Design and Analysis
of Algorithms I

Linear-Time Selection

Deterministic Selection (Algorithm)

The Problem

For simplicity

Input : array A with n distinct numbers and a number

Output : i^{th} order statistic (i.e., i^{th} smallest element of input array)

10	8	2	4

3^{rd} order statistic

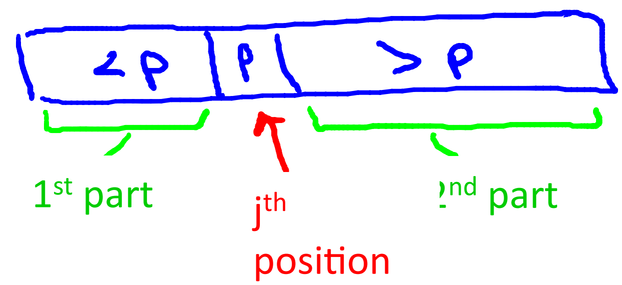
Example : median.

($i = (n+1)/2$ for n odd,

$i = n/2$ for n even)

Randomized Selection

Rselect (array A, length n, order statistic i)



0) if n = 1 return A[1]
1) Choose pivot p from A uniformly at random
2) Partition A around p
   let j = new index of p
3) If j = i, return p
4) If j > i, return Rselect(1st part of A, j-1, i)
5) [if j<i] return Rselect (2nd part of A, n-j, i-j)

Guaranteeing a Good Pivot

Recall : “best” pivot = the median ! (seems circular!)

Goal : find pivot guaranteed to be pretty good.

Key Idea : use “median of medians”!

A Deterministic ChoosePivot

ChoosePivot(A,n)

-- logically break A into n/5 groups of size 5 each
-- sort each group (e.g., using Merge Sort)
-- copy n/5 medians (i.e., middle element of each sorted group)
   into new array C
-- recursively compute median of C   (!)
-- return this as pivot

Tim Roughgarden

The DSelect Algorithm

DSelect(array A, length n, order statistic i)

ChoosePivot

1. Break A into groups of 5, sort each group
2. C = the n/5 “middle elements”
3. p = DSelect(C,n/5,n/10) [recursively computes median of C]
4. Partition A around p
5. If j = i return p
6. If j < i return DSelect(1st part of A, j-1, i)
7. [else if j > i] return DSelect(2nd part of A, n-j, i-j)

Same as before

Ter
ver

How many recursive calls does DSelect make?

- 0
- 1
- 2
- 3

Running Time of DSelect

Dselect Theorem : for every input array of length n, Dselect runs in O(n) time.

Warning : not as good as Rselect in practice

1) Worse constraints 2) not-in-place

History : from 1973

Blum – Floyd – Pratt – Rivest – Tarjan
('95) ('78) ('02) ('86)