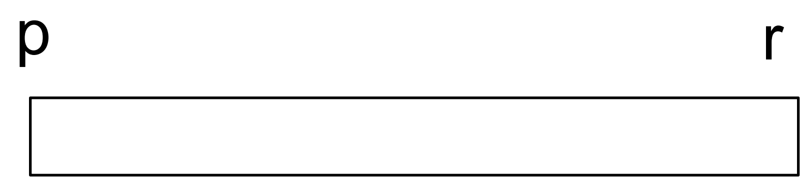
Data Structures and Algorithm Analysis (CSC317)

Randomized Algorithms (part 3)

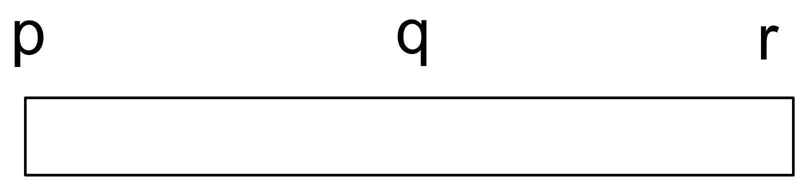
Quicksort



Quicksort(A, p, r)

1. **If** p<r
   2. q = Partition(A,p,r)
   3. Quicksort(A,p,q-1)
   4. Quicksort(A,q+1,r)

Quicksort



Quicksort(A, p, r)

1. **If** p<r
   2. q = Partition(A,p,r)
   3. Quicksort(A,p,q-1)
   4. Quicksort(A,q+1,r)

Quicksort on average run time

- We'll prove that average run time with random pivots for any input array is O(n log n)

- Randomness is in choosing pivot

- Average as good as best case!

Quicksort on average runtime

Prelims:

- Most work of Quicksort in comparisons
- Each call to partition is constant plus number of comparisons in for loop
- Let X = total number of comparisons in all calls to Partition

Quicksort on average runtime

- Let X = total number of comparisons in ***all*** calls to Partition

- Rename smallest and largest elements in A as
$z_1, z_2, ..z_n$

Example: [4 1 2 3]
z1=1; z2=2; z3=3; z4=4

This is for analysis, not that we presort!

Quicksort on average runtime

- Let X = total number of comparisons in all calls to Partition

- Rename smallest and largest elements in A as $z_1, z_2, ..z_n$

- Consider $z_i ; z_j$ with indices i<j

$X_{ij} =$ 1 if z_i and z_j compared
0 if not compared

Quicksort on average runtime

How many times will zi and zj be compared?

Example:
A=[8 1 6 4 0 3 9 5]; zi=3; zj=9

When will two elements be compared?

Only if one of them (3 or 9) is chosen as a pivot, since in Partition, each element compared only with pivot. They will then never be compared again, since pivot is not in the subsequent recursions to Partition

Quicksort on average runtime

How many times will zi and zj be compared?

Example:
A=[8 1 6 4 0 3 9 5]; zi=3; zj=9

When will two elements ***not*** be compared?

If pivot=5, then none of [8 6 9] will be compared to [1 4 0 3], so zi and zj not compared.
Not in this call of partition, or any other call, since these sets will then be separated

Quicksort on average runtime

Probability that zi and zj be compared?

Consider i<j and set $Z_{ij} = z_i, z_{i+1}, .. z_j$ Not necessarily in order in array A

- This set will remain together in calls to Partition, unless one of them is a pivot

- Because otherwise pivot will be smaller than i or larger than j, so these will remain together

Quicksort on average runtime

Probability that zi and zj be compared?

Consider i<j and set $Z_{ij} = z_i, z_{i+1}, .. z_j$

- If i or j chosen first as pivot within this set, they'll be compared
- Otherwise, another element will be chosen as pivot, and they will be separated
- Since (j-i+1) elements in this set, the prob of any element as pivot: $\frac{1}{j-i+1}$

Quicksort on average runtime

Pr { zi compared to zj } =

Pr { zi or zj chosen first as pivot in Zij } =

(since mutually exclusive)

Pr { zi first element chosen from Zij } +
Pr { zj first element chosen from Zij } =

$$\frac{1}{j-i+1}+\frac{1}{j-i+1}=\frac{2}{j-i+1}$$

Quicksort on average runtime

- Let X = total number of comparisons in all calls to Partition

$$X_{ij} = \begin{cases} 1 \text{ if } z_i \text{ and } z_j \text{ compared} \\ 0 \text{ if not compared} \end{cases}$$

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

All possible comparisons i and j

Quicksort on average runtime

- Let X = total number of comparisons in all calls to Partition

$X_{ij} =$ 1 if z_i and z_j compared
0 if not compared

$$E[X] = E[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]$$

Prob zi compared to zj

Quicksort on average runtime

$$E[X] = E[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] =$$

Prob zi compared to zj

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$$

Quicksort on average runtime

$$E[X] = E[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}] =$$

$$\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} =$$ (Change of var: k = j-i)

$$\sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} <$$

$$\sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k} =$$

$$\sum_{i=1}^{n-1} O(\log n) =$$

$$O(n \log n)$$

Quicksort on average run time

- We'll shown that average run time with random pivots for any input array is O(n log n)
- Randomness is in choosing pivot
- Average as good as best case!

Order statistics: another use of partition!

- Array n unsorted
- Find kth smallest
- k=1: Minimum

Order statistics: another use of partition!

- Array n unsorted
- Find kth smallest element
- k=1: Minimum
- $k = \left\lfloor \frac{n+1}{2} \right\rfloor ; \left\lceil \frac{n+1}{2} \right\rceil$?

Order statistics: another use of partition!

