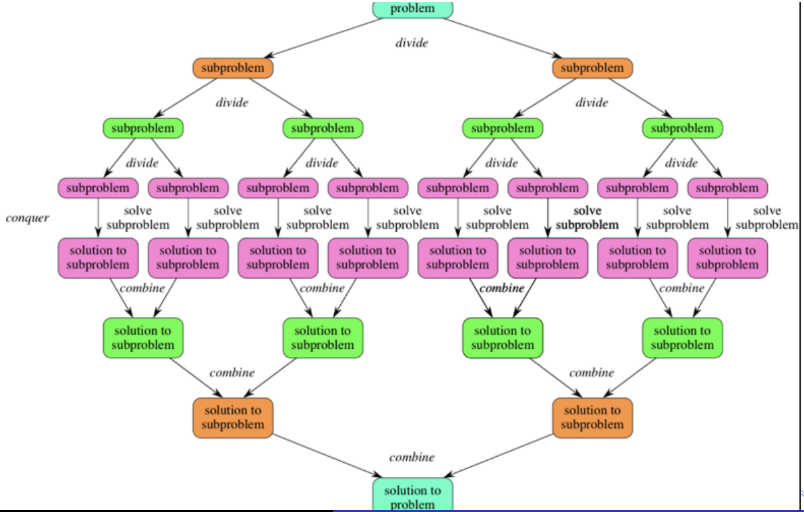
Divide-and-conquer: Selection



The divide-and-conquer strategy.

1. Break the problem into smaller subproblems,
2. recursively solve each problem,
3. appropriately combine their answers.

Known Examples:

- Binary search
- Merge-sort
- Quicksort
- Strassen matrix multiplication

Julius Caesar (I-BC)
"Divide et impera"

J. von Neumann
(1903-57) Merge sort

A basic Master Theorem

There are several versions of the Master Theorem to solve D&C recurrences. The one presented below is taken from DPV's book.

Theorem (DPV-2.2)

If $T(n) = aT(\lceil n/b \rceil) + O(n^d)$ *for constants* $a \geq 1, b > 1, d \geq 0,$ *then has asymptotic solution:*

$$T(n) = \begin{cases} O(n^d), & \text{if } d > \log_b a, \\ O(n^d \lg n), & \text{if } d = \log_b a, \\ O(n^{\log_b a}), & \text{if } d < \log_b a. \end{cases}$$

Master Theorems

- This basic Master Theorem does not provide always exact bounds.
- A different one can be found in CLRS's book providing exact bounds but leaving cases outside.
- For stronger versions look at Akra-Bazi Theorem or Salvador Roura Theorems

Selection

From 9.3 in CLRS

Selection

From 9.3 in CLRS

- We use the term rank for the position that occupies an element after sorting A.

Selection

From 9.3 in CLRS

- We use the term rank for the position that occupies an element after sorting A.
- Selection Problem: Given an array A of n unordered distinct keys, and $i \in \{1, \ldots, n\}$, output the rank i element in A, that is the key that is larger than exactly $i - 1$ other keys in A.

Selection

From 9.3 in CLRS

- We use the term rank for the position that occupies an element after sorting A.
- Selection Problem: Given an array A of n unordered distinct keys, and $i \in \{1, \dots, n\}$, output the rank i element in A, that is the key that is larger than exactly $i - 1$ other keys in A.
- Notice that i can be any rank value between 1 and n, in particular when:
  1. $i = 1$, the MINIMUM element
  2. $i = n$, the MAXIMUM element
  3. $i = \lfloor \frac{n+1}{2} \rfloor$, the MEDIAN

A first algorithm

Sort A in $(O(n \lg n))$ steps, then the i-th smallest key is $A[i]$.

Can we do it faster? in linear time?

A first algorithm

Sort A in $(O(n \lg n))$ steps, then the i-th smallest key is $A[i]$.

Can we do it faster? in linear time?

Yes, selection is easier than sorting

The algorithm: High level

- Chose a split element x.
- Let k be the rank of x, if $k = i$, we found the i-th element. Otherwise,
- Use x to determine a partition of A, smaller than x to the left and larger to the right.
- Compute recursively the i-th element in the left part, when $i < k$, or the $i - k$-th element in the right part, when $i > k$.

