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 $W_P(n) = \max\{t_P(x) \mid x \in I \land \text{size}(x) = n\}$

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

\[
\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B
\Rightarrow \max\{t_P(x) \mid x \in I \land \text{size}(x) = n\} \leq cU(n)
\]
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

\[
\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B
\Rightarrow \max\{t_P(x) \mid x \in I \land \text{size}(x) = n\} \geq cL(n)
\]
That is: $\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B$

\[
\Rightarrow \exists x \in I, \text{size}(x) = n \land t_P(x) \geq cL(n)
\]
def IS(A):
    """ IS(A) sorts the elements of A in non-decreasing order """
    n = len(A)
    for i in range(1, n):
        t = A[i]
        j = i
        while j > 0 and A[j-1] > t:
            j = j-1
        A[j] = t
    return A

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

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

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 $W_{IS} \in \mathcal{O}(n^2)$
prove $W_{IS} \in \Omega(n^2)$
maximum slice

$|L| = n$

$$L_G = \begin{array}{c}
\text{loop} \\
\text{guard}
\end{array}$$

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

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

$W_{ms}(n) \leq 4 + 4n + 7n^2 + 3n^3 \leq 18n^3 \neq n \geq 1$
maxim um slice

```python
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
```

Thus, \( W_{max}(n) \geq \left\lfloor \frac{n}{3} \right\rfloor \cdot \left\lfloor \frac{n}{3} \right\rfloor \cdot \left\lfloor \frac{n}{3} \right\rfloor \geq \frac{n^3}{27} \)
maximum slice

```python
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
Challenge 2
make it linear

Computer Science
UNIVERSITY OF TORONTO