



Design and Analysis
of Algorithms I

Linear-Time Selection

Deterministic
Selection (Algorithm)

The Problem

Input: array A with n distinct numbers *for simplicity*
and a number $i \in \{1, 2, \dots, n\}$.

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

Example: median.

$i = \frac{n+1}{2}$ for n odd,
 $i = n/2$ for n even)

10	8	2	4
----	---	---	---

\nearrow 3rd order statistic

Randomized Selection

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

⑥ if $n=1$ return $A[1]$

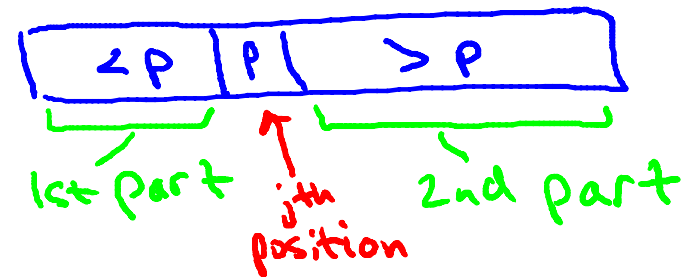
① choose pivot p from A
uniformly at random

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 MergeSort)
- 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

The DSelect Algorithm

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

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

Choose Pivot

Same as before

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 QSelect in practice
① worse constants ② not-in-place

History: from 1973.

Blum - Floyd - Pratt - Rivest - Tarjan
 '75 '70 '02 '06



Design and Analysis
of Algorithms I

Linear-Time Selection

Deterministic
Selection (Analysis)

The DSelect Algorithm

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

1. Break A into groups of 5, sort each group
2. C = the $n/5$ "middle elements"
3. $p = \text{DSelect}(C, n/5, n/10)$ [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, $n-j$, $i-j$)

Choose
Pivot

Same
as in
RSelect

What is the asymptotic running time of step 1 of the DSelect algorithm?

☐ $\theta(1)$

☐ $\theta(\log n)$

☒ $\theta(n)$

☐ $\theta(n \log n)$

Note: sorting an array with 5 elements takes ≤ 120 operations.

[why 120? take $m=5$ in our $6m(\log_2 m + 1)$ bound for MergeSort]

$$\begin{aligned} & \xrightarrow{\text{# groups}} 6 \cdot 5 \cdot (\log_2 5 + 1) \leq 120 \quad (\log_2 5 \leq 3) \\ \text{So: } & \leq \underbrace{n/5}_{\text{ops per group}} \cdot 120 = 24n = O(n) \text{ for all groups} \end{aligned}$$

The DSelect Algorithm

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

1. Break A into groups of 5, sort each group $\rightarrow \Theta(n)$
2. C = the $n/5$ "middle elements" $\rightarrow \Theta(n)$
3. $p = \text{DSelect}(C, n/5, n/10)$ [recursively computes median of C]
4. Partition A around p $\rightarrow \Theta(n)$ $T(\frac{n}{5})$
5. If $j = i$ return p
6. If $j < i$ return DSelect(1st part of A, $j-1$, i) $T(?)$
7. [else if $j > i$] return DSelect(2nd part of A, $n-j$, $i-j$) $T(?)$

Rough Recurrence

Let $T(n)$ = maximum running time of DSelect on an input array of length n .

There is a constant $c \geq 1$ such that :

$$\textcircled{1} T(1) = 1$$

$$\textcircled{2} T(n) \leq \underbrace{cn}_{\text{sorting the groups, partition}} + \underbrace{T\left(\frac{n}{5}\right)}_{\text{recursive call in line 3}} + \underbrace{T(?) }_{\text{recursive call in line 6 or 7}}$$

sorting the groups,
partition

recursive
call in line 3

recursive
call in
line 6 or 7

The Key Lemma

Key Lemma: 2nd recursive call (in the 6 or 7) guaranteed to be on an array of size $\leq \frac{7}{10}n$ (roughly).

Upshot: Can replace "?" by " $\frac{7}{10}n$ ".

Rough Proof: Let $k = \frac{n}{5} = \#$ of groups.

Let $x_i = i^{\text{th}}$ smallest of the k "middle elements".

[so pivot = $x_{k/2}$]

Goal: 7,30% of input array smaller than $x_{k/2}$,
7,30% is bigger.

Rough Proof of Key Lemma

Thought experiment:

Imagine we lay out elements of A in a 2-D grid:



Key point: $x_{k/2}$ bigger than 3 out of 5 (60%) of the elements in $\approx 50\%$ of the groups
 \Rightarrow bigger than 30% of A (Similarly, smaller than 30% of A)

Example

Input

7	2	17	12	13	8	20	4	6	3	19	1	9	5	16	10	15	18	14	11
---	---	----	----	----	---	----	---	---	---	----	---	---	---	----	----	----	----	----	----

After
Sorting
groups & S

2	7	12	13	17	3	4	6	8	20	1	5	9	16	19	10	11	14	15	18
---	---	----	----	----	---	---	---	---	----	---	---	---	----	----	----	----	----	----	----

our pivot!

The
grid:

20	19	17	18
8	16	13	15
6	9	12	14
4	5	7	11
3	1	2	10

Rough Recurrence (Revisited)

Analysis of Rough Recurrence