We have assumed any parameter as cost variable
- Formally: The function of number of steps by number of input bits.

$$steps = f(bits)$$

- If input parameter x and n number of input bits, and function is in $O(x)$, then the cost is:

$$O(x) \quad = [n = \log_2 x \Rightarrow 2^n = 2^{\log_2 x} \Rightarrow 2^n = x] = \quad O(2^n)$$

Divide and conquer

Multiplication

Let's take a look how we do integer multiplication

- Several additions $O(2^n)$
  $10 \times 5 = 10 + 10 + 10 + 10 + 10$

Multiplication

Let's take a look how we do integer multiplication

- Several additions $O(2^n)$
  $10 \times 5 = 10 + 10 + 10 + 10 + 10$
- Decimal multiplication $O(n^2)$
  $57 \times 32 = (5 * 10 + 7) \times (3 * 10 + 2) =$
  $3 \times 5 * 10^2 + (3 \times 7 + 2 \times 5) * 10 + 2 \times 7 = 1500 + 310 + 14 = 1824$
  4 multiplications (roughly speaking:
  $O(n^2) \equiv num.multp. \times costmultp. = digits \times costmultp.$)

Multiplication

Let's take a look how we do integer multiplication

- Several additions $O(2^n)$
  $10 \times 5 = 10 + 10 + 10 + 10 + 10$
- Decimal multiplication $O(n^2)$
  $57 \times 32 = (5 * 10 + 7) \times (3 * 10 + 2) =$
  $3 \times 5 * 10^2 + (3 \times 7 + 2 \times 5) * 10 + 2 \times 7 = 1500 + 310 + 14 = 1824$
  4 multiplications (roughly speaking:
  $O(n^2) \equiv num.multp. \times costmultp. = digits \times costmultp.$)
- Binary multiplication $O(n^2)$
  - binary shift and binary addition (Ex. 10×5)
    $110 \times 11 = 110 + 1100 = 10010$

Multiplication

Let's take a look how we do integer multiplication

- Several additions $O(2^n)$
  $10 \times 5 = 10 + 10 + 10 + 10 + 10$
- Decimal multiplication $O(n^2)$
  $57 \times 32 = (5 * 10 + 7) \times (3 * 10 + 2) =$
  $3 \times 5 * 10^2 + (3 \times 7 + 2 \times 5) * 10 + 2 \times 7 = 1500 + 310 + 14 = 1824$
  4 multiplications (roughly speaking:
  $O(n^2) \equiv num.multp. \times costmultp. = digits \times costmultp.$)
- Binary multiplication $O(n^2)$
  - binary shift and binary addition (Ex. 10×5)
    $110 \times 11 = 110 + 1100 = 10010$
- Al-Khwarizmi (780–850) method $O(n^2)$
  $10 \times 5; 5 \times 10; 2 \times 20; 1 \times 40 = 10 + 40$
  - divide and multiply by 2, add odds

Multiplication

```
function product( x, y ) is
    x = 0 → return 0
    x > 0 →
        p ← product( x−1, y )
        return  y + p
```

Multiplication

```
function product( x, y ) is
    x = 0 → return 0
    x > 0 →
        p ← product( x−1, y )
        return  y + p
```

Computational cost

$T(x) = T(x-1) + 1 \Rightarrow T(x) = O(x)$ $T(n) = O(2^n)$

Multiplication

Let us multiply 11 by 13 using Al-Khwarizmi method.

11	13
5	26
2	52
1	104

Multiplication

Let us multiply 11 by 13 using Al-Khwarizmi method.

11	**13**
5	**26**
2	52
1	**104**

Multiplication

Let us multiply 11 by 13 using Al-Khwarizmi method.

11	**13**
5	**26**
2	52
1	**104**
	143

Observe that only the addition and the 2 tables are needed.

Divide and conquer

Multiplication

$$x \cdot y = \begin{cases} 2(x \cdot (y/2)) & \text{if x is even} \\ x + 2(x \cdot (y/2)) & \text{if x is odd} \end{cases}$$

Multiplication

```
function product( x, y ) is
    x = 0 or y = 0 → return 0
    y > 0 →
        p ← product( 2*x, y/2 )
        return (x * (y%2)) + p
```

Multiplication

```
function product( x, y ) is
    x = 0 or y = 0 → return 0
    y > 0 →
        p ← product( 2∗x, y/2 )
        return (x ∗ (y%2)) + p
```

Computational cost

$T(y) = T(y/2) + 1 \Rightarrow T(y) = O(\log_2 y)$
$T(n) = O(n^2)[T(n) = T(n-1) + O(n)]$

Exercises

1. Power by multiplications
2. Division by substractions
3. Natural square root
4. Combinatorial number $\binom{m}{n}$

Divide and conquer

The divide and conquer strategy solves a problem by:

1. Breaking it into subproblems
2. Recursively solving these subproblems
3. Appropriately combining their answers.

Divide and conquer

Multiplication

Gauss' (1777–1855) method $O(n^{1.59})$

- complex num. multiplication:
  $(a + bi)(c + di) = ac - bd + (bc + ad)i$, and
  $bc + ad = (a + b)(c + d) - ac - bd$
- Given 4 "digits", instead of 4 multiplications, there are 3.