The algorithm: High level

- Chose a split element x.
- Let k be the rank of x, if $k = i$, we found the i-th element. Otherwise,
- Use x to determine a partition of A, smaller than x to the left and larger to the right.
- Compute recursively the i-th element in the left part, when $i < k$, or the $i - k$-th element in the right part, when $i > k$.

The algorithm is correct, independently of the rule used to determine x, as x's rank is correctly computed.

The algorithm: High level

- Chose a split element x.
- Let k be the rank of x, if $k = i$, we found the i-th element. Otherwise,
- Use x to determine a partition of A, smaller than x to the left and larger to the right.
- Compute recursively the i-th element in the left part, when $i < k$, or the $i - k$-th element in the right part, when $i > k$.

The algorithm is correct, independently of the rule used to determine x, as x's rank is correctly computed.

The time depends on the quality of the splitting element to divide fairly the elements

Selection: Finding a splitting element

If $n \leq 5$ return their median.

Selection: Finding a splitting element

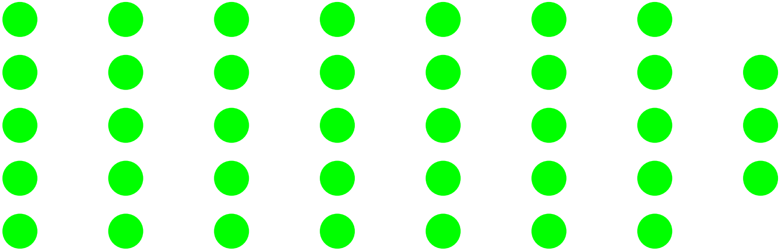
If $n \leq 5$ return their median.

Otherwise, divide the n elements in $\lceil n/5 \rceil$ groups, each with 5 elements except one group that might have < 5 elements).

Selection: Finding a splitting element

If $n \leq 5$ return their median.

Otherwise, divide the n elements in $\lceil n/5 \rceil$ groups, each with 5 elements except one group that might have < 5 elements).

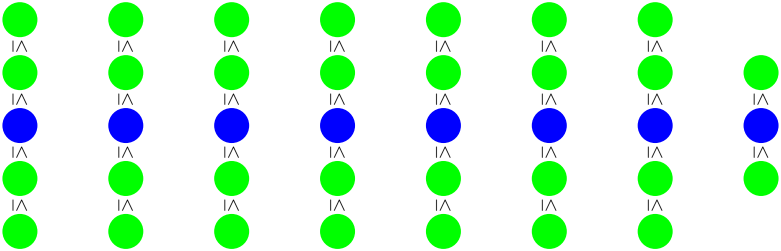


Selection: Finding a splitting element

Sort the elements in each group and find its median.
(Each sort needs ≤ 25 comparisons, i.e. $\Theta(1)$).
Call x_j the median of the j-th group.

Selection: Finding a splitting element

Sort the elements in each group and find its median.
(Each sort needs ≤ 25 comparisons, i.e. $\Theta(1)$).
Call x_j the median of the j-th group.



The splitting element x is the median of the set of medians, $\{x_j \mid 1 \leq j \leq \lceil n/5 \rceil\}$.

The algorithm for $n > 5$

Select(A, i)
Divide A into $m = \lceil n/5 \rceil$ groups, all but at most one with 5 elements
$X[j]$ = median of group j, $j = 1, \ldots, m$
$x =$ **Select**$(X, \lfloor (m+1)/2 \rfloor)$ i.e. the median of X
Let k be the rank of x in A
if $i = k$ **then**
  return x
else
  $L =$ the elements of A smaller than x (left)
  $R =$ the elements of A bigger than x (right)
  if $i < k$ **then**
    return **Select**(L, i)
  else
    return **Select**$(R, i - k)$

Example: Find the median

Let $n = 15$, we want to get the 5-th element on the following input:

$$A = \quad 3\ 13\ 9\ 4\ 5 \quad 1\ 15\ 12\ 10\ 2 \quad 6\ 14\ 8\ 17\ 11$$

An example

Let $n = 15$, we want to get the 5-th element on the following input:

$$A = \boxed{3\ 13\ 9\ 4\ 5}\ \boxed{1\ 15\ 12\ 10\ 2}\ \boxed{6\ 14\ 8\ 17\ 11}$$