- Array n unsorted
- Find kth smallest element
- k=1: Minimum
- $k = \left\lfloor \frac{n+1}{2} \right\rfloor ; \left\lceil \frac{n+1}{2} \right\rceil$: Median

Order statistics

- Array n unsorted
- 1st, 2nd, 3rd, smallest or largest; median …
- Another use of Partition

Order statistics: start simple (1st order)

Minimum(A)
1. Min = A[1]
2. **for** i=2 to A.length
3. **if** min>A[i]
4. min = A[i]
5. **return** min

Worst case?
Best case?

Order statistics: start simple (1st order)

Minimum(A)
1. Min = A[1]
2. **for** i=2 to A.length
3.   **if** min>A[i]
4.   min = A[i]
5. **return** min

Worst case? $\Theta(n)$
Best case? $\Theta(n)$

Order statistics

Selection problem – more general problem

Input: set A with n distinct numbers

Output: find ith smallest element

Order statistics

Selection problem – more general problem

Input: set A with n distinct numbers

Output: find ith smallest element

Upper bound?

Order statistics

Selection problem – more general problem

Input: set A with n distinct numbers

Output: find ith smallest element

Upper bound?

We can solve in $O(n \log n)$ since we can always sort and find ith index in array

Order statistics

Selection problem – more general problem

Input: set A with n distinct numbers

Output: find ith smallest element

Upper bound?

We can solve in $O(n \log n)$ since we can always sort and find ith index in array. We would like to do better!

Selection problem

Selection problem – more general problem

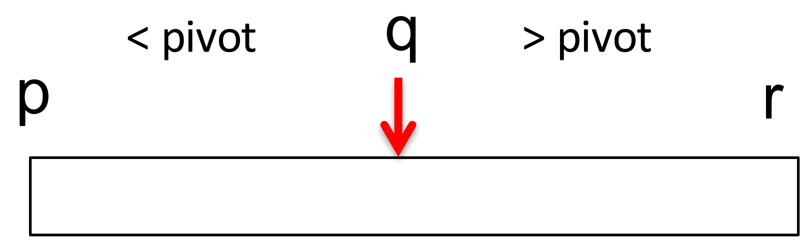
Input: set A with n distinct numbers

Output: find ith smallest element

$O(n)$ on average with randomized algorithm

Amazing that (at least on average) similar to finding just minimum!

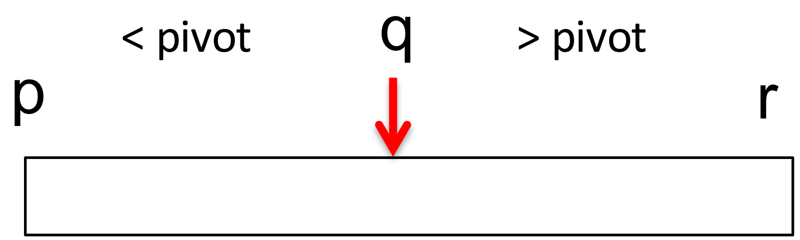
Selection problem




```
Randomized-Select(A,p,r,i)
if p==r   // base case
    return A[p]
q=Randomized-Partition(A,p,r)
k=q-p+1          // number elements from left up to pivot
if i==k          // pivot is the ith smallest!
    return A[q]
elseif i<k
    return Randomized-Select(A,p,q-1,i)   // ith smallest left
else return Randomized-Select(A,q+1,r,i-k) //on right
```

Selection problem

Randomized-Select(A,p,r,i)



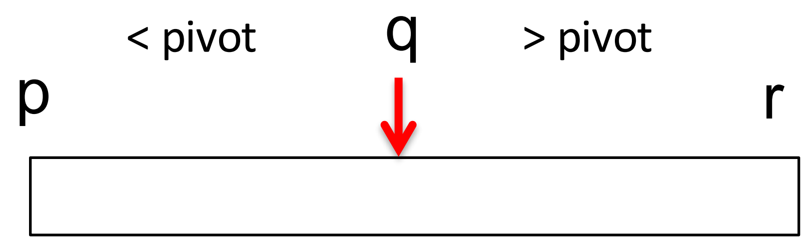
Example:

A = [4 1 6 5 3]

i=3

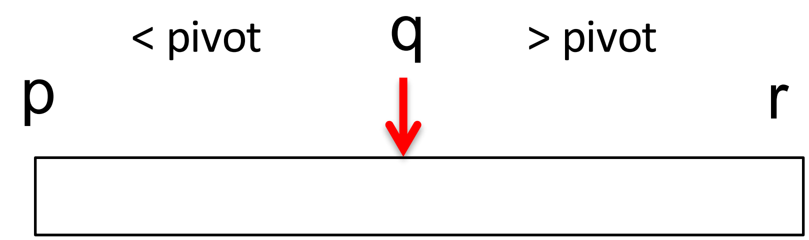
(find 2nd smallest; 3rd smallest element
On the board...)

Selection problem



How is this different from randomized version of Quicksort?

Selection problem



How is this different from randomized version of Quicksort?

Answer: Only one recursion (left or right); not two

Selection problem

Analysis

Worst case: $T(n) = T(n-1) + \Theta(n)$

$= \Theta(n^2)$

always partition around largest remaining element, and recursion on array size n-1

Worse than a good sorting scheme!

Selection problem

Analysis: But 1/10 to 9/10 good… $\Theta(n)$

$$T(n) = T\left(\frac{9n}{10}\right) + \Theta(n) =$$

Selection problem

Analysis: But 1/10 to 9/10 good… $\Theta(n)$

$$T(n) = T\left(\frac{9n}{10}\right) + \Theta(n)$$

Master theorem….

$$n^{\log_b a} = n^{\log_{10/9} 1} = n^0 = 1$$

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

$$T(n) = \Theta(n)$$

Selection problem

Average case solution also good! $\Theta(n)$

We won't prove, but similar to Quicksort….