Selection problem

Given a set of “n” numbers we can say that,

- ✓ ***Mean:*** Average of the “n” numbers
- ✓ ***Median:*** Having sorted the “n” numbers, the value which lies in the middle of the list such that half the numbers are higher than it and half the numbers are lower than it

  (if n is even, can declare (n-1)/2 or (n+1)/2 lowest number as median)

k-Selection problem: given a list of n numbers, find the k^{th} smallest number

Median is when k = n/2

To simplify things, we assume throughout that numbers are distinct

Selection problem

Exercise: Design algorithms that solve the k-selection problem in time:

a) O(kn)

b) O(n log n)

c) O(n + k log n)

A:

a) Do k linear passes, each time removing the smallest element; return the last one you removed

b) Sort the list and look at the kth position

c) Build a Heap with the numbers (time O(n)), remove the minimum element k times (O(log n) per removal), return the last removed item

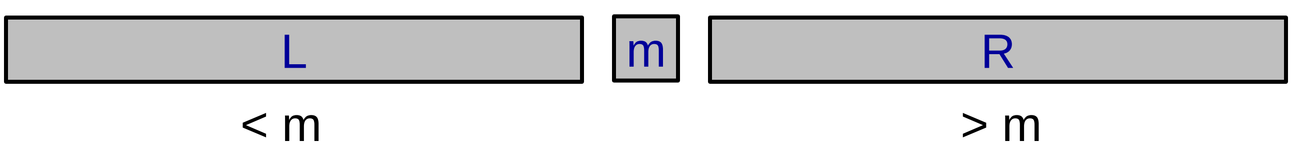
We will see how to do this in time O(n)

Selection in linear time

We will first cheat: we assume we know we can solve the median problem for free

ExotericSelect(A,k)

- Ask the oracle for the median m of A
- Partition numbers around the median so that values less than m are in set L and values greater than m are in set R
- If |L|=k-1 then
- If |L|>k-1 then
- Else



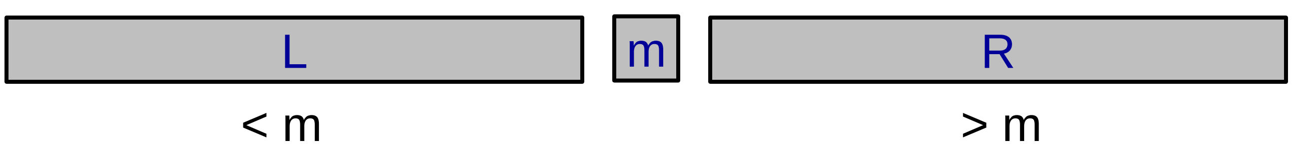
Obs: We did not use so far that m is the median, we will only need this in the analysis of running time

Selection in linear time

We will first cheat: we assume we know we can solve the median problem for free

ExotericSelect(A,k)

- Ask the oracle for the median m of A
- Partition numbers around the median so that values less than m are in set L and values greater than m are in set R
- If |L|=k-1 then return m
- If |L|>k-1 then ExotericSelect(L,k)
- Else ExotericSelect(R, k-|L|-1)



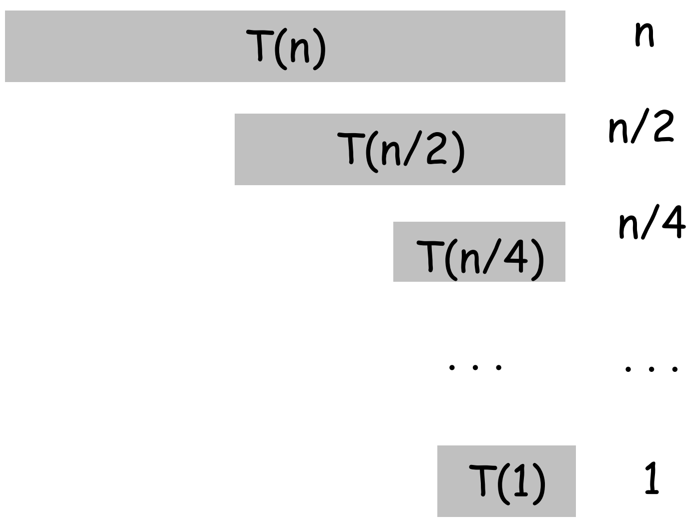
Obs: We did not use so far that m is the median, we will only need this in the analysis of running time

Selection in linear time: ExotericSelect Analysis

The recurrence equation of the algorithm is

$$T(1) = 1$$
$$T(n) = n + T(n/2) \quad \text{if } n>1$$

Q: What is the solution to this recurrence?



