

Math 135 - Algorithms - Ch. 3.1, 3.3

Note Title

10/13/2010

Announcements

- HW due in 1 week
- No class next Monday

Last time

Basic programming Structures:

- variables

letters or words that store values.

Ex: $a_1, a_2, \dots, a_n, x, \text{number}$

- if statements

Ex: if ($x > 1$)
 $x := x + 1$

else
 $x := x - 1$

- for loops

$x := 0$
for ($i = 1$ to n)
 $x := x + i$

Suppose $n = 5$

$i =$ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~

$x =$ ~~0~~ ~~1~~ ~~3~~ ~~6~~ ~~10~~ ~~15~~

- while loops

$i := 0$
while ($i < n$)
 $i := i + 1$
 $x = x + i$

} ← repeat as long
as $i < n$

While loop example :

Compute $\lceil \log_2 n \rceil$ (given a number n).

power := 0
number := n

while (number > 1)
 number := number / 2
 power := power + 1

n = 16
← number = ~~16~~ / ~~8~~ / ~~4~~ / ~~2~~ / 1
power = ~~0~~ / ~~1~~ / ~~2~~ / ~~3~~ / 4

Pseudocode

Our goal is to show how to write a program (without actually programming).

Remember, a program is a sequence of instructions to tell a computer how to solve a problem.

Complexity

We define complexity in terms of the number of operations.

Usually, an operation is:

- add 2 things (or subtract or multiply)
- compare 2 things
- set a variable equal to something

count (worst case) # of operations

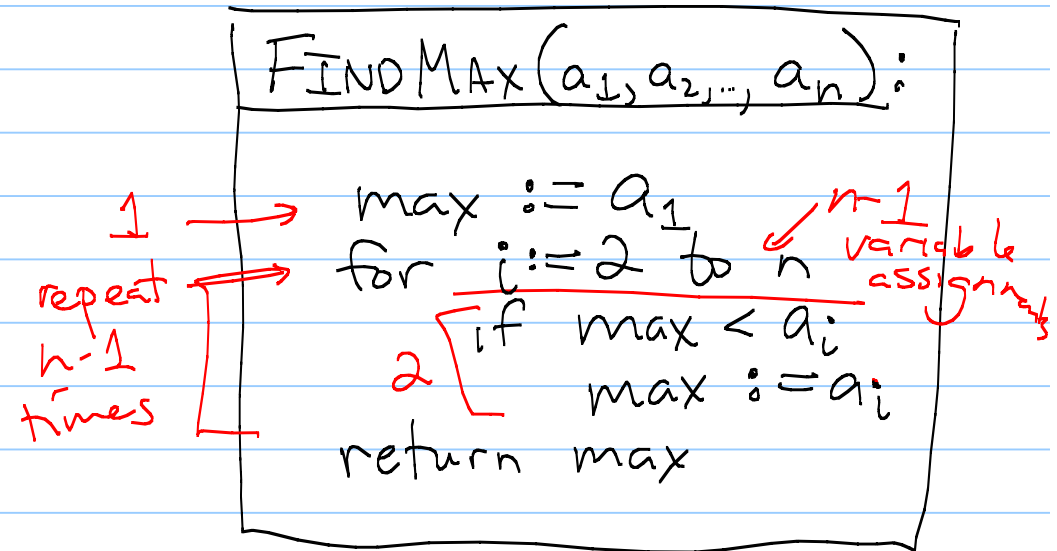
Ex: What is worst case complexity of FindMax?

operations =

$$1 + (n-1)(2) + (n-1)$$

$$= 1 + 2n - 2 + n - 1$$

$$= \boxed{3n - 2} = O(n)$$



$a_1 a_2 a_3 \dots a_n$
 x

What is worst case complexity of Linear Search?

LINEARSEARCH(x, a_1, \dots, a_n):

$i := 1$
while ($i \leq n$ and $x \neq a_i$)
 $i := i + 1$
if $i \leq n$
 location := i
else
 location := 0

1 → repeats n times
1 comparison →
2 comparisons →
1 →
1 of these happens →

$$1 + n(2+1) + 1 + 1$$

$$= \underline{3n + 3} = \mathcal{O}(n)$$

What is the complexity of Bubble Sort?

BUBBLE SORT ($a_1 \dots a_n$):

→ for $i := 1$ to $n-1$

→ for $j := 1$ to $n-i$

4 [if $(a_j > a_{j+1}) \leftarrow 1$ comparisons
Swap a_j and $a_{j+1} \leftarrow 3$ operations

} for loops
translate to
summations

$$\sum_{i=1}^{n-1} \sum_{j=1}^{n-i} 4 = \sum_{i=1}^{n-1} 4(n-i)$$

$$\underbrace{4 + 4 + 4 + \dots + 4}_{n-i} = 4(n-i)$$

$$n - (n-1)$$

$$\sum_{i=1}^{n-1} 4(n-i) = \sum_{i=1}^{n-1} 4n - \sum_{i=1}^{n-1} 4i$$

$$4n(n-1)$$

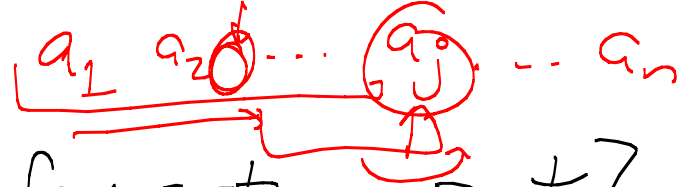
$$4 \sum_{i=1}^{n-1} i = \frac{4n(n-1)}{2}$$

$$= 4n(n-1) - 2n(n-1) = 2n(n-1)$$

$$\frac{2n^2 - 2n}{2} = n(n-1)$$

$$4 \sum_{i=1}^{n-1} (n-i) = 4((n-1) + (n-2) + (n-3) + \dots + 1) = 4 \sum_{i=1}^{n-1} i$$
$$= 4 \frac{(n-1)n}{2}$$

Method #2



What is complexity of insertion sort?

$$\sum_{j=2}^n (1+1+1+2^j)$$

$$= \sum_{j=2}^n (2^j + 3)$$

$$\vdots$$

$$= O(n^2)$$

repeats
n-1
times
total
j times

```

INSERTION SORT (a1..an):
  for j := 2 to n
    1 → i := 1
      while aj > ai
        i := i + 1
      1 → temp := aj
      for k := j to i + 1
        s → ak := ak-1
      1 → ai := temp
  
```

Why is big-O a good idea?

© The McGraw-Hill Companies, Inc. all rights reserved.

TABLE 2 The Computer Time Used by Algorithms.

Problem Size <i>n</i>	Bit Operations Used					
	$\log n$	n	$n \log n$	n^2	2^n	$n!$
10	3×10^{-9} s	10^{-8} s	3×10^{-8} s	10^{-7} s	10^{-6} s	3×10^{-3} s
10^2	7×10^{-9} s	10^{-7} s	7×10^{-7} s	10^{-5} s	4×10^{13} yr	*
10^3	10×10^{-8} s	10^{-6} s	1×10^{-5} s	10^{-3} s	*	*
10^4	13×10^{-8} s	10^{-5} s	1×10^{-4} s	10^{-1} s	*	*
10^5	17×10^{-8} s	10^{-4} s	2×10^{-3} s	10 s	*	*
10^6	2×10^{-8} s	10^{-3} s	2×10^{-2} s	17 min	*	*

$n \log n$ versus $5 n \log n$
 10×10^{-2}

So to analyze an algorithm's running time,
we often use big-O analysis.

Rather than $16n-12$, just say $O(n)$.