Selection in Linear Time

- 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.
- *Thus the problem of finding the median can be generalized to finding the k^{th} smallest number where k = n/2.*

Selection in Linear Time

- k^{th} smalles number can be found using:
- ✓ Scan Approach with a time complexity $T(n) = kn$
- ✓ Sort Approach with a time complexity $T(n) = n\log n$

Selection in Linear Time

- ***Input:*** S = {a1,a2,……..,an}
- k such that $1 \leq k \leq n$
- ***Output:*** k^{th} smallest number
- ***Algorithm:*** k^{th}_smallest (S, k)

Selection in Linear Time

- ExotericSelect(A,k)
  - Ask the oracle for the median
  - Partition original data around the median 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 Median
  - If |L|>k-1 then ExotericSelect(L,k)
  - Else ExotericSelect(R, k-|L|-1)

Selection in Linear Time

- ExotericSelect Analysis
  - T(n)=n+T(n/2) if n>1; T(1) =1
  - T(n) = O(n)

Can we implement the oracle function at least approximately?

Selection in Linear Time

Kth_SMALLEST(S,k)

- ***Steps:***

1) Group the numbers into sets of 5

2) Sort individual groups and find the median of each group

3) Let “M” be set of medians and find median of “M” using MedianOfMedian (MOM) = k^{th}_smallest (M,|M|/2)

4) Partition original data around the MOM 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 MOM else

   If |L| > k-1, then return k^{th}_smallest(L,k) else

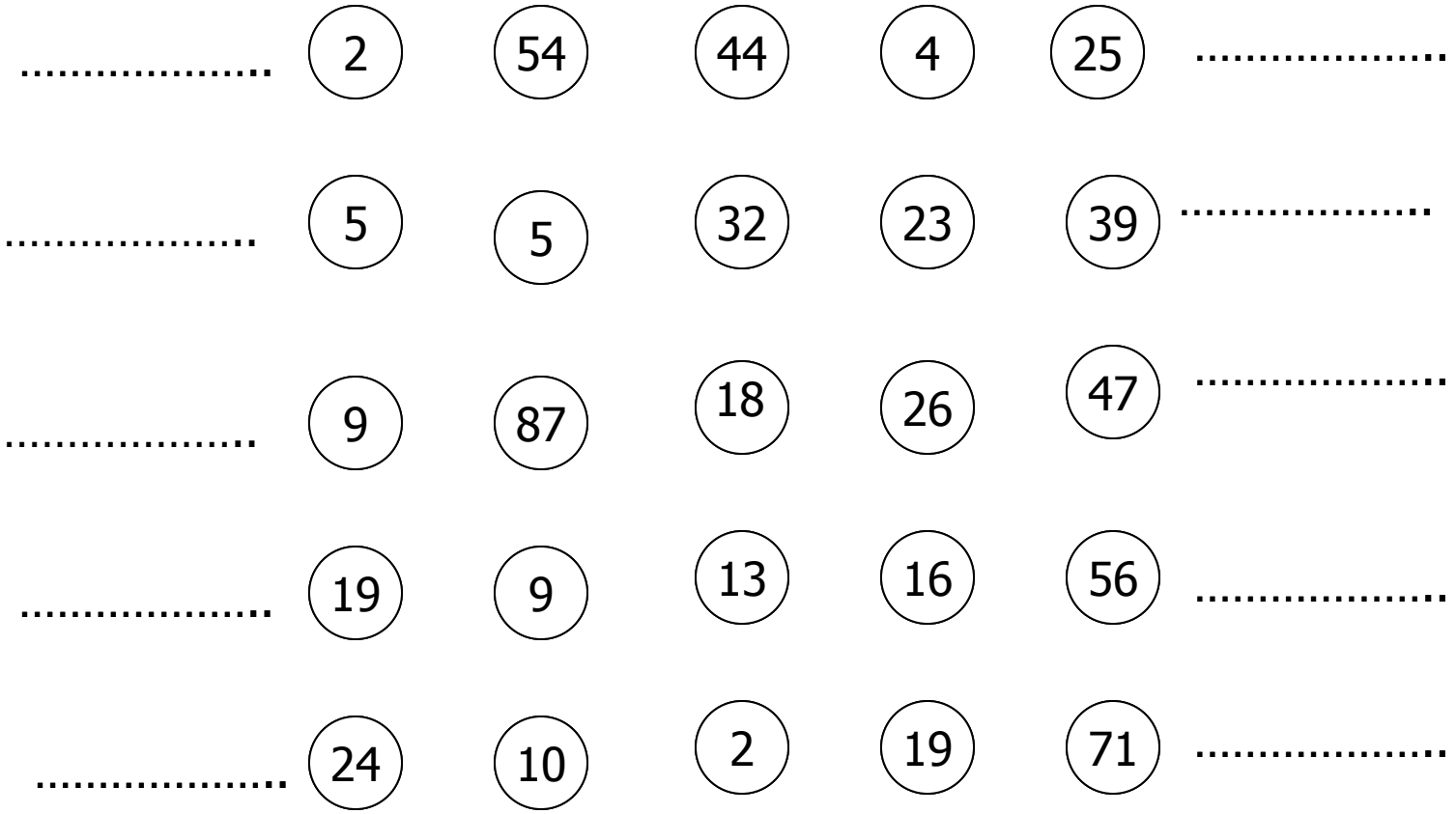
- return k^{th}_smallest(R,k-|L|-1)

