Overview

- Select in linear time
  - Randomized
  - Deterministic
- Average case performance of Quick-sort

Select

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

Select

- FIND-MIN is $\Omega(n) \Rightarrow$ Select is $\Omega(n)$
- Theorem: The general select problem can be solved in linear time.

Selection: Take 1

Select(S,k)

Choose x from S

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

$$T(n) \leq T(n-1) + cn = \Theta(n^2)$$

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(n/b)+cn

$$= cn \sum_{i=0\ldots m} 1/b^i \quad (m=\log_b n)$$

$$= O(n)$$

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.

"Average Case Analysis"

- Deterministic Algorithm: The expected running time when the input is chosen uniformly at random from all inputs of size *n*.
- **Randomized Algorithm: The expected running time for any (worst-case) input of size *n*.**

Random Selection in Expected Linear Time

Random Select(S,k)

Choose x randomly from 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, $k-\|S_1\|-1$)

Analysis of Random-Select

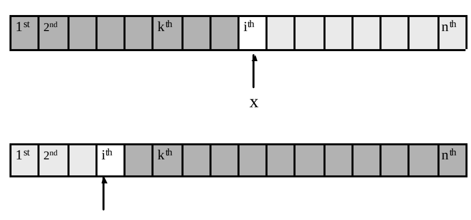
$E[T(n)] \leq \Sigma_{i=1..n}\ E[T(N_i)]\cdot Pr(rank(x)=i) + cn$

Where N_i is the size of the recursive call when rank(x)=i.

Analysis of Random-Select

$$E[T(n)] \leq \Sigma_{i=1..n}\ E[T(N_i)] \cdot Pr(rank(x)=i) + cn$$
$$= (1/n)\, \Sigma_{i=1..n}\ E[T(N_i)] + cn$$

$N_i \leq \max(i-1, n-i)$



Analysis of Random-Select

$E[T(n)] \le \Sigma_{i=1..n}\ E[T(N_i)] \cdot Pr(rank(x)=i) + cn$

$= (1/n)\ \Sigma_{i=1..n}\ E[T(N_i)] + cn$

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

Analysis of Random-Select

$E[T(n)] \le \Sigma_{i=1..n}\ E[T(N_i)] \cdot Pr(rank(x)=i) + cn$

$= (1/n)\ \Sigma_{i=1..n}\ E[T(N_i)] + cn$

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

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

$= O(n)$

Inductive proof – but be careful!

What is wrong with this picture? A proof that $n^2=O(n)$!!!!

- $1^2=O(1)$
- Assume $(n\text{-}1)^2=O(n\text{-}1)$
- Then $n^2 = (n\text{-}1+1)^2 = (n\text{-}1)^2 + 2(n\text{-}1) + 1$

  $= O(n\text{-}1) + O(n\text{-}1) + O(1)$

  $= O(n)$

Back to Basics

- Claim: There exists constant c and M such that $n^2 \le cn$ for all $n \ge M$.
- Proof:
  - Assume c and M are chosen so $M^2 \le cM$.
  - Let n-1be at least M. Suppose $(n\text{-}1)^2 \le c(n\text{-}1)$.
  - Then $n^2 = (n\text{-}1+1)^2 = (n\text{-}1)^2 + 2(n\text{-}1) + 1$

    $\le c(n\text{-}1) + 2(n\text{-}1) + 1$

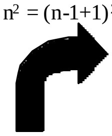
    $= cn + 2(n\text{-}1) - c + 1$
- Claim follows iff $c \ge 2(n\text{-}1) + 1$. But no constant c can satisfy this requirement.

What is wrong with this picture? A proof that $n^2=O(n)$!!!!

- $1^2=O(1)$
- Assume $(n\text{-}1)^2=O(n\text{-}1)$
- Then $n^2 = (n\text{-}1+1)^2 = (n\text{-}1)^2 + 2(n\text{-}1) + 1$

  $= O(n\text{-}1) + O(n\text{-}1) + O(1)$

  $= O(n)$



This is what is wrong!

Analysis of Random-Select

Claim: There exists constants d and M such that $E[T(n)] \le d \cdot n$ for all $n \ge M$.

- $E[T(n)] \le (2/n)\ \Sigma_{i=n/2..n}\ E[T(i\text{-}1)] + cn$
- $E[T(n)] \le (2/n)\ \Sigma_{i=n/2..n}\ d \cdot (i\text{-}1) + cn$

  $\le dn - d(n/4 - ½ + 1/n) + cn$

Overview

- Select in linear time
  - Randomized algorithm
  - **Deterministic algorithm**
- Average case performance of Quick-sort

Deterministic Selection in Linear Time

Select(S,k)

Choose a “good pivot” x ε S

Partition S into

S_1 = {y ε S-{x} | y < x}

S_2 = {y ε S-{x} | 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.

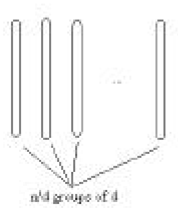
If we always choose a good pivot we get $\Theta(n)$ running time.

The thing …

- Median of medians pivot

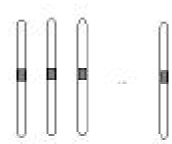
Median of medians pivot

- Divide the input into groups of d.



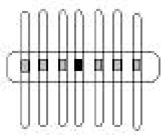
Median of medians pivot

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



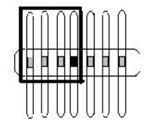
Median of medians pivot

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

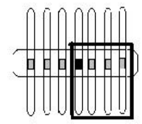


Median of medians pivot

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

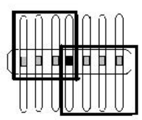


- Elements in lower right quadrant are larger than median of medians.



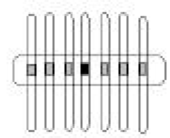
Median of medians pivot

- 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!

Overview

- Select in linear time
  - Randomized
  - Deterministic
- **Average case performance of Quick-sort**

Quick-sort(S)

- If $||S|| \leq 1$ then return
- Let s be the “median of medians” pivot of S
- $S1=\{t \in S-\{s\} \mid t > s\}$
- $S2=\{t \in S-\{s\} \mid t \leq s\}$
- Return Quick-sort(S1),s,Quick-sort(S2)

Quick-sort Analysis

- Read the text.