

CSC165 winter 2013

Mathematical expression

Danny Heap

heap@cs.toronto.edu

BA4270 (behind elevators)

<http://www.cdf.toronto.edu/~heap/165/W13/>

416-978-5899

Course notes, chapter 4



Outline

more asymptotics

notes



worst case

denote the worst-case complexity for program P with input $x \in I$, where the input size of x is n as $W_P(n) = \max\{t_P(x) \mid x \in I \wedge \text{size}(x) = n\}$

The upper bound $W_P \in \mathcal{O}(U)$ means

$$\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B$$

$$\Rightarrow \max\{t_P(x) \mid x \in I \wedge \text{size}(x) = n\} \leq cU(n)$$

$$\text{That is: } \exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall x \in I, \text{size}(x) \geq B$$

$$\Rightarrow t_P(x) \leq cU(\text{size}(x))$$

The lower bound $W_P \in \Omega(L)$ means

$$\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B$$

$$\Rightarrow \max\{t_P(x) \mid x \in I \wedge \text{size}(x) = n\} \geq cL(n)$$

$$\text{That is: } \exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B$$

$$\Rightarrow \exists x \in I, \text{size}(x) = n \wedge t_P(x) \geq cL(n)$$



bounding a sort

```
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 $W_{IS} \in \mathcal{O}(n^2)$.

big-oh of n^2

We know, or have heard, that all quadratic functions grow at “roughly” the same speed. Here’s how we make “roughly” explicit.

$$\mathcal{O}(n^2) = \{f : \mathbb{N} \mapsto \mathbb{R}^{\geq 0} \mid \exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B \Rightarrow f(n) \leq cn^2\}$$

Those are a lot of symbols to process. They say that $\mathcal{O}(n^2)$ is a set of functions that take natural numbers as input and produce non-negative real numbers as output. An additional property of these functions is that for each of them you can find a multiplier c , and a breakpoint B , so that if you go far enough to the right (beyond B) the function is bounded above by cn^2 .

In terms of limits, this says that as n approaches infinity, $f(n)$ is no bigger than cn^2 (once you find the appropriate c).



prove $W_{\text{IS}} \in \mathcal{O}(n^2)$

prove $W_{\text{IS}} \in \Omega(n^2)$

maximum slice

$$|L| = n$$

LG = Loop Guard

● executed 1
● executed \leq for each $i \rightarrow n$
● executed \leq for each $j \rightarrow n^2$
 $i, \dots, j-1 \rightarrow n^3$
● \leq each k

```
def max_sum(L):  
    """maximum sum over slices of L"""
```

```
    max = 0
```

```
    i = 0
```

```
    while i < len(L):
```

```
        j = i + 1
```

```
        while j <= len(L):
```

```
            sum = 0
```

```
            k = i
```

```
            while k < j:
```

```
                sum = sum + L[k]
```

```
                k = k + 1
```

```
            if sum > max:
```

```
                max = sum
```

```
            j = j + 1
```

```
            i = i + 1
```

```
    return max
```

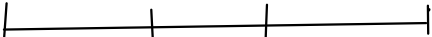
$$W_{ms}(n) \in \Theta(n^3)$$

$$\begin{aligned} W_{ms}(n) &\leq 4 + 4n + 7n^2 + 3n^3 \\ &\leq 18n^3 \end{aligned}$$

$\# n \geq 1$



maximum slice

```
def max_sum(L) : 
    """maximum sum over slices of L""" # show  $\lceil \frac{n}{3} \rceil \geq n-1$ 
```

```
    max = 0
```

```
    i = 0
```

```
    while i < len(L) :
```

```
        j = i + 1
```

```
        while j <= len(L) :
```

```
            sum = 0
```

```
            k = 1
```

```
            while k < j :
```

```
                sum = sum + L[k]
```

```
                k = k + 1
```

```
            if sum > max :
```

```
                max = sum
```

```
            j = j + 1
```

```
            i = i + 1
```

```
    return max
```

$$\lceil \frac{5}{3} \rceil = 2$$

$i = 0$

$\text{while } i < \text{len}(L) :$

$j = i + 1$

$\text{while } j \leq \text{len}(L) :$

$\text{sum} = 0$

$k = 1$

$\text{while } k < j :$

$\text{sum} = \text{sum} + L[k]$

$k = k + 1$

$\text{if } \text{sum} > \text{max} :$

$\text{max} = \text{sum}$

$j = j + 1$

$i = i + 1$

return max

$0, 1, \dots, \lceil \frac{n}{3} \rceil - 1$

j takes at least for each i

$n - \lceil \frac{n}{3} \rceil + 1, \dots, n - \lceil \frac{n}{3} \rceil + \lceil \frac{n}{3} \rceil$

$k = \lceil \frac{n}{3} \rceil - 1, \dots, n - \lceil \frac{n}{3} \rceil$

$\# n - \lceil \frac{n}{3} \rceil + 1 > \lceil \frac{n}{3} \rceil - 1$

$\# n + 2 > 2 \lceil \frac{n}{3} \rceil$

$\# n + 2 > 2 \frac{n+3}{3} > 2 \lceil \frac{n}{3} \rceil$

$\# 3n + 6 > 2n + 6$

$\# n > 0$

$\Omega(n^3)$

$$\# n - \lceil \frac{n}{3} \rceil + 1 \geq \lceil \frac{n}{3} \rceil$$

$$\geq \lceil \frac{n}{3} \rceil$$

Thus, $WMS(n) \geq \lceil \frac{n}{3} \rceil \cdot \lceil \frac{n}{3} \rceil \cdot \lceil \frac{n}{3} \rceil \geq \frac{n^3}{27}$

maximum slice

```
def max_sum(L) :  
    """maximum sum over slices of L"""  
    max = 0  
    i = 0  
    while i < len(L) :  
        j = i + 1  
        while j <= len(L) :  
            sum = 0  
            k = i  
            while k < j :  
                sum = sum + L[k]  
                k = k + 1  
            if sum > max :  
                max = sum  
            j = j + 1  
        i = i + 1  
    return max
```

make this
quadratic

Con Sum.

$L[j]$

challenge 2
make it
linear



Notes

Notes