Selection in Linear Time

- ***Example:***
- (2,5,9,19,24,54,5,87,9,10,44,32,18,13,2,4,23, 26,16,19,25,39,47,56,71) is a set of “n” numbers

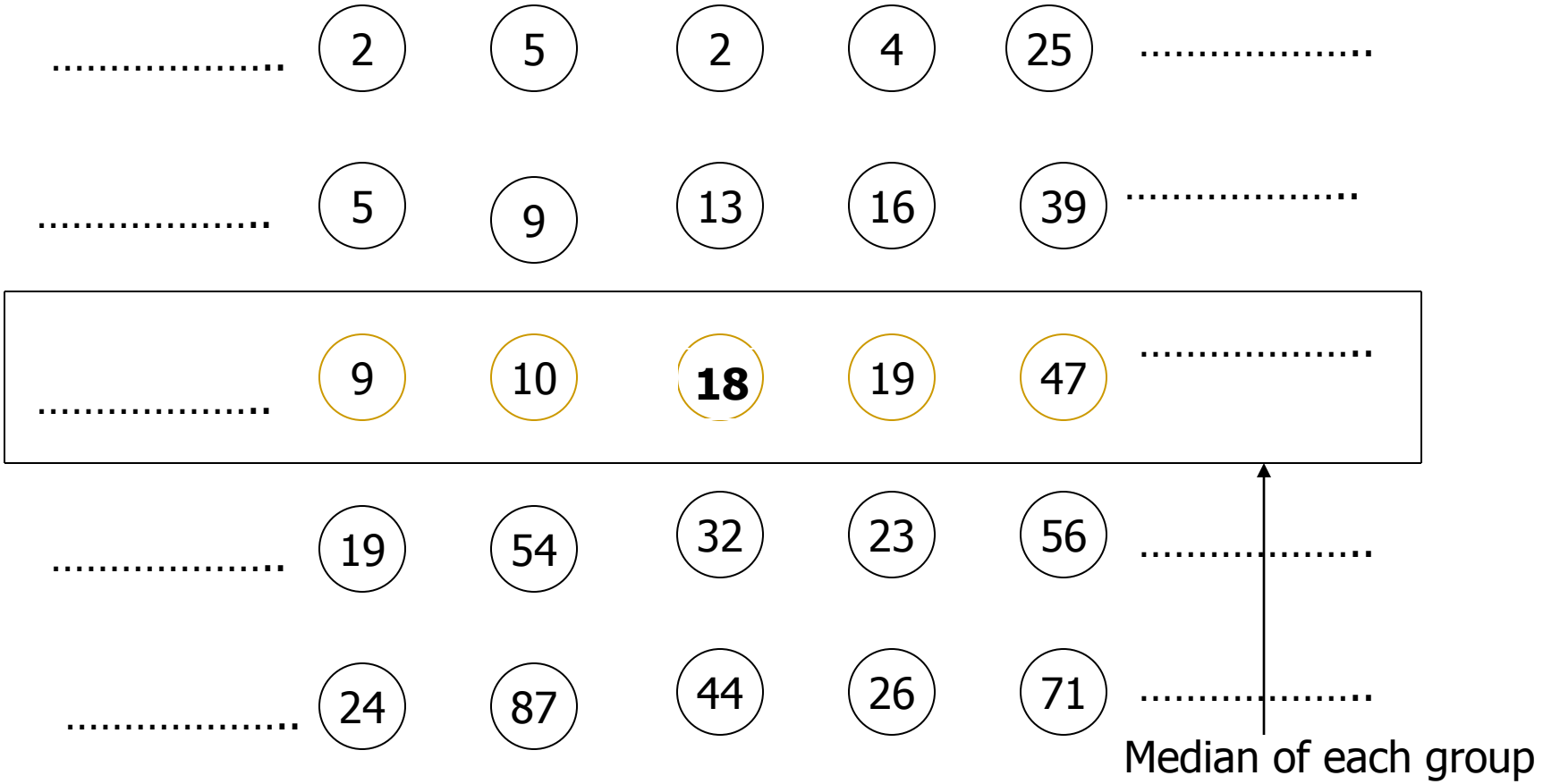
Selection in Linear Time

- ***Step1:*** Group numbers in sets of 5 (Vertically)



