CS3000
5/9 (Tues.)

Admin

- First recitation today!
- Short HW1 out, due 5/11 9pm
- Please say name in class

Agenda

1. Complexity classes
2. Sorting
3. Algorithm Analysis

Recap

- Linear search worst case scenario
  key is not in A
- w.c. # steps
  $2n+2$
- w.c. bound
  $\Theta(n)$
- $\lg = \log_2$ $\lg 16$? 4 $\lg 64$? 6 $2^x = 16$ $2^x = 64$
- $\sum_{i=1}^{n} i = 1+2+\ldots+n$ $\frac{(n)(n+1)}{2}$

$$\sum_{i=1}^{n-1} i \quad \frac{n(n-1)}{2}$$

1. Complexity Classes

Worst case linear search

#steps $2n+2 \rightarrow \Theta(n)$

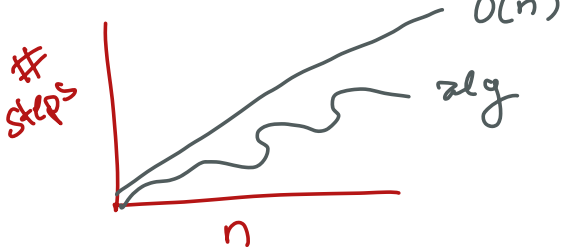
↳ complexity class

$2n+2$ $\quad 4n+18$ $\quad 2n$ } drop coefficients, drop lower-order terms

$2n+3$ $\quad \frac{1}{2}n+5$

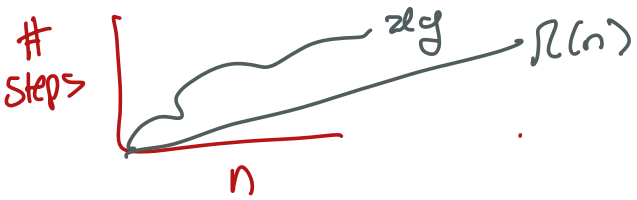
$\Theta(n)$

#steps as n grows arbitrarily large



→ upper bound

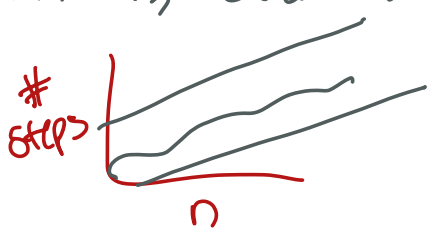
$n \lg \leq c \cdot n$



→ lower bound

$n \lg \geq c' \cdot n$

Drop coefficients, lower order terms ⟶ tight bound



$c' n \leq n \lg \leq c \cdot n$

$\Theta(n)$

linear search:

$2n+2$

↳ $\Theta(n)$

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

$c' \quad\quad\quad c$

worst case ≠ upper bound

Practice

# steps	band	
48	$\Theta(1)$	constant
$2 \lg n$	$\Theta(\lg n)$	log
$18n + 3/2$	$\Theta(n)$	linear
$\frac{1}{2} n \lg 2n$	$\Theta(n \lg n)$	sorting band
$5n^2 + 18n - 2$	$\Theta(n^2)$	quadratic
$5n^3 + 252$	$\Theta(n^3)$	cubic
$2^n + n^2$	$\Theta(2^n)$	exponential
$(n)(n-1)(n-2) \ldots (1)$	$\Theta(n!)$	factorial

2. Sorting

input: array A

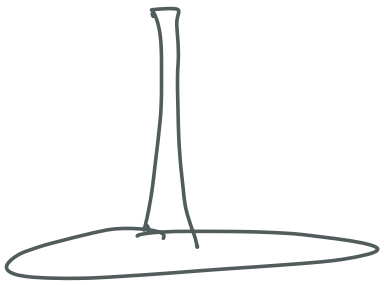
output: modified array A where

$$a[1] \leq a[2] \leq \ldots \leq a[n]$$

A good sort:

- readable
- efficient in time
- efficient in space
- correct

Today's: Selection Sort



- repeatedly finding the element from the unsorted array that goes "next"

Selection Sort Implementation

- Sort "in place"
- Loop i that tracks the next position
- second loop to find the min that hasn't been sorted

```
SELECTION(A)
 for i = 1 to A.length-1        ⟶ i is position where next smallest element goes
    min = i
    for j = i+1 to A.length
        if A[j] < A[min]        ⟶ Find min element A[i..n]
            min = j
    swap A[min], A[i]           ⟶ Put the min A[i..n] in position i
```

10:49

3. Algorithm Analysis ⟵ efficiency ★, correctness

```
SELECTION (A)                                   cost    times
1.  for i = 1 to A.length-1                      1       n
2.      min = i                                  1       n-1
3.      for j = i+1 to A.length                  1       Σ_{i=1}^{n-1} t_i
4.          if A[j] < A[min]                     1       Σ_{i=1}^{n-1} (t_i - 1)
5.              min = j                          1       Σ_{i=1}^{n-1} (t_i - 1)
6.      swap A[min], A[i]                        1       n-1
```

Inner for loop (the j loop)

- # times it runs depends on the value of i

t_i = # times the j loop test executes when $i = 1, 2, \ldots, n-1$

ex

$i = 1$
$j = 2, 3, \ldots, n$ — loop test: n times

$i = 2$
$j = 3, 4, \ldots, n$ — loop test: n - 1

...

$i = n-1$
$j = n$ — loop test: 2

express t_i in terms of n, i

$$t_i = n - i + 1$$

Formally show n-time...

$$n + (n-1) + (n-1) + \sum_{i=1}^{n-1} t_i + \sum_{i=1}^{n-1} (t_i - 1) + \sum_{i=1}^{n-1} (t_i - 1)$$

$$\boxed{\begin{aligned} t_i &= n - i + 1 \\ t_i - 1 &= n - i \end{aligned}}$$

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

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

$$= 3n - 2 + n^2 - 1 + n^2 - n + n^2 - n - 3 \cdot \sum_{i=1}^{n-1} i$$

$$= 3n^2 + n - 3 - 3\left[\frac{(n)(n-1)}{2}\right]$$

$$= 3n^2 + n - 3 - 3\left(\frac{n^2 - n}{2}\right)$$

$$= 3n^2 + n - 3 - \frac{3}{2}n^2 + \frac{3}{2}n$$

$$= \frac{3}{2}n^2 + \frac{5}{2}n - 3$$

$\boxed{\Theta(n^2)}$ Best + worst case

$\geq$ have shown $\leq$ n time $\leq$:)

What about correctness?

- loop invariant
- true before/after a loop
- argue init, maintenance, termination

```
SELECTION (A)
 for i = 1 to A.length-1
    min = i
    for j = i+1 to A.length
        if A[j] < A[min]
            min = j
    swap A[min], A[i]
```

Loop invariant:
A[1..i]
↳ sorted

1. Init i=1
A[1..i] = A[1]
↳ trivially sorted

2. maintenance
assume ok A[1..i-1]
inner loop finds
min A[i..n] assume ok
min is put in A[i]
i is incremented

3. Termination
when outer loop finishes,
i = n
loop invariant tells us
A[1..n] is sorted!

ex = A = <8, 3, 1, 9>
i=1 ↳min

end of loop: swap
A = <1, 3, 8, 9>
i=2
A = <1, 3, 8, 9>
i=3
A = <1, 3, 8, 9>