

CSC 165

program lower bound

week 11, lecture 1

Danny Heap

heap@cs.toronto.edu

www.cdf.toronto.edu/~heap/165/F09

Admin E3 - due either last night or tonight -
E4 - due Friday
A3 - due Friday (December 4)

counting minimum costs

Ch 6 notes

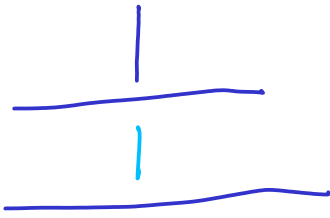
```
def IS(A) :  
    """ IS(A) sorts the elements of A in non-decreasing order """  
    1. i = 1  
    2. while i < len(A) :  
    3.     t = A[i]  
    4.     j = i  
    5.     while j > 0 and A[j-1] > t :  
    6.         A[j] = A[j-1] # shift up  
    7.         j = j-1  
    8.     A[j] = t  
    9.     i = i+1
```

I want to prove that $W_{IS} \in \Omega(n^2)$.

$$\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow \exists A \in I, \text{size}(A) = n \wedge t_{IS}(A) \geq c n^2$$

I want to choose a badly-performing (though perhaps not the worst) example

scratch

set $C =$ 

Assume $n \in \mathbb{N}$ and assume $n \geq B$ # so $n \geq 1$

Set $A = [n-1, \dots, 0]$. Then $\text{size}(A) = n$

Then all element $[0..i-1]$ will be $> t$

for each value of i and t , since these values haven't changed in

algorithm \rightarrow lines 5, 6, 7 execute i times

yielding $3i \geq 2i+1$ steps for each i .

Thus, lines 5, 6, 7 yield $\geq 2+1 + 2 \cdot 2+1 + \dots + 2(n-1)+1$

$$= \sum_{i=1}^{n-1} 2i+1 = 3+5+7+\dots+2(n-1)+1$$

Since line 1 contributes 1 step, we have at least

$$1+3+5+7+9+\dots+2(n-1)+1 = \sum_{i=1}^n 2i-1 \quad (\text{first } n \text{ odd \#s})$$

$= n^2$ // this material changed after lecture.

Thus $t_{15}(A) \geq cn^2$ // $c=1$.

loop, line 5
when $i=n$

computer mis-statements

You will have encountered unsatisfying results in python:

- ```
>>> x = 1/10.0
>>> x
>>> for i in range(9) : x += 1/10.0
>>> x
```
- ```
>>> import math
>>> math.pi
>>> math.e
```
- ```
>>> bf = 2.0
>>> for i in range(10) :
... bf *= bf
... print bf
...
```

# how numbers are represented

There are other ways to represent numbers: do arithmetic on ratios (scheme, lisp)  
but there are always costs

$$\beta = -2$$

If you fix the cost of arithmetic operations, you fix the size of numbers  
Each number is given the same space (usually bits)

Result: floating numbers are represented in scientific notation using some base  $\beta$ ,  
a fixed number of digits,  $t$ , a certain range of exponents  $e \in [e_{\min}, e_{\max}]$ ,  
and some way to store the sign.

# example

Base 10

radix  
1.5  
↑  
decimal  
point

Suppose your base  $\beta = 2$ , you allow a bit for the sign,  
you have room for  $t = 3$  digits, and your exponents are from  $[-2, 3]$ .

There are several ways to represent  $1\frac{1}{2}$ :  $1.1 \times 2^0$ ,  
 $0.11 \times 2^1$ ,  $11.0 \times 2^{-1}$ . Choose the normalized form —

There is one digit to the left of the radix point for non-zero quantities

non-zero

What's the smallest positive number you can represent in this system?

What's the largest positive number you can represent in this system?

$$1.00 \times 2^{-2}$$
$$1.11 \times 2^3$$
$$8 + \frac{8}{2} + \frac{8}{4} = 14$$

# a number list

A number-line of the entire list of positive numbers isn't evenly-spaced  
However the ratio of the gaps to the magnitude is roughly constant