Divide-and-conquer: Order Statistics

Curs: 2018

The divide-and-conquer strategy.

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

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

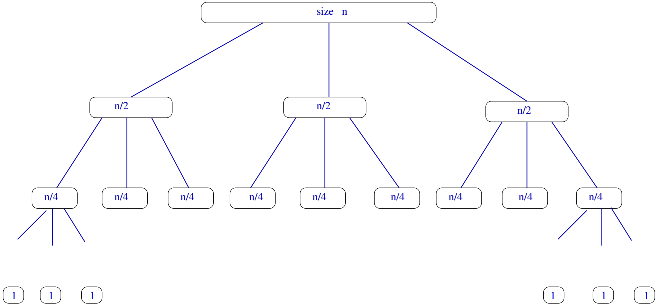
Known Examples:

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

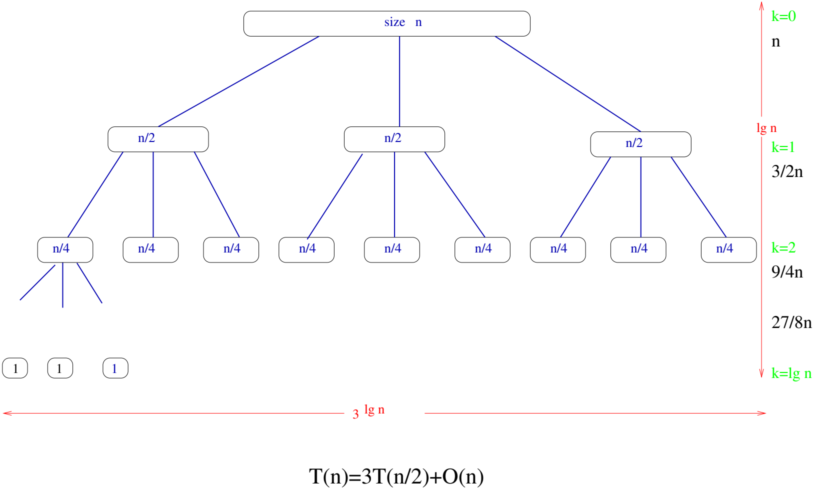
J. von Neumann (1903-57)
Merge sort

Recurrences Divide and Conquer

$T(n) = 3T(n/2) + O(n)$
The algorithm under analysis divides input of size n into 3 subproblems, each of size $n/2$, at a cost (of dividing and joining the solutions) of $O(n)$



$$T(n) = 3T(n/2) + O(n).$$



size n
n/2
n/2
n/2
n/4
n/4
n/4
n/4
n/4
n/4
n/4
n/4
n/4
1
1
1
k=0
n
lg n
k=1
3/2n
k=2
9/4n
27/8n
k=lg n
3 lg n
T(n)=3T(n/2)+O(n)

At depth k of the tree there are 3^k subproblems, each of size $n/2^k$.

For each of those problems we need $O(n/2^k)$ (splitting time + combination time).
Therefore the cost at depth k is:

$$3^k \times \left(\frac{n}{2^k}\right) = \left(\frac{3}{2}\right)^k \times O(n).$$

with max. depth $k = \lg n$.

$$\left(1 + \frac{3}{2} + (\frac{3}{2})^2 + (\frac{3}{2})^3 + \cdots + (\frac{3}{2})^{\lg n}\right) \Theta(n)$$

Therefore $T(n) = \sum_{k=0}^{\lg n} O(n)(\frac{3}{2})^k$.

From $T(n) = O(n)\left(\underbrace{\sum_{k=0}^{\lg n}(\frac{3}{2})^k}_{(*)}\right)$,

We have a geometric series of ratio $3/2$, starting at 1 and ending at $((\frac{3}{2})^{\lg n}) = \frac{n^{\lg 3}}{n^{\lg 2}}) = \frac{n^{1.58}}{n}$.