Total: $n + n/2 + n/4 + \ldots + 1$

$= n\,(1 + \frac{1}{2} + \frac{1}{4} + \ldots + 1/n) <= 2n$

↑ Geometric series

So the ExotericSelect algorithm is O(n)

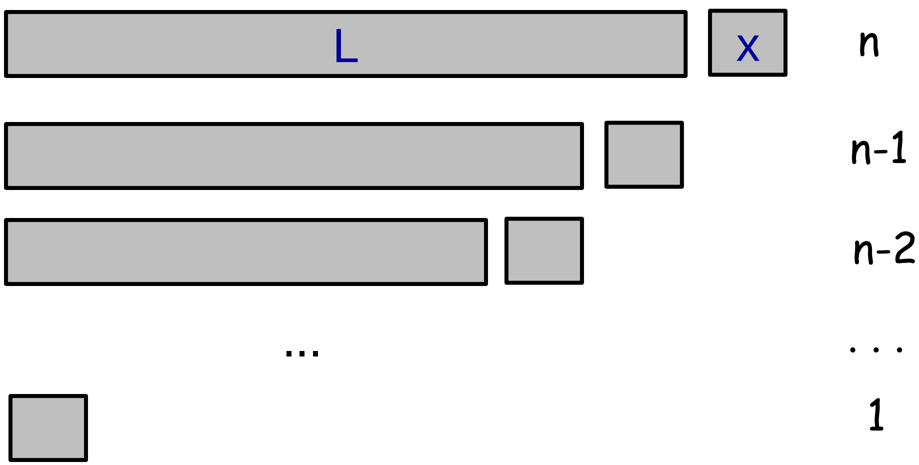
Selection in linear time

But ExotericSelect cheated by assuming we have the median for free

We cannot yet compute median, need to use something simpler

Q: What is the problem of using any number x to split our list into < x and > x?

A: Running time can be $\Omega(n^2)$ if x is always the largest element (and k=1)



Total: n + (n-1) + ... + 1 = $\Omega(n^2)$

↑ Arithmetic series

Idea: Use a relaxed version of median spliting around a value x such that there are at least 3n/10 numbers smaller and at least 3n/10 larger

Selection in linear time

Median of medians algorithm (Blum, Floyd, Pratt, Rivest, Tarjan ‘73)

MOM_algo(A,k)

Replaces median(A) oracle

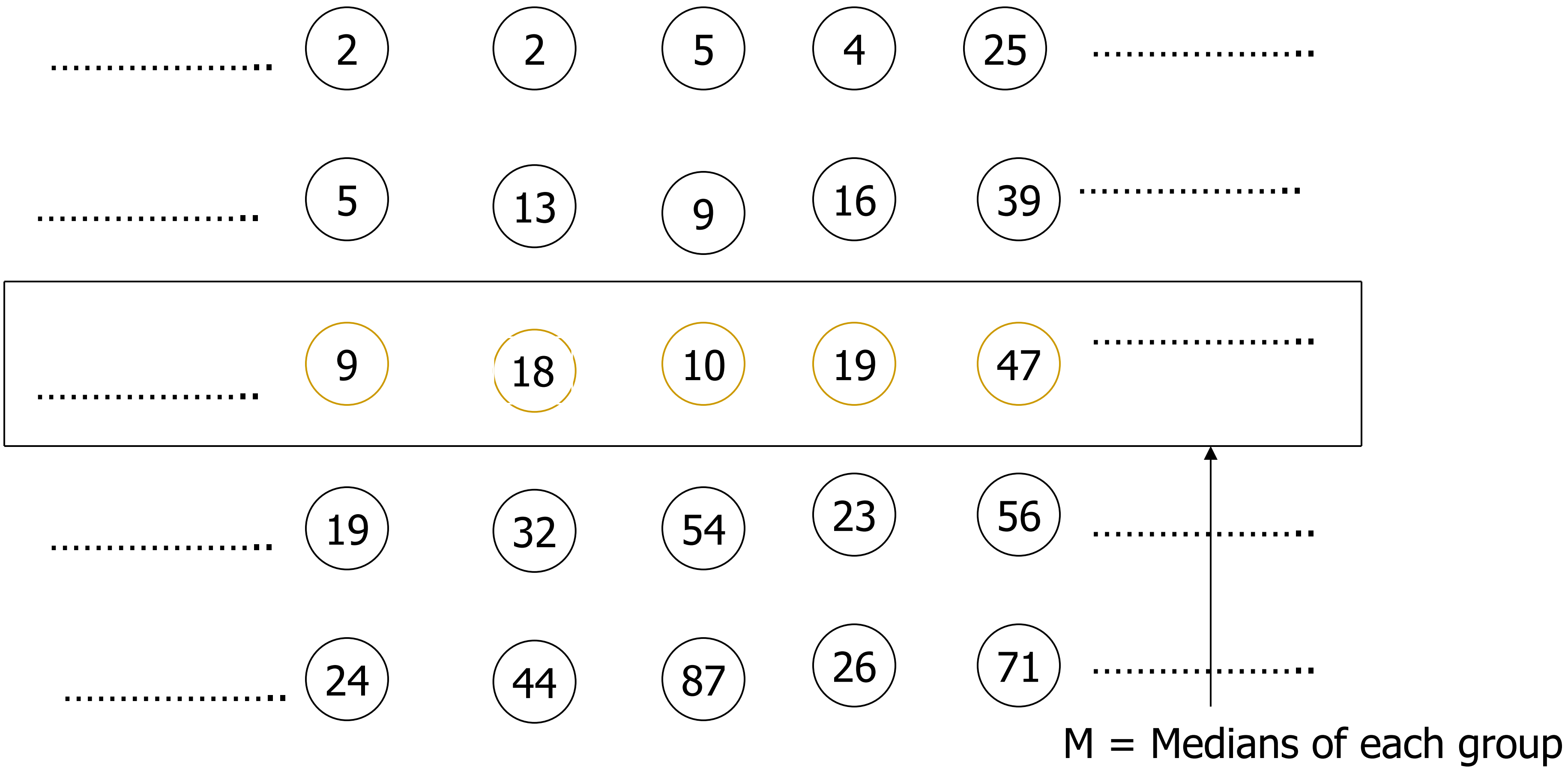
1. Group the numbers into sets of 5
2. Sort individual groups and find the median of each group; put these medians in a set M
3. Find median m’ of set M using MOM_algo(M,|M|/2)
4. Partition original data around m’ such that values less than it are in set L and values greater than it are in set R
5. If |L| = k-1, then return m’
   If |L| > k-1, then return MOM_algo(L,k)
   If |L|< k-1 then return MOM_algo(R,k-|L|-1)

