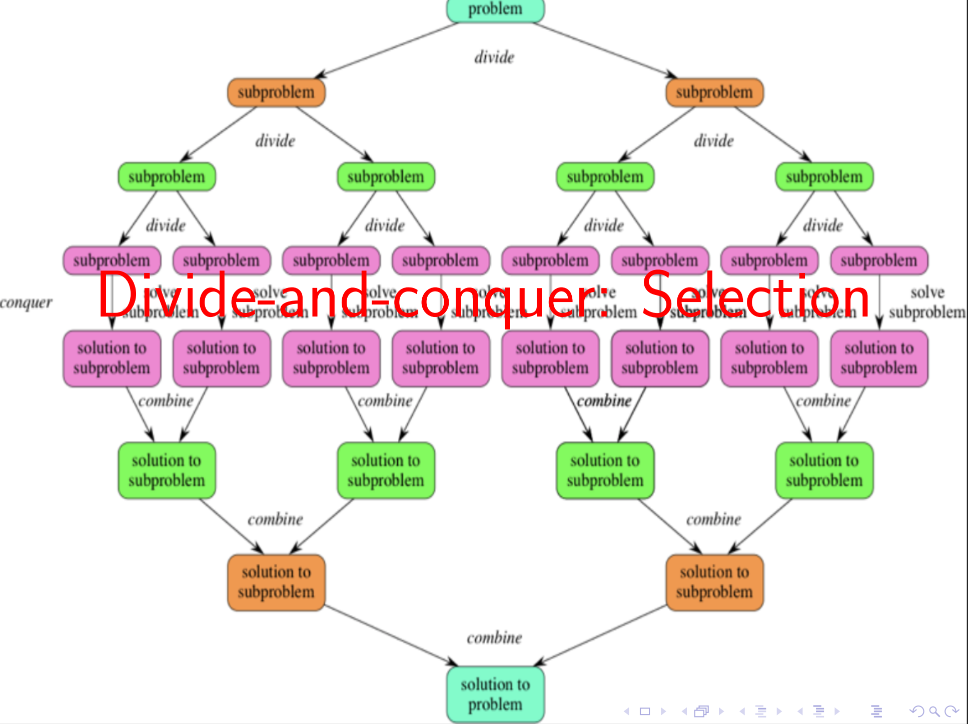
Divide-and-conquer: Selection



Some math. you should remember

Given an integer $n > 0$ and a real $a > 1$ and $a \neq 0$:

- Arithmetic summation: $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$.
- Geometric summation: $\sum_{i=0}^{n} a^i = \frac{1-a^{n+1}}{1-a}$.

Logarithms and Exponents: For $a, b, c \in \mathbb{R}^+$,

- $\log_b a = c \Leftrightarrow a = b^c \Rightarrow \log_b 1 = 0$
- $\log_b ac = \log_b a + \log_b c$, $\log_b a/c = \log_b a - \log_b c$.
- $\log_b a^c = c \log_b a \Rightarrow c^{\log_b a} = a^{\log_b c} \Rightarrow 2^{\log_2 n} = n$.
- $\log_b a = \log_c a / \log_c b \Rightarrow \log_b a = \Theta(\log_c a)$

Stirling: $n! = \sqrt{2\pi n}(n/e)^n + 0(1/n) + \gamma$.

n-Harmonic: $H_n = \sum_{i=1}^{n} 1/i \sim \ln n$.

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"

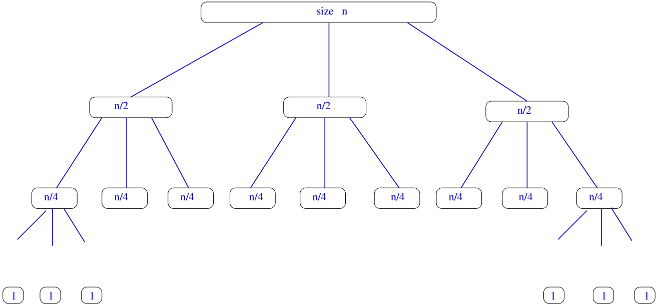
Known Examples:

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

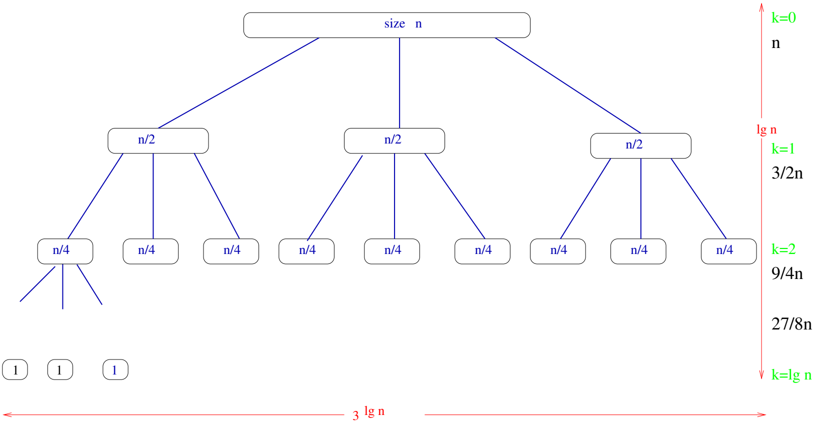
J. von Neumann (1903-57)
Merge sort

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 $O(n)$



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



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 $((\frac{3}{2})^{\lg n}) = \frac{n^{\lg 3}}{n^{\lg 2}} = \frac{n^{1.58}}{n} = n^{0.58}$.

