CS3000

5/7 - Weds.

Admin

- APP1 today!

Agenda

1. $T(n) \rightsquigarrow$ complexity class
2. sorting an array
3. correctness / efficiency
   APP1

0. complexity classes

Algo design $\rightsquigarrow$ pswdocode $\rightarrow$ $T(n) \rightsquigarrow$ complexity class

$g(n) = n \lg n \quad 3 \quad 4^n$

$n! \quad \lg n \quad 5n$

$n^2 \quad n^4$

slowest to fastest

3

$\lg n$

$5n$

$n \lg n$

n^2

n^4

$\vdots$

4^n

$\vdots$

$n!$

1. $T(n) \rightsquigarrow$ complexity class

LINEARSEARCH(A, n, key)	cost	# times	
1 **for** $i = 1$ **to** n	c_1	$n+1$	n
2 **if** $A[i] == key$	c_2	n	n
3 **return** i	c_3	0	1
4 **return** NIL	c_4	1	0

$T(n) = (c_1 + c_2)n + (c_1 + c_4)$

- worst case: key not found
- worst case: key is last item in A

$n = 4$

$9, 11, 12, 7$

$key = 7$

$i = 1 \quad 1 \leq 4$? yes

if $A[1] == 7$

$i = 2 \quad 2 \leq 4$? yes

if $A[2] == 7$

$i = 3 \quad 3 \leq 4$? yes

if $A[3] == 7$

$i = 4 \quad 4 \leq 4$? yes

if $A[4] == 7$

return 4

↳ $T(n) = c_1 \cdot n + c_2(n \quad + c_3 \cdot 1 + c_4 \cdot 0$

$= c_1 \cdot n + c_2 \cdot n \quad + c_3$

$= (c_1 + c_2) n \quad + c_3$

- algo function: $f(n)$
- class function: $g(n)$

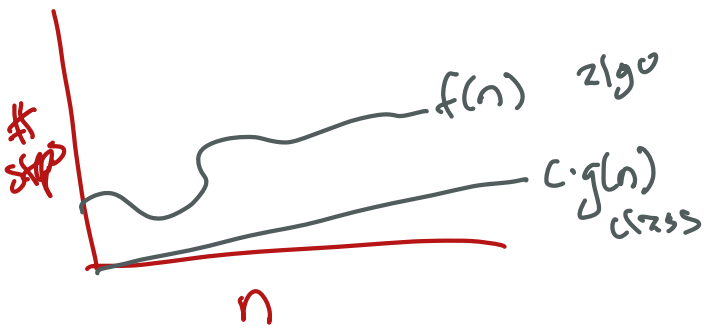
$g(n)$ is a bound for $f(n)$

upper bound $f(n) = O(g(n))$ iff

$f(n) \leq c \cdot g(n)$

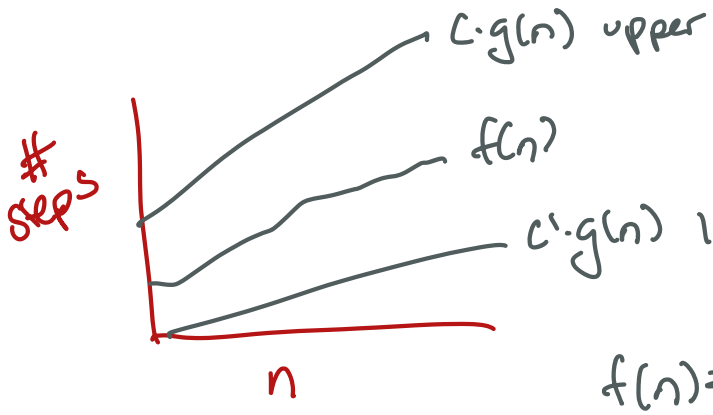
lower bound $f(n) = \Omega(g(n))$ iff

$f(n) \geq c \cdot g(n)$



$T(n) \rightsquigarrow$ complexity class

- remove coeffs
- remove lower order terms
- gives a tight bound



$f(n) = \Theta(g(n))$

(ex) $f(n) = 2n + 2$

- drop coeffs, lower order terms

~~$2n + 2$~~ $\rightsquigarrow$ n

$f(n) = \Theta(n)$

$g(n) = n$

$\exists\ c, c'$ such that

$f(n) \leq c \cdot g(n)$ and $f(n) \geq c' \cdot g(n)$

$2n + 2 \leq 4 \cdot n$

$2n + 2 \geq 1 \cdot n$

$f(n) = n^2 + 4n$

$= \Theta(n^2)$

2. Sorting an Array

↳ input: array A, length n

output: permutation of A such that $a_1 \leq a_2 \leq a_3 \leq \ldots \leq a_n$

- assume: can use comparison on any two objects
- if we modify A, we don't need to return it

Insertion Sort

→ pseudo → $T(n)$

- consider A two subarrays, sorted and unsorted
- take next unsorted element, put in its correct position in sorted array

ex

3	13	11	8	4
1	2	3	4	5

→ pseudocode

- loop using i
- A[i] is "next" unsorted element

Qs:

- what positions do I compare A[i] to?
- if A[i] > do what?
- else do what?

↳ go to next element to the left

Stop! in final position!

Sorted

3

3, 13

3, 11, 13

3, 8, 11, 13

3, 4, 8, 11, 13

i-1
i-2
i-3
⋮
1

3. Correctness + Efficiency

↳ does it work?

prove correctness: {loop invariant

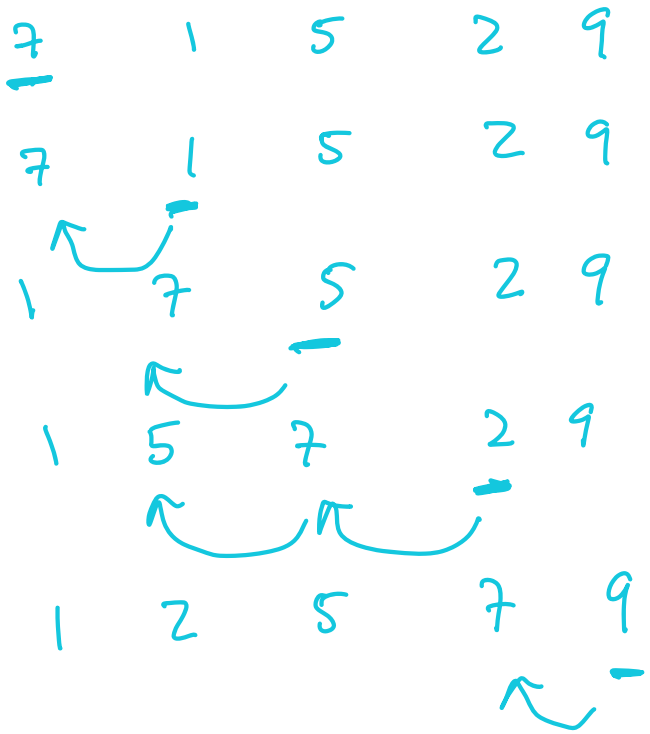
- holds before, during, after a loop

Insertion Sort loop invariant:

- A[i] is "next" unsorted element
- A[1..i-1] is correctly sorted

```
INSERTIONSORT(A, n)
1  for i = 2 to n
2      key = A[i]
3      j = i − 1
4      while j > 0 and A[j] > key
5          A[j + 1] = A[j]
6          j = j − 1
7      A[j + 1] = key
```

ex



i=2
key = A[2]=1
j=2-1=1

i=3
key=3-1=2

APP1!

ex

1	2	3	6	7
~~7~~	~~3~~	~~6~~	~~7~~	~~9~~
key			j	i j+1