Ex: (2,5,9,19,24,54,5,87,9,10,44,32,18,13,2,4,23,26,16,17,25,39,47, 56,71)

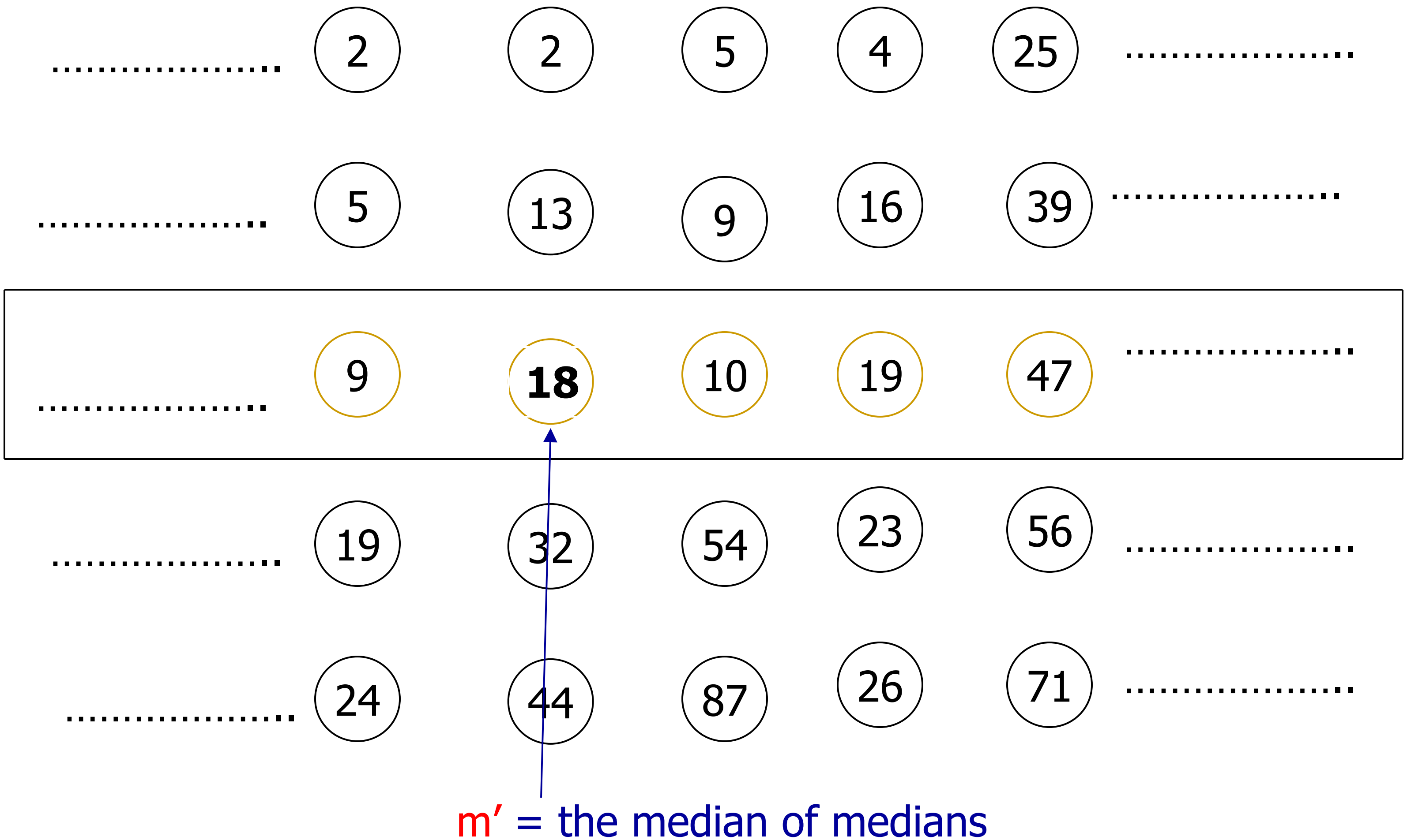
Step1: Group numbers in sets of 5 (Vertically)