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

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

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

Notice the work at all leaves is equal to the number of leaves, which is $2^h = 2^{\lg n} = n$.

So $T(n) = n + n^2\left(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots\right) = n + n^2 \sum_{i=0}^{(\lg n)-1} (\frac{1}{2})^i$.

The sum is computed using the geometric series:

$$\sum_{i=0}^{h} x^i = \frac{x^{h+1} - 1}{x - 1}.$$

To get $T(n) = 2n^2 - 2n + n = 2n^2 - n$.

Using Mathematica (Mapple):

RSolve[{T[n]==2T[n/2] + n^2, T[1]==1}, T[n], n]

{T[n] $\rightarrow$ $n(-1 + 2n)$}

The 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. A different one can be found in CLRS's book Theorem 4.1

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:*

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

The basic M.T. leave many cases outside. For stronger MT:

Akra-Bazi Theorem: `https://courses.csail.mit.edu/6.046/spring04/handouts/akrabazzi.pdf`

Salvador Roura Theorems `http://www.lsi.upc.edu/~diaz/RouraMT.pdf`

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

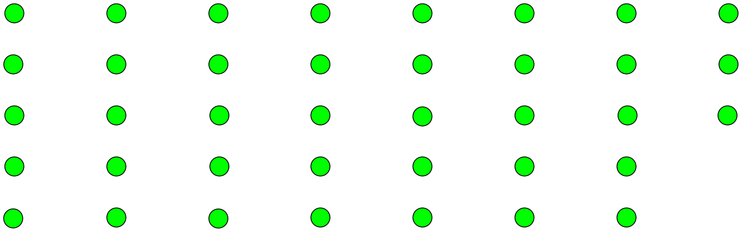
Notice if:

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 more easy than sorting

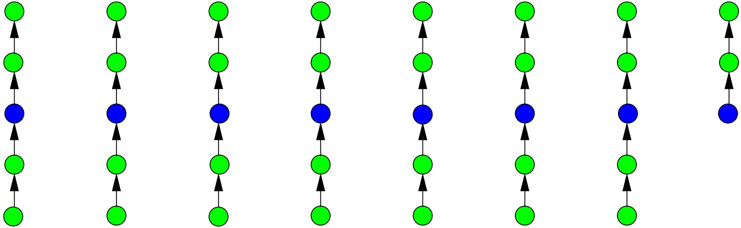
Deterministic linear selection

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.

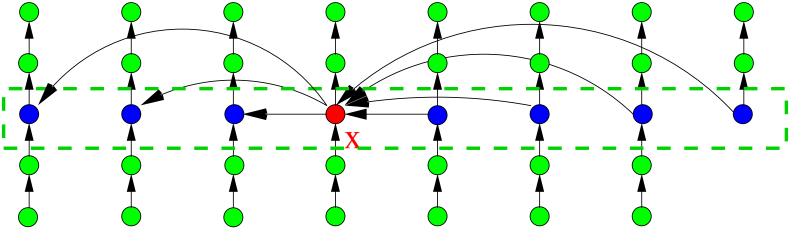
Sort each set to find its median, say x_i. (Each sorting needs 5 comparisons, i.e. $\Theta(1)$) Total: $\lceil n/5 \rceil$



Deterministic linear selection.

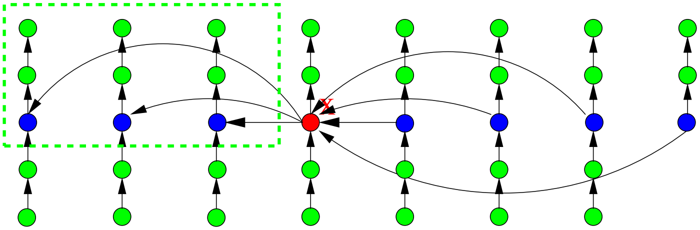
• Use recursively **Select** to find the median x of the medians $\{x_i\}, 1 \leq i \leq \lceil n/5 \rceil$.

• Using **Partition** function taking as pivot the median of the medians x_i, partition the input array around x_i. Let x_i be the k-th element of the array after partitioning, so that there are $k-1$ elements on the low side of the partition and $n-k$ elements on the high side.



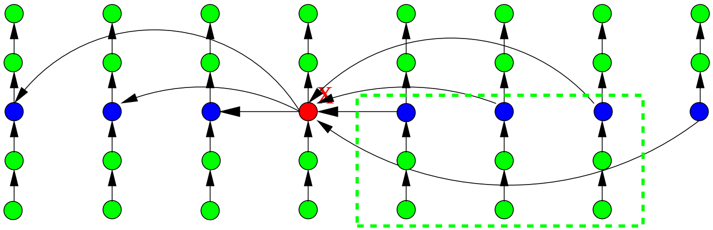
Deterministic linear selection.

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** the elements under consideration around x.
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 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

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** around x. $O(n)$
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 $T(n) = \Theta(n)$

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

Arbitrary i-order statistics

Given $A[1, \dots, n]$ we can use the median algorithm as a black-box algorithm to solve the ith. order statistics o A, i.e. finding the i-smaller element in A.

1. Find the median m.
2. Partition the array based on that median:
   2.1 If i is less than half the length of the original array, recurse on the first half,
   2.2 if i is exactly half the length of the array, return the founded median.