An example

Let $n = 15$, we want to get the 5-th element on the following input:

$$A = \boxed{3\ 13\ 9\ 4\ 5}\ \boxed{1\ 15\ 12\ 10\ 2}\ \boxed{6\ 14\ 8\ 17\ 11}$$

3	1	6
4	2	8
5	10	11
9	12	14
13	15	17

An example

Let $n = 15$, we want to get the 5-th element on the following input:

$$A = \boxed{3\ 13\ 9\ 4\ 5}\boxed{1\ 15\ 12\ 10\ 2}\boxed{6\ 14\ 8\ 17\ 11}$$

3	1	6
4	2	8
5	10	11
9	12	14
13	15	17

The median of $X = (5, 10, 11)$ is 10 which has rank 9

An example

Let $n = 15$, we want to get the 5-th element on the following input:

$$A = \boxed{3\ 13\ 9\ 4\ 5}\ \boxed{1\ 15\ 12\ 10\ 2}\ \boxed{6\ 14\ 8\ 17\ 11}$$

3	1	6
4	2	8
5	10	11
9	12	14
13	15	17

The median of $X = (5, 10, 11)$ is 10 which has rank 9
As $5 < 9$, recursively ask for the 5-th element in the left part with respect to $x = 10$, i.e., $(3, 9, 4, 5, 1, 2, 6, 8)$

Example: Find the median

In the next call $n = 8$, we look for the 5-th element in the following input:

$$A = \;\; 3\;9\;4\;5\;1\;\;\;2\;6\;8$$

An example

In the next call $n = 8$, we look for the 5-th element in the following input:

$$A = \boxed{3\ 9\ 4\ 5\ 1}\boxed{2\ 6\ 8}$$

An example

In the next call $n = 8$, we look for the 5-th element in the following input:

$$A = \boxed{3\ 9\ 4\ 5\ 1}\boxed{2\ 6\ 8}$$

1	
3	2
4	6
5	8
9	

An example

In the next call $n = 8$, we look for the 5-th element in the following input:

$$A = \boxed{3\ 9\ 4\ 5\ 1}\boxed{2\ 6\ 8}$$

$$\begin{array}{|c|c|}\hline 1 & \\ 3 & 2 \\ \color{red}{4} & \color{red}{6} \\ 5 & 8 \\ 9 & \\ \hline\end{array}$$

The median of $X = (4, 6)$ is $\color{red}{4}$ which has rank 4.

An example

In the next call $n = 8$, we look for the 5-th element in the following input:

$$A = \boxed{3\ 9\ 4\ 5\ 1}\boxed{2\ 6\ 8}$$

1	
3	2
4	6
5	8
9	

The median of $X = (4, 6)$ is 4 which has rank 4.
As $5 > 4$ the algorithm looks for the 1st element in the right part $(5, 6, 8, 9)$, which is 5.

Selection algorithm: Cost

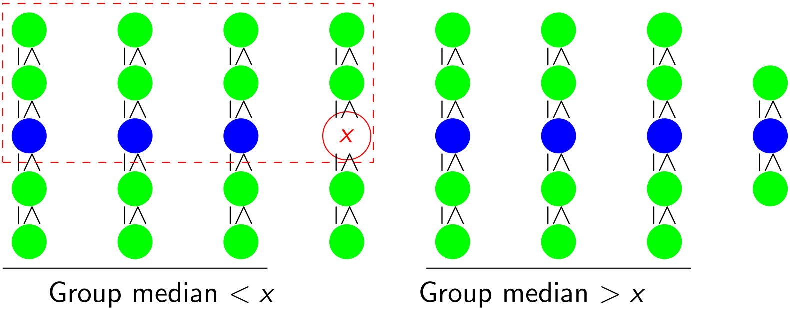
Select(A, i)
if $n \leq 5$ **then**
  Sort A and return the i-th element $O(1)$
Divide A into $m = \lceil n/5 \rceil$ groups, all but at most one with 5 elements $O(n)$
$X[j]$ = median of group j, $j = 1, \ldots, m$ $O(n)$
x = **Select**$(X, \lfloor (m+1)/2 \rfloor)$ i.e. the median of X $T(n/5)$
Let k be the rank of x in A
if $i = k$ **then**
  return x
else
  L = the elements of A smaller than x $O(n)$
  R = the elements of A bigger than x $O(n)$
  if $i < k$ **then**
    return **Select**(L, i) $T(?)$
  else
    return **Select**$(R, i-k)$ $T(?)$