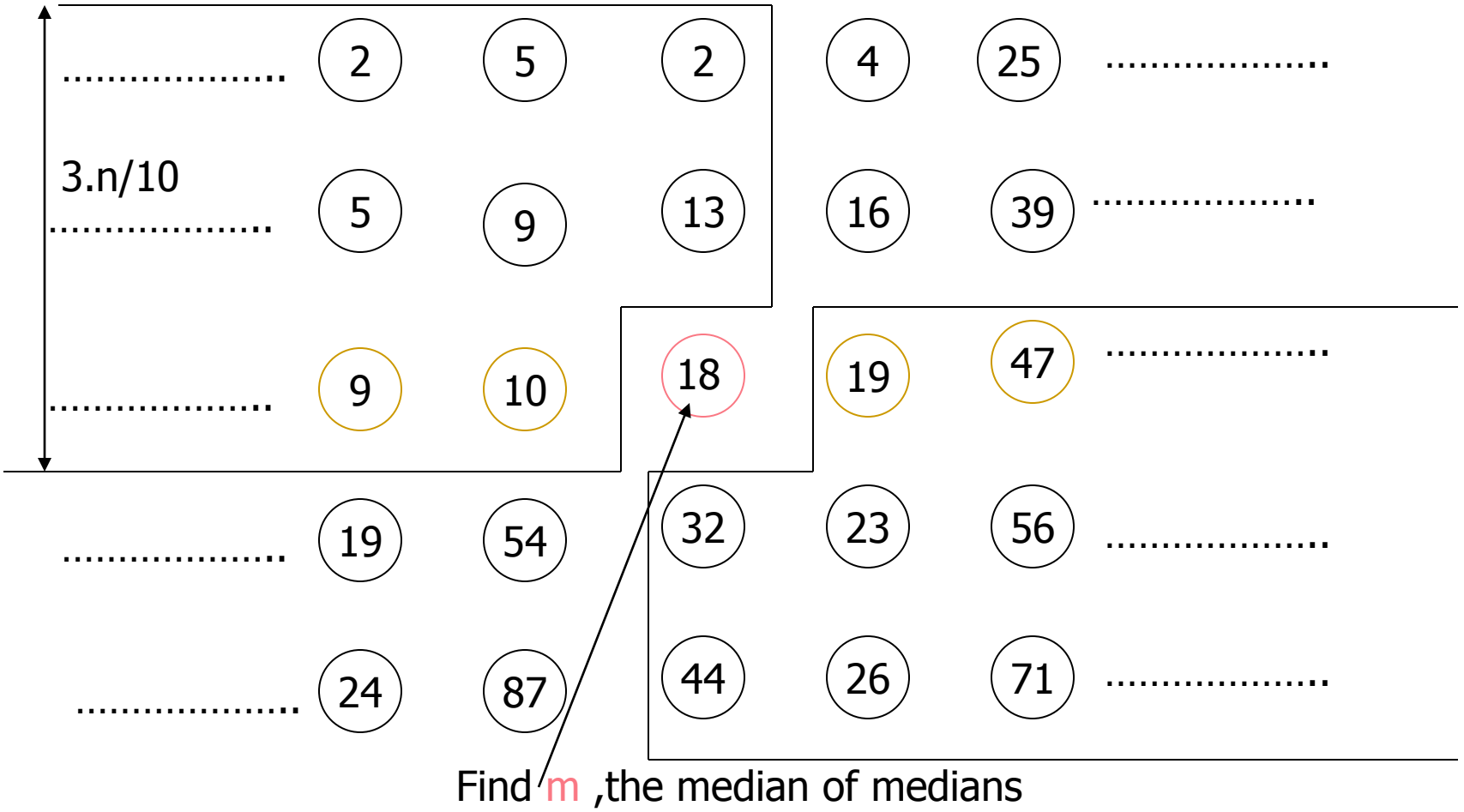
Selection in Linear Time

- ***Step2:*** Find Median of each group



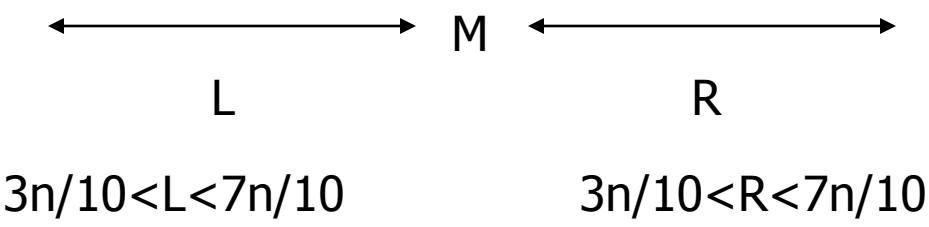
Selection in Linear Time

- ***Step3:*** Find the MedianOfMedians



Selection in Linear Time

- ***Step4:*** Partition original data around the MOM



Step5: If |L| = k-1, then return MOM else
If |L| > k-1, then return kth_smallest (L,k) else
return kth_smallest (R,k-(|L|+1))

Selection in Linear Time

- ***Time Analysis:***

Step	***Task***	***Complexity***
1	Group into sets of 5	O (n)
2	Find Median of each group	O (n)
3	Find MOM	T (n/5)
4	Partition around MOM	O (n)
5	Condition	T (7n/10) {Worst Case}

Selection in Linear Time

- ***Time Complexity of Algorithm:***
- $T(n) = O(n) + T(n/5) + T(7n/10)$
- $T(1) = 1$
- Assume $T(n) \leq Cn$ (For it to be linear time)
- L.H.S = Cn
- R.H.S = $C_1 n + Cn/5 + 7Cn/10 = (C_1 + 9/10C)n$
- Hence, L.H.S = R.H.S if $\boldsymbol{C = 10C_1}$
  - ***Thus it is a Linear Time Algorithm***

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?

QuickSelect

Linear time algorithm has a large constant

- How to proceed in practice?

QuickSelect(A,k)

- Select a pivot Random p
- 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

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

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