As the series is increasing, $T(n)$ is dominated by the last term:

$$T(n) = O(n) \times \left(\frac{n^{\lg 3}}{n}\right) = O(n^{1.58}).$$

$T(n) = T(n/4) + T(n/2) + n^2$

Series of costs: $(1 + ((\frac{1}{4})^2 + (\frac{1}{2})^2) + ((\frac{1}{16})^2 + (\frac{1}{8})^2) + \cdots)n^2$
$= (1 + \frac{5}{16} + \frac{25}{256} + \cdots)n^2$
Decreasing geometric series dominated by 1st. term, n^2.

$$T(n) = O(n^2)$$

General setting: Basic Theorem

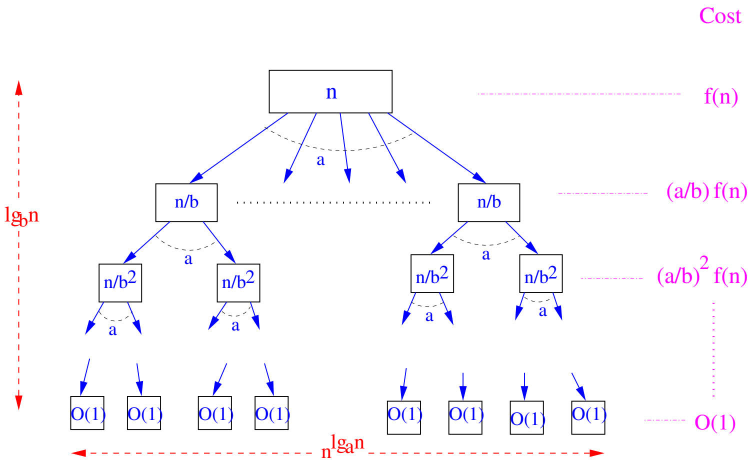
$$T(n) = aT(\frac{n}{b}) + f(n),$$

where n: size of the problem,
n/b: size of the subproblems
$f(n)$: cost of divide the problem and combine the solutions

Theorem

Let $a \geq 1, b > 1, d \geq 0$ be constants. The recurrence $T(n) = aT(n/b) + O(n^d)$ has asymptotic solution:

1. $T(n) = O(n^d)$, *if* $d > \log_b a$,
2. $T(n) = O(n^d \lg n)$ *if* $d = \log_b a$,
3. $T(n) = O(n^{\log_b a})$, *if* $d < \log_b a$.


Cost
n
f(n)
a
n/b
n/b
(a/b) f(n)
lgbn
n/b2
n/b2
n/b2
n/b2
(a/b)2 f(n)
a
a
a
a
a
a
O(1)
O(1)
O(1)
O(1)
O(1)
O(1)
O(1)
O(1)
O(1)
nlgan

Selection and order statistics

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 larger than exactly $i-1$ other elements in A.

Notice if:

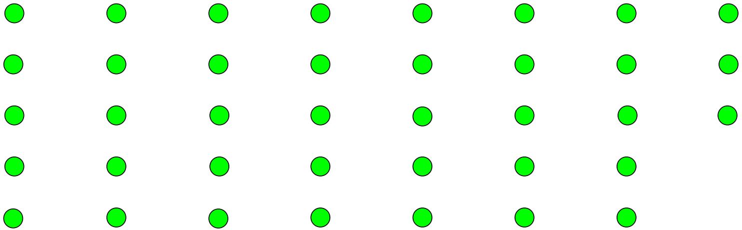
1. $i = 1 \Rightarrow$ MINIMUM element
2. $i = n \Rightarrow$ MAXIMUM element
3. $i = \lfloor \frac{n+1}{2} \rfloor \Rightarrow$ the MEDIAN
4. $i = \lfloor 0.9 \cdot n \rfloor \Rightarrow$ *order statistics*

Sort A ($O(n \lg n)$) and search for $A[k]$.

