CSC165 fall 2014

Mathematical expression

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cdf.toronto.edu/~heap/165/F14/
416-978-5899

Course notes, chapter 4

Outline

counting costs

want a coarse comparison of algorithms "speed" that ignores hardware, programmer virtuosity

which speed do we care about: best, worst, average? why?

define idealized "step" that doesn't depend on particular hardware and idealized "time" that counts the number of steps for a given input.

linear search

```
def LS(A,x) :
  """ Return index i such that x == L[i].  Otherwise, ret
    i = 0
    while i < len(A) :
        if A[i] == x :
            return i
        i = i + 1
    return -1
```

Trace `LS([2,4,6,8],4)`, and count the time complexity $t_{\mathrm{LS}}([2,4,6,8],4)$

What is $t_{\mathrm{LS}}(A,x)$, if the first index where x is found is j?

What is $t_{\mathrm{LS}}(A,x)$ is x isn't in A at all?

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

$$\exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall n \in \mathbb{N}, n \geq B$$
$$\Rightarrow \max\{t_P(x) \mid x \in I \wedge \text{size}(x) = n\} \leq cU(n)$$
$$\text{That is:} \quad \exists c \in \mathbb{R}^+, \exists B \in \mathbb{N}, \forall x \in I, \text{size}(x) \geq B$$
$$\Rightarrow t_P(x) \leq cU(\text{size}(x))$$

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

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

bounding a sort

```
def IS(A) :
    """ IS(A) sorts the elements of A in non-decreasing ord
    i = 1
    while i < len(A) :
        t = A[i]
        j = i
        while j > 0 and A[j-1] > t :
            A[j] = A[j-1] # shift up
            j = j-1
        A[j] = t
        i = i+1
```

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

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

```
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
```

Notes

annotated slides

- friday's annotated slides