……………….	2	44	54	4	25	……………….
……………….	5	32	5	23	39	……………….
……………….	9	18	87	26	47	……………….
……………….	19	13	9	16	56	……………….
……………….	24	2	10	19	71	……………….

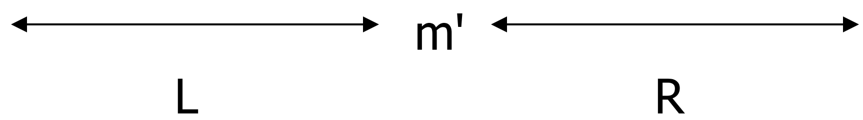
Step2: Sort each group, find median of each group



Step3: Find the median of medians



Step4: Partition original data around the median-of-medians



Step5: If |L| = k-1, then return MOM else

If |L| > k-1, then return kth_smallest (L,k) else

If |L| < k-1 then return kth_smallest (R,k-(|L|+1))

Selection in linear time

Median of medians algorithm (Blum, Floyd, Pratt, Rivest, Tarjan '73)

MOM_algo(A,k)

Replaces median(A) oracle

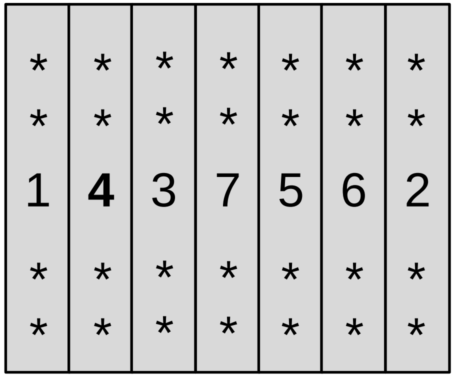
1. Group the numbers into sets of 5
2. Sort individual groups and find the median of each group; put these medians in a set M
3. Find median m' of set M using MOM_algo(M,|M|/2)
4. Partition original data around m' such that values less than it are in set L and values greater than it are in set R
5. If |L| = k-1, then return m'
   If |L| > k-1, then return MOM_algo(L,k)
   If |L|< k-1 then return MOM_algo(R,k-|L|-1)

This is now ok because we want median of a smaller set of numbers, can use induction

Selection in linear time: Analysis

We show that the median-of-medians m’ gives a balanced partition

Lemma: The set L has at least 3n/10 elements. Similarly |R| >= 3n/10



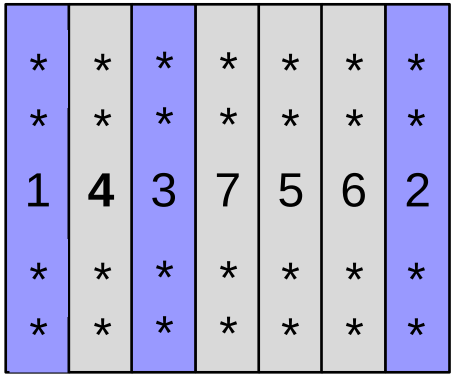
Selection in linear time: Analysis

We show that the median-of-medians m’ gives a balanced partition

Lemma: The set L has at least 3n/10 elements. Similarly |R| >= 3n/10

Proof:

- groups have median smaller than m’, so groups



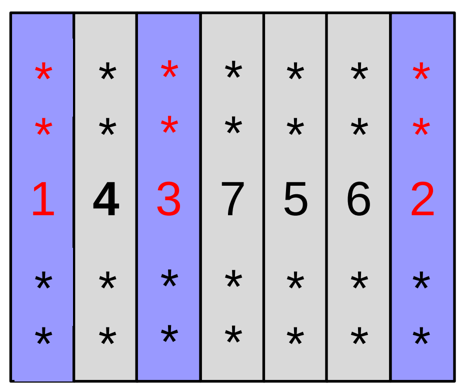
Selection in linear time: Analysis

We show that the median-of-medians m’ gives a balanced partition

Lemma: The set L has at least 3n/10 elements. Similarly |R| >= 3n/10

Proof:

- Half of the groups have median smaller than m’, so (n/5)/2 = n/10 groups
- Each such group has at least elements smaller than m’



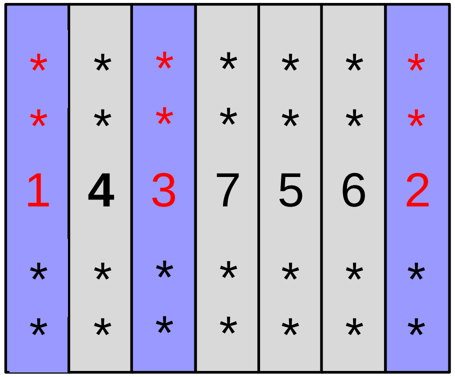
Selection in linear time: Analysis

We show that the median-of-medians m’ gives a balanced partition

Lemma: The set L has at least 3n/10 elements. Similarly |R| >= 3n/10

Proof:

- Half of the groups have median smaller than m’, so (n/5)/2 = n/10 groups
- Each such group has at least three elements smaller than m’
- So there is a total of (n/10)*3 = 3n/10 elements smaller than m’
=> L has at least 3n/10 elements



Corollary: |L| <= 7n/10 and |R| <= 7n/10

Selection in linear time: Analysis

Time Analysis:

Step	***Task***	***Complexity***
1	Group into sets of 5	O (n)
2	Find Median of each group	O (n)
3	Find med-of-med m’	T(n/5)
4	Partition around m’	O (n)
5	Recurse on L or R	<= T(7n/10)

Recurrence relation:

$$T(1) = 1$$

$$T(n) <= cst*n + T(n/5) + T(7n/10) \quad \text{for } n > 1$$

By induction, we show that T(n) <= 10*cst*n:

- T(n) <= cst*n + 10*cst*(n/5) + 10*cst*(7n/10) = 10*cst*n

So the algorithm is O(n)

Selection in linear time

Natural Questions

- Can we split the list into groups of 3 elements instead of 5?
- Can we split the list into groups of 7 elements instead of 5?

Selection in linear time

Natural Questions

- Can we split the list into groups of 3 elements instead of 5?
  - $T(n)=cn+T(n/3)+T(2n/3)$. This recurrence does not have a linear solution

- Can we split the list into groups of 7 elements instead of 5?
  - $T(n)=cn+T(n/7)+T(5n/7)$. Linear with a different constant

QuickSelect

Linear time algorithm has a large constant

- How to proceed in practice?

Just like in quicksort, all we want is an element (pivot) that partitions input list in balanced way

QuickSelect(A,k)

- Select a Random element p from the list
- Partition original data around the pivot p such that values less than it are in set L and values greater than it are in set R
- If |L|=k-1 then return p
- If |L|>k-1 then QuickSelect(L,k)
- Else QuickSelect(R, k-|L|-1)

QuickSelect

Analysis

- $T(n)$: expected time to select the k-th smallest element from a list of n numbers

- Pivot is the i-th smallest element with probability 1/n.
  - If the pivot is the i-th smallest we have to solve a problem of size (n-i) if i<k and of size (i-1) if i>k. If i=k the algorithm stops.

- $T(n) = cn + 1/n\,[T(k)+T(k+1)+\ldots+T(n-1)+T(n-k-1)+\ldots+T(n-1)]$
- $T(1)=1$

- We can prove by induction that $T(n) <= 4cn$

Exercise

Exercise: Use the linear time algorithm for finding the median to design a version of Quicksort that runs in time O(n log n) (in the worst case)