Can we do it in linear time?
Yes, Selection is easier than Sorting

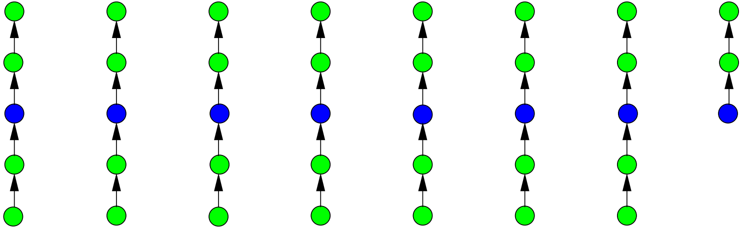
Deterministic linear selection

Generate deterministically a good split element x.
Divide the n elements in $\lfloor n/5 \rfloor$ groups, each with 5 elements ($+$ possible one group with < 5 elements).



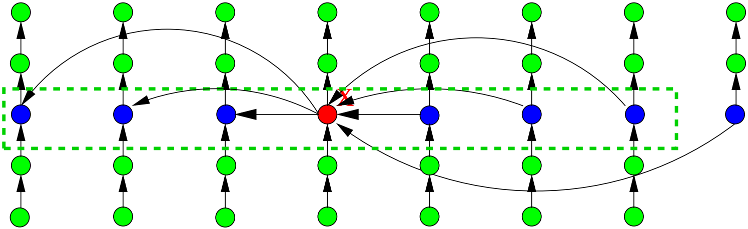
Deterministic linear selection.

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



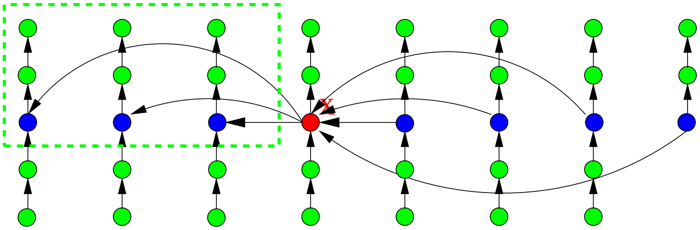
Deterministic linear selection.

• Use recursively **Select** to find the median x of the medians $\{x_i\}, 1 \leq i \leq \lceil n/5 \rceil$.
• Use deterministic **Partition** (quick sort) to re-arrange the groups corresponding to medians $\{x_i\}$ around x, in linear time on the number of medians.



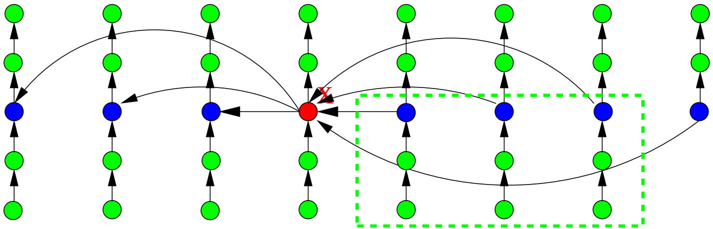
Deterministic linear selection.

Al least $3(\frac{1}{2}\lfloor n/5 \rfloor) = \lfloor 3n/10 \rfloor$ of the elements are $\leq x$.



Deterministic linear selection.

Al least $3(\frac{1}{2}\lfloor n/5 \rfloor) = \lfloor 3n/10 \rfloor$ of the elements are $\geq x$.



The deterministic algorithm

Select (A, i)
1.- Divide the n elements into $\lfloor n/5 \rfloor$ groups of 5 $O(n)$
 plus a possible extra group with < 5 elements
2.- Find the median by insertion sort, and take
 the middle element
3.- Use **Select** recursively to find the median x of the $\lfloor n/5 \rfloor$
 medians
4.- Use **Partition** to place x and its group. Let k=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 7th element (mean)

call SELECT on this instance

The deterministic algorithm

Select (A, i)
1.- Divide the n elements into $\lfloor n/5 \rfloor$ groups of 5 $O(n)$
  plus a possible extra 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 $\lfloor n/5 \rfloor$
  medians $T(n/5)$
4.- Use **Partition** to place x and its group. $O(n)$
  Let $k =$ 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

Worst case Analysis.

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

Therefore, we have

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

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

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 $T(n) = O(n \ln n)$.