Selection algorithm: elements smaller than x

At least $3\left(\left\lceil \frac{1}{2}\lceil n/5 \rceil \right\rceil - 2\right) \geq \frac{3n}{10} - 6$ of the elements are $< x$.

Selection algorithm: elements smaller than x

At least $3\left(\left\lceil\frac{1}{2}\lceil n/5\rceil\right\rceil - 2\right) \geq \frac{3n}{10} - 6$ of the elements are $< x$.

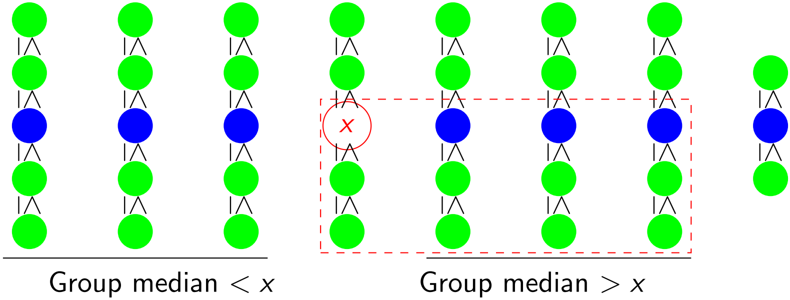


Selection algorithm: elements bigger than x

Al least $3\left(\left\lceil \frac{1}{2}\lceil n/5\rceil \right\rceil - 2\right) \geq \frac{3n}{10} - 6$ of the elements are $> x$.

Selection algorithm: elements bigger than x

Al least $3\left(\left\lceil\frac{1}{2}\lceil n/5\rceil\right\rceil - 2\right) \geq \frac{3n}{10} - 6$ of the elements are $> x$.



Selection algorithm: the recurrence

- As at least $\geq \frac{3n}{10} - 6$ of the elements are $< x$ $(> x)$, at most $n - (\frac{3n}{10} - 6) = 6 + 7n/10$ elements are $\leq x$ $(\geq x)$.

Selection algorithm: the recurrence

- As at least $\geq \frac{3n}{10} - 6$ of the elements are $< x$ ($> x$), at most $n - (\frac{3n}{10} - 6) = 6 + 7n/10$ elements are $\leq x$ ($\geq x$).
- In the worst case, **Select** recursively calls on a vector with size $\leq 6 + 7n/10$. So, step 5 takes time $\leq T(6 + 7n/10)$.

Selection algorithm: the recurrence

- As at least $\geq \frac{3n}{10} - 6$ of the elements are $< x$ ($> x$), at most $n - (\frac{3n}{10} - 6) = 6 + 7n/10$ elements are $\leq x$ ($\geq x$).
- In the worst case, **Select** recursively calls on a vector with size $\leq 6 + 7n/10$. So, step 5 takes time $\leq T(6 + 7n/10)$.

Selecting 140 (instead of 5) as the size to stop the recursion, we have

$$T(n) = \begin{cases} \Theta(1) & \text{if } n < 140, \\ T(\lceil n/5 \rceil) + T(6 + 7n/10) + \Theta(n) & \text{if } n \geq 140. \end{cases}$$

Selection algorithm: the recurrence

- As at least $\geq \frac{3n}{10} - 6$ of the elements are $< x$ ($> x$), at most $n - (\frac{3n}{10} - 6) = 6 + 7n/10$ elements are $\leq x$ ($\geq x$).
- In the worst case, **Select** recursively calls on a vector with size $\leq 6 + 7n/10$. So, step 5 takes time $\leq T(6 + 7n/10)$.

Selecting 140 (instead of 5) as the size to stop the recursion, we have

$$T(n) = \begin{cases} \Theta(1) & \text{if } n < 140, \\ T(\lceil n/5 \rceil) + T(6 + 7n/10) + \Theta(n) & \text{if } n \geq 140. \end{cases}$$

Solving the equation, we get $T(n) = \Theta(n)$.

