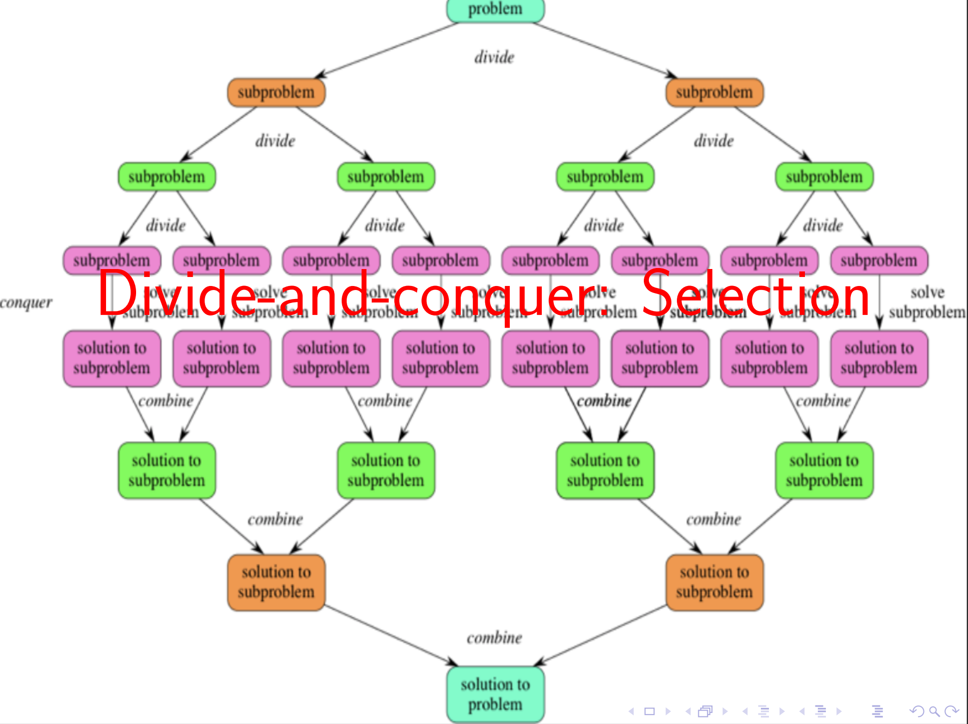
Divide-and-conquer: Selection



Selection

From 9.3 in CLRS

Problem: Given a list A of n of unordered distinct keys, and a $i \in \mathbb{Z}, 1 \leq i \leq n$, select the i-smallest element $x \in A$, that is x is larger than exactly $i - 1$ other elements in A.

Notice if:

1. $i = 1 \Rightarrow$ MINIMUM element
2. $i = n \Rightarrow$ MAXIMUM element
3. $i = \lfloor \frac{n+1}{2} \rfloor \Rightarrow$ the MEDIAN
4. $i = \lfloor 0.25 \cdot n \rfloor \Rightarrow$ *order statistics*

Non smart approach:
Sort A in $(O(n \lg n))$ steps and search for $A[k]$.
Can we do it in linear time?
Yes, selection is easier than sorting

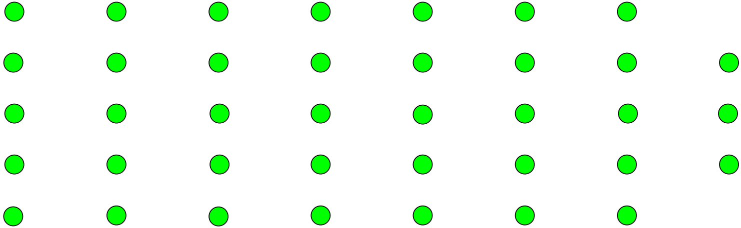
Deterministic linear selection: The algorithm

- Generate deterministically a good split element x.
- Use x to determine the partition (with respect to x) in which the i-th element lies.
- Apply the algorithm recursively to the selected part.

The position that has an element in the sorted list is often called its rank.

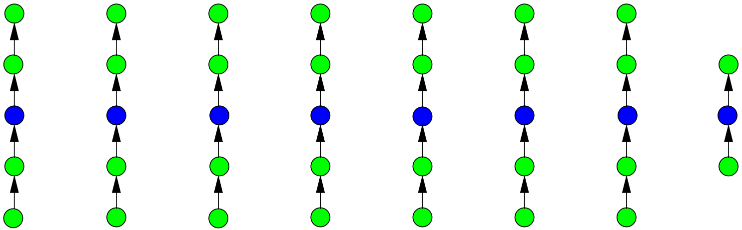
Deterministic linear selection: Finding a splitting element

If $n \leq 5$ return their median. Otherwise, Divide the n elements in $\lfloor n/5 \rfloor$ groups, each with 5 elements ($+$ possible one group with < 5 elements).



