CS140: Algorithms

Z Sweedyk
Lecture 4

Today

- Selection

Select

- Input: Set of (distinct) integers S and an integer k
- Output: k^{th} smallest integer in S

Select: special cases

- k=1
- k=n
- k=n/2

How fast can we solve these cases?

Select: Take 1

Select(S,k)
 Let x=S[0]
 Partition S into
  $S_1 = \{y \in S-\{x\} \mid y < x\}$
  $S_2 = \{y \in S-\{x\} \mid y > x\}$
 If $||S_1|| \geq k$ then return Select(S_1,k)
 Else if $||S_1|| = k-1$ then return x
 Else return Select(S_2, $k-||S_1||-1$)

Analysis of Select(S,k)

In worst we get

$T(n) \lesssim T(\_\_\_) + \_\_\_ = \Theta(\_\_\_)$

Suppose …

- Suppose we could choose x so that the recursive call is always on a set of size n/b, for some constant b>1.
- Then T(n)=T(___)+___

  =__________

To warm up …

Suppose we could choose x so that the recursive call is **typically** on a set of size n/b, for some constant b>1.

Randomized Select in Expected Linear Time

Randomized Select(S,k)

Choose x randomly from S

Partition S into

$S_1 = \{y \ \varepsilon\ S\text{-}\{x\} \mid y < x\}$

$S_2 = \{y \ \varepsilon\ S\text{-}\{x\} \mid y > x\}$

If $||S_1|| \geq k$ then return Randomized Select(S_1,k)

Else if $||S_1|| = k-1$ then return x

Else return Randomized Select(S_2, $k-||S_1||-1$)

Analysis of Randomized-Select

E[T(n)] = E[T(N)] + cn

Where N is the size of the input to the recursive call.

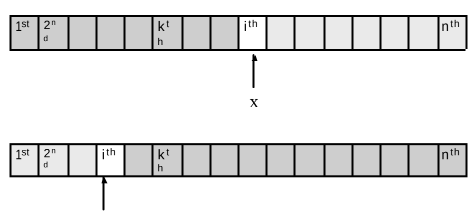
Analysis of Randomized-Select

$E[T(n)] = E[T(N)] + cn$

$= \Sigma_{i=1\ldots n} E[T(N) \mid rank(x)=i]\ P(rank(x)=i) + cn$

$= (1/n)\ \Sigma_{i=1\ldots n} E[T(N) \mid rank(x)=i] + cn$

N ≤ max(i-1, n-i) when rank(x)=i



Analysis of Randomized-Select

$E[T(n)] = E[T(N)\ ] + cn$

$= \Sigma_{i=1...n} E[T(N) \mid rank(x)=i]\ P(rank(x)=i) + cn$

$= (1/n)\ \Sigma_{i=1...n} E[T(N) \mid rank(x)=i] + cn$

$\leq (1/n)\ \Sigma_{i=1...n} E[T(max(i-1,n-i)] + cn$

$\leq (2/n)\ \Sigma_{i=\lceil(n+1)/2\rceil..n} E[T(i-1)] + cn$

Analysis of Randomized-Select

$E[T(n)] \leq (2/n)\Sigma_{i=\lceil(n+1)/2\rceil..n} E[T(i-1)]+cn$

$= O(n)$

Analysis of Randomized-Select

Claim: There exists a constant b such that $E[T(n)] \leq bn$ for all $n\geq$

- Base Case: hold for $b \geq c$
- Inductive Step
- $E[T(n)] \leq (2/n)\ \Sigma_{i=\lceil(n+1)/2\rceil..n} E[T(i-1)] + cn$
- $E[T(n)] \leq (2/n)\ \Sigma_{i=\lceil(n+1)/2\rceil..n} b\cdot(i-1) + cn$

  $\leq bn$ provided $b>2c$

Select: Take 2

What if all permutations of S are equally likely?

Select(S,k)

Let x=S[0]

Partition S into

$S_1 = \{y\ \varepsilon\ S-\{x\} \mid y < x\}$

$S_2 = \{y\ \varepsilon\ S-\{x\} \mid y > x\}$

If $||S_1|| \geq k$ then return Select(S_1,k)

Else if $||S_1|| = k-1$ then return x

Else return Select(S_2, $k-||S_1||-1$)

Analysis

- Let me wave my hands a bit ...

Deterministic Select in Linear Time

Select(S,k)

Choose a "good pivot" x ε S

Partition S into

$S_1 = \{y\ \varepsilon\ S-\{x\} \mid y < x\}$

$S_2 = \{y\ \varepsilon\ S-\{x\} \mid y > x\}$

If $||S_1|| \geq k$ then return Select(S_1,k)

Else if $||S_1|| = k-1$ then return x

Else return Select(S_2, $k-||S_1||-1$)

What is a good pivot?

We say x ε S is a good pivot if its rank is between n/c and (c-1)n/c for some constant c>1.

If we always choose a good pivot we get Θ(n) running time.

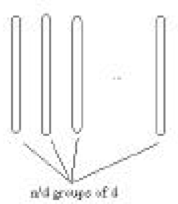
The thing ...

- Median of medians pivot

The next few slides are ANALYSIS – not the algorithm

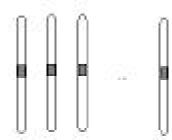
Median of medians pivot

- Divide the input into groups of d.



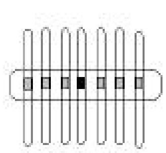
Median of medians pivot

- Divide the input into groups of d.
- Sort each group and mark its median.



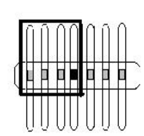
Median of medians pivot

- Divide the input into groups of d.
- Sort each group and mark its median.
- Sort the groups by their medians. Mark median of medians

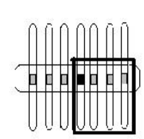


Median of medians pivot

- Elements in upper left quadrant are smaller than median of medians.

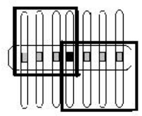


- Elements in lower right quadrant are larger than median of medians.



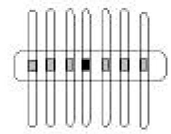
Median of medians pivot

- How many elements of S are smaller than the median of medians?
- How many are larger?



Median of medians

- Median of medians is a good pivot provided d satisfies the following:



BUT

- Finding the good pivot requires a recursive call to Select
- We hadn't counted on this …

New Analysis

1. Divide the input into groups of *5*. Find the median of each group.
2. Find the median of the medians.
3. Partition the input around the median of medians.
4. Recurse on appropriate set of the partition.

1. O(1) time per group, O(n) groups ➔ O(n)
2. T(n/5)
3. O(n)
4. T(3n/4)

Linear selection

$T(n) = T(n/5) + T(3n/4) + O(n) = \Theta(n)$

BUT BE CAREFUL OF DETAILS!