Divide and conquer

Multiplication

Divide and Conquer

- Supose x and y are two n-bit integers
- Split each of them into their left and right halves

$$x = 2^{n/2} x_L + x_R$$

$$y = 2^{n/2} y_L + y_R$$

- The product:

$$xy = (2^{n/2} x_L + x_R)(2^{n/2} y_L + y_R) = 2^n x_L y_L + 2^{n/2}(x_L y_R + x_R y_L) + x_R y_R$$

- Recurrence relation: $T(n) = 4T(n/2) + O(n)$
- Gauss' trick: $x_L y_R + x_R y_L = (x_L + x_R)(y_L + y_R) - x_L y_L - x_R y_R$
- Three multiplications: $x_L y_L, x_R y_R$ and $(x_L + x_R)(y_L + y_R)$
- Recurrence relation: $T(n) = 3T(n/2) + O(n)$

Divide and conquer
Multiplication

```
function product( x, y ) is
    n = max( size of x, size of y)
    if n = 1 → return xy

    xL, xR = leftmost n/2, rightmost n/2 bits of x
    yL, yR = leftmost n/2, rightmost n/2 bits of y

    P1 = multiply( xL, yL )
    P2 = multiply( xR, yR )
    P3 = multiply( xL + xR, yL + yR)

    return P1 2^n + (P3 − P2 − P2) 2^(n/2) + P2
```

Divide and conquer

Multiplication

```
function product( x, y ) is
    n = max( size of x, size of y)
    if n = 1 → return xy

    xL, xR = leftmost n/2, rightmost n/2 bits of x
    yL, yR = leftmost n/2, rightmost n/2 bits of y

    P1 = multiply( xL, yL )
    P2 = multiply( xR, yR )
    P3 = multiply( xL + xR, yL + yR)

    return P1 2^n + (P3 − P2 − P2) 2^(n/2) + P2
```

Computational cost

$$T(n) = 3T(n/2) + O(n) \Rightarrow T(n) \in O(n^{1.59})$$

Exercises

1. Triminoes problem
2. Pancake sorting problem
3. Skyline problem

Transformation

Transformation

Exercises

Convert to tail call and to iterative, tracing the calls:

1. Product by additions
2. Product by al-Khwarizmi
3. $f(x) = x/f(x-1)$
4. $f(x) = x - f(x-1)$

Transformation

- Conversion to tail call not allways possible
- It requires the operation to be commutative

When conversion to tail call is not possible

```
x' = x
while not base case
    x' = reduce( x' )

a = trivial( x' )
while x ≠ x'
    x' = inverse reduce( x' )
    a = compute( a, x' )

return a
```

Example: integer division

Transformation

When inverse is not possible

```
while not base case
    push to stack x
    reduce( x )

a = trivial( x )
while stack not empty
    a = compute( a, top )
    pop

return a
```

Transformation

Exercises

Convert to tail call and to iterative, tracing the calls:

1. $f(x) = x/f(x-1)$
2. $f(x) = x - f(x-1)$

Exercises

1. Power by multiplications
2. Division by substractions
3. Natural square root
4. Combinatorial number $\binom{m}{n}$

A bit of history


孫子兵法

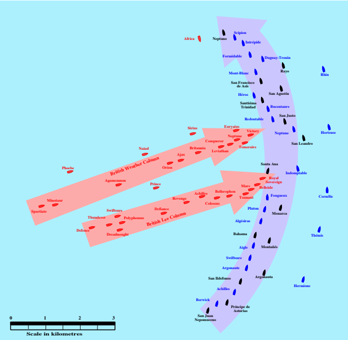
A bit of history

Very well known moto: Divide et impera. It was used by Julius Cæsar, Napoleon and Sun Tzu:

> *The art of using troops is this: When ten to the enemy's one, surround him; When five times his strength, attack him; If double his strength, divide him; . . .*
>
> *Sun Tzu, in The Art of War, Chapter 3*

This is the basis of the method: When the problem is complex, divide it; when the problem is tractable, atack it; when the problem is trivial, there is nothing to be done.

A bit of history



Known Battles:

- Marathon (490 BC) – Persians divided, Greeks attacked both separately
- Trafalgar (1805) – British (Nelson) split Franco-Spanish forces,
- Italian Campaign of Napoleon (1796–1797), . . .

Syllabus

What's been done

- Formal specification
- Cost
- Transformation recursion $\rightarrow$ iteration
- Divide and conquer
- Sorting

Syllabus

What we'll do next day

- Formal specification
- Cost
- Transformation recursion $\rightarrow$ iteration
- Divide and conquer
- Sorting