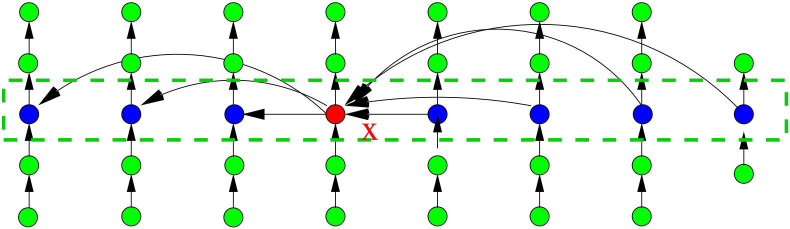
Deterministic linear selection. Finding a splitting element

Sort each set to find its median, say x_i. (Each sorting needs 5 comparisons, i.e. $\Theta(1)$) Total: $\lceil n/5 \rceil$



The splitting element will be the median x of the medians $\{x_i\}, 1 \leq i \leq \lceil n/5 \rceil$.

Deterministic linear selection: Selecting a part



Using as pivot x, the median of the medians x_i, partition the input array around x. If x is the k-th element of the array, after partitioning, there are $k-1$ elements on the low side of the partition and $n-k$ elements on the high side.

The deterministic algorithm

Select (A, i)

1.- Divide the n elements into $\lceil n/5 \rceil$ groups of 5 elements each with a possible group with < 5 elements

2.- Find the medians on each group by insertion sort.

3.- Use **Select** recursively to find the median x of the $\lfloor n/5 \rfloor$ medians

4.- Partition the elements of A around x.
Smaller to the left and larger to the right.
Let k be the rank of x

5.- **if** $i = k$ **then**
  return x
else if $i < k$ **then**
  use **Select** to find the i-th smallest in the left
else
  use **Select** to find the $i - k$-th smallest in the right
end if

Example: Find the median

Get the median ($\lfloor (n+1)/2 \rfloor$) on the following input:

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

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

PARTITION around 10:

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

To get the 8th element (median)

call SELECT on 3 4 5 9 1 2 6 8 to get 3 1 2 4 5 9 6 8

and iterate until getting the median

The deterministic algorithm: Cost

Select (A, i)

1.- Divide the n elements into $\lceil n/5 \rceil$ groups of 5 $O(n)$
 with a possible group with < 5 elements

2.- Find the median by insertion sort, and take
 the middle element $O(n)$

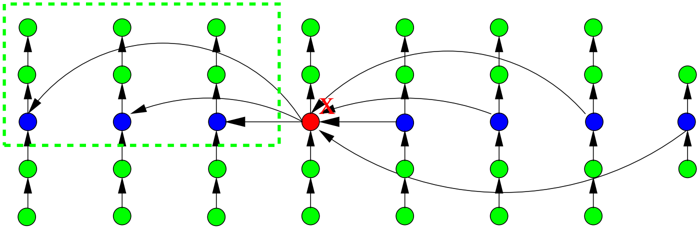
3.- Use **Select** recursively to find the median x of the $\lceil n/5 \rceil$
 medians $T(n/5)$

4.- Partition the elements of A around x. $O(n)$
 Smaller to the left and larger to the right.
 Let k be the rank of x

5.- **if** $i = k$ **then**
  return x
 else if $i < k$ **then**
  use **Select** to find the i-th smallest in the left $T(?)$
 else
  use **Select** to find the $i - k$-th smallest in the right $T(?)$
 end if

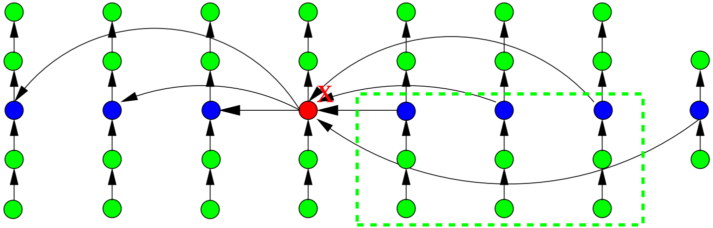
Analysis: Deterministic linear selection.

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



Analysis: Deterministic linear selection.

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



Worst case Analysis.

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

Therefore, selecting 50 as the size to stop the recursion, we have

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 50, \\ T(\lceil n/5 \rceil) + T(6 + 7n/10) + \Theta(n) & \text{if } n > 50. \end{cases}$$

Solving we get $T(n) = \Theta(n)$

Worst case Analysis.

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

Therefore, selecting 50 as the size to stop the recursion, we have

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 50, \\ T(\lceil n/5 \rceil) + T(6 + 7n/10) + \Theta(n) & \text{if } n > 50. \end{cases}$$

Solving we get $T(n) = \Theta(n)$ How?

Solving the recurrence

- Use substitution.

Solving the recurrence

- Use substitution.
- Assume that $T(n) \leq cn$ for some constant c and $n \leq 50$. Note that $6 + 7n/10 < n$, for $n > 20$.

Solving the recurrence

- Use substitution.
- Assume that $T(n) \leq cn$ for some constant c and $n \leq 50$. Note that $6 + 7n/10 < n$, for $n > 20$.
- Prove that $T(n) \leq cn$ always by induction.

$$\begin{aligned} T(n) &\leq T(\lceil n/5 \rceil) + T(6 + 7n/10) + \Theta(n) \\ &\leq c \; n/5 + c + c(6 + 7n/10) + \Theta(n) \\ &\leq 9cn/10 + 7c + \Theta(n) \leq cn \end{aligned}$$

We can select c larger than the constant in the term $\Theta(n)$.

Remarks on the cardinality of the groups

Notice:

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