Selection algorithm: the recurrence

- As at least $\geq \frac{3n}{10} - 6$ of the elements are $< x$ ($> x$), at most $n - (\frac{3n}{10} - 6) = 6 + 7n/10$ elements are $\leq x$ ($\geq x$).
- In the worst case, **Select** recursively calls on a vector with size $\leq 6 + 7n/10$. So, step 5 takes time $\leq T(6 + 7n/10)$.

Selecting 140 (instead of 5) as the size to stop the recursion, we have

$$T(n) = \begin{cases} \Theta(1) & \text{if } n < 140, \\ T(\lceil n/5 \rceil) + T(6 + 7n/10) + \Theta(n) & \text{if } n \geq 140. \end{cases}$$

Solving the equation, we get $T(n) = \Theta(n)$. How?

Solving the recurrence

- Use substitution. Note, $6 + 7n/10 < n$, for $n > 12$.

Solving the recurrence

- Use substitution. Note, $6 + 7n/10 < n$, for $n > 12$.
- Let c_1 be a constant, so that $T(n) \leq c_1 n$, for $n \leq 140$.

Solving the recurrence

- Use substitution. Note, $6 + 7n/10 < n$, for $n > 12$.
- Let c_1 be a constant, so that $T(n) \leq c_1 n$, for $n \leq 140$.
- Attempt to prove that $T(n) \leq cn$ by induction ($n \geq 140$), and find additional restrictions on c.
- We replace the $\Theta(n)$ term by its exact value dn, for some $d > 0$.

$$
\begin{aligned}
T(n) &\leq T(\lceil n/5 \rceil) + T(6 + 7n/10) + dn \\
&\leq c\lceil n/5 \rceil + c(6 + 7n/10) + dn \\
&\leq c(n/5 + 1) + c(6 + 7n/10) + dn \\
&= 9\frac{cn}{10} + 7c + dn \\
&= cn + (-\frac{cn}{10} + 7c + dn)
\end{aligned}
$$

Which is at most cn, if $-\frac{cn}{10} + 7c + dn \leq 0$.

Solving the recurrence

- We replace the $\Theta(n)$ term by its exact value dn, for some $d > 0$.

$$\begin{aligned} T(n) &\leq T(\lceil n/5 \rceil) + T(6 + 7n/10) + dn \\ &= cn + (-\frac{cn}{10} + 7c + dn) \end{aligned}$$

Which is at most cn, if $-\frac{cn}{10} + 7c + dn \leq 0$.

Solving the recurrence

- We replace the $\Theta(n)$ term by its exact value dn, for some $d > 0$.

$$T(n) \leq T(\lceil n/5 \rceil) + T(6 + 7n/10) + dn$$
$$= cn + (-\frac{cn}{10} + 7c + dn)$$

Which is at most cn, if $-\frac{cn}{10} + 7c + dn \leq 0$.

- As we assume that $n \geq 140$, choosing $c \geq 20d$ will satisfy the inequality.

Solving the recurrence

- We replace the $\Theta(n)$ term by its exact value dn, for some $d > 0$.

$$T(n) \leq T(\lceil n/5 \rceil) + T(6 + 7n/10) + dn$$
$$= cn + (-\frac{cn}{10} + 7c + dn)$$

Which is at most cn, if $-\frac{cn}{10} + 7c + dn \leq 0$.

- As we assume that $n \geq 140$, choosing $c \geq 20d$ will satisfy the inequality.
- By taking c so that $c \geq c_1$ and $c \geq 20d$, we have that the base case and the inductive step hold, so $T(n) = \Theta(n)$.

Remarks on the size of the groups

Notice:

- If we make groups of 7, the number of elements $\geq x$ is $\frac{2n}{7}$, which yield $T(n) \leq T(n/7) + T(5n/7) + \Theta(n)$ with solution $T(n) = \Theta(n)$.
- However, if we make groups of 3, then $T(n) \leq T(n/3) + T(2n/3) + \Theta(n)$, which has a solution $T(n) = \Theta(n \ln n)$.