CS3000
6/16 - mon.

Admin

- exam 2 graded!
  - ↳ regrades till 6/18 9pm
- XC exam @ 6/18 during class
- Second chance HW 6/18 9pm

Agenda

1. Expected run-time
2. Randomized Quicksort
3. Rand-Select

1. Expected Runtime

↳ run-time of an algorithm

usually... $\Theta()$ upper bound + lower bound

↳ usually... worst case scenario

(ex) Linear Search

b.c. $\Theta(1)$ first thing!

w.c. $\Theta(n)$ target not in array

Insertion Sort

b.c. $\Theta(n)$ sorted!

w.c. $\Theta(n^2)$ reverse sorted

w/same algo, in b.c. or w.c. depending on
1. luck
2. adversary?

↳ can't avoid bad luck

instead, in w.c. scenario, make it luckier w/ insert $\boxed{\text{randomization}}$

1. shuffle input
2. randomize selection

↳ now, compute run-time with a probabilistic model

How do we compute probabilistic run-time?

expected run-time ⟿ Expected Value

$$EV[X] = \sum P_r(S_i) \cdot X_i$$

↳ prob of outcome → value of the outcome

$EV[\text{Jeop}] = Pr(\text{right}) \cdot \text{bet} + Pr(\text{wrong}) * -\text{bet}$

$= (.8) \cdot 1000 + (.2)(-1000)$

80% chance in algo category :)

↳ in run time

$EV[\text{algo}] = Pr(\text{run-time}) \cdot \text{run-time} + \ldots$

$= (.75)(n) + (.25)(n^2)$

↳ 75% chance linear
25% chance n^2

> EV is real-world average!

↳ if EV better than w.c., we win! (we beat them)

2. Randomized Quicksort

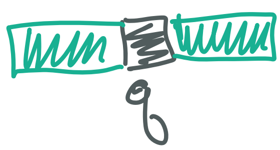
```
QSORT(A, p, r)
  if p < r
      q = PART(A, p, r)
      QSORT(A, p, q-1)
      QSORT(A, q+1, r)
```

→ choose A[r] as pivot
- smaller on left
- larger on right
- return position of pivot (never moves again!)

(always choose A[r] as pivot arbitrary, but not random)

Best Case (lucky)

$T(n) = T(n/2) + T(n/2) + c \cdot n$

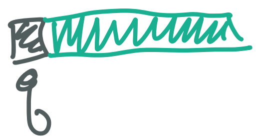


roughly even split

Worst Case

$T(n) = T(n-1) + T(0) + c \cdot n$

$= \Theta(n^2)$



very uneven split

→ $= \Theta(n \lg n)$ split is 1/2, 1/2 in best case

variation: split is 2/3, 1/3

$T(n) = T\left(\frac{2}{3}n\right) + T(n/3) + c \cdot n$

$= \Theta(n \lg n)$

variation: split is 3/4, 1/4

$T(n) = T(\frac{3}{4}n) + T(\frac{n}{4}) + c \cdot n$

$= \Theta(n \lg n)$

$\boxed{\text{Tasks}}$:

1. intro randomization ⇝ randomly choose pivot
2. decide on "good" split ⇝ balanced?

what is balance?

ex) 4 elements in array

_ p _ _ balanced!

_ _ p _

p _ _ _ unbalanced :(

_ _ _ p

generalized...

- balanced = p is between 1/4 n to 3/4 n

- unbalanced = p is anywhere else

Balanced prob = 50%

Balanced split = $T(n) = T(\frac{3}{4}n) + T(\frac{n}{4}) + c \cdot n$

Expected run-time: (.5)(balanced) + (.5)(unbalanced)

$$\hookrightarrow T(n) \le \frac{1}{2}\left(T(\tfrac{3}{4}n) + T(\tfrac{n}{4}) + c \cdot n\right) + \frac{1}{2}\left(T(n) + c \cdot n\right)$$

$$\le \frac{1}{2}T(n) + \frac{1}{2}cn + \frac{1}{2}cn + \frac{1}{2}T(\tfrac{3}{4}n) + \frac{1}{2}T(\tfrac{n}{4})$$

$$\frac{1}{2}T(n) \le c \cdot n + \frac{1}{2}T(\tfrac{3}{4}n) + \frac{1}{2}T(\tfrac{n}{4})$$

$$T(n) \le T(\tfrac{3}{4}n) + T(\tfrac{n}{4}) + 2cn$$

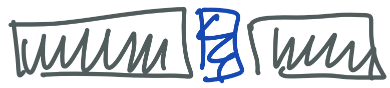
$= \Theta(n \lg n)$

3. Rand-Select

- given an array A
  - unsorted
  - length n

- want: k^{th} smallest element in A

Brute force:
- sort A
- return value at position k


1 ... k-1 k k+1 n

if sorted

- How would you use randomized-partition to solve in linear time?

- shuffle?
- random pivot?

RANDSELECT
- shuffle A
  (any input equally likely)
- partition on left element A[p]
- q = final position of pivot
  - q==k? done!
  - q<k? throw away left, recurse on right
  - q>k? throw away right, recurse on left

ex: 11, 10, 8, 13, 9, 3

n=6
k=2

partition

10, 8, 9, 3 $\boxed{11}$, 13 → throw away

5

10, 8, 9, 3

partition

8, 9, 3 | $\boxed{10}$

4 throw away

8, 9, 3

partition

3, $\boxed{8}$ 9

2 = k! done

return A[2]=8