Divide & Conquer

Prepared by *Pawan K. Mishra*

A general theme…

$$\text{Prob}(A[1..n]) = \begin{cases} \text{Combine}(\text{Prob}(A[1..m]), \text{Prob}(A[1..m+1])), & \text{if } n > n_0 \\ \text{Basis solution}, & \text{if } n = n_0 \end{cases}$$

m = n/2, n/4 e.g.. Merge Sort, Binary Search…

m depends on the partition e.g.. Quick Sort

Meaning of D&C

Divide the problem into a number of subproblems that are smaller instances of the same problem.

Conquer the subproblems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner.

Combine the solutions to the subproblems into the solution for the original problem

P1. Finding the minimum of n elements in an array

fmin(A, begin, end)

If (begin==end) return A[begin] → T(1)=1

Else

mid= $\lfloor$(begin+end-1)/2)$\rfloor$

Return min{fmin(begin,mid), fmin(mid+1, end)}

1 T(n/2) T(n/2)

$T(n)= 1 + T(n/2)+ T(n/2)= 2T(n/2)+1$

Can we divide in other fractions?

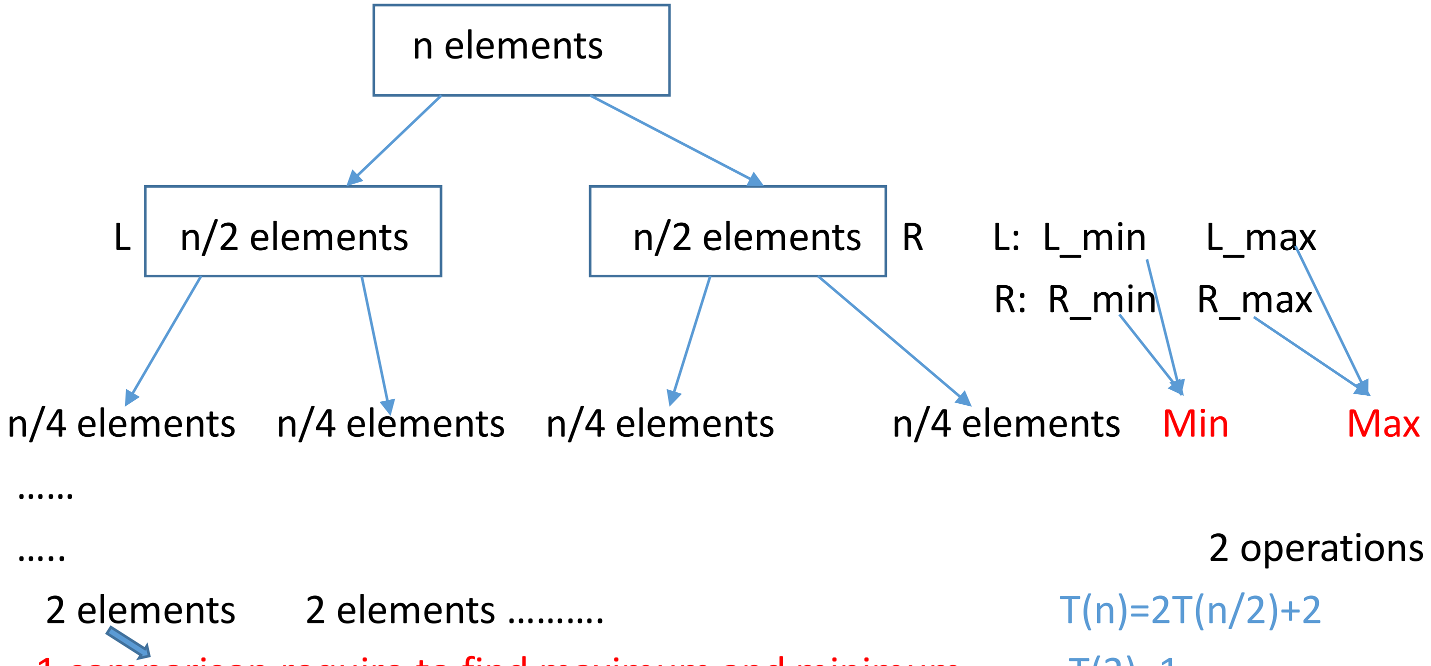
$n/3, 2n/3$

$$T(n)=T(n/3)+T(2n/3)+1 \text{ ??}$$

$n/3, n/3, n/3$

$$T(n)=3T(n/3)+1 \text{ ??}$$

P2. Finding the maximum and the minimum of n elements in an array.



1 comparison require to find maximum and minimum.

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

$T(2)=1$

Prepared by Pawan K. Mishra

P3. Finding the maximum and the second maximum of n elements in an array.

Similarly, we can solve this problem too.

P4. Binary Search

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

P5. Merge Sort

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

$T(2)=\text{constant}$

Merging two sorted arrays ??

Template: D&C

```
RecursiveD&C(A[1..n])
        solve base case (n_0)                ---- in constant time
         if (n > n_0)
              dosomething(A)                   other steps depend on problem
              A_1= extract(A)
              A_2= extract(A)
              B_1= RecursiveD&C(A_1)
              B_2= RecursiveD&C(A_2)
               B  =rebuild(B_1, B_2)
           Return B
```

T(n)= contant, Base Case

= aT(f(n))+bT(g(n))+.. +ExtraWork

dosomething(A) { findmin,   findmid
findmax, findmedian

extract(A) ---- divide(A), partition(A)

rebuild(B_1,B_2)– merge, compare

P7. Compute a^n

I/P: a and an intger n

O/P: a^n.

P6. Bisection method: root finding

I/P: A polynomial function f(x), and +ve integer n

O/P:integer root “r” s.t. f(r)=0, where $0 \le r < n$

e.g. f(x)=x^2-x-12=0, n=8

r=4

Prove via induction

$T(n)=2T(n/2)+ c\,n$

$T(2)$=constant

Thm: $T(n)=O(n)$ A guess… so need to prove via induction.

Proof: (by induction)

Base Case: $T(2)$=constant $= O(1)$

Ind Hyp: Assume statement is true for $k<n$, i.e., $T(k)=O(k)$. [Strong Induction]

Ind Step: To prove theorem for $T(n)$, using ind hyp $T(n/2)=O(n/2)$

$T(n/2) \leq dn/2$

$$T(n)= 2T(n/2)+ c\,n$$

$$\leq 2d(n/2)+ c\,n \quad \text{(by Hypothesis)}$$

$$=(c+d) = O(n)$$

can you find the the mistake ?

Prove via induction

$T(n)=2T(n/2)+ c\ n$

$T(1)=\text{constant}$

Thm: $T(n) \leq b\ n$ A guess… so need to prove via induction

Proof: (by induction)

Base Case: $T(1)=\text{constant} \leq b.2=b$ [here, constant/2 $\leq b$]

Ind Hyp: Assume statement is true for $k<n$, i.e., $T(k) \leq bk$. [Strong Induction]

Ind Step: To prove theorem for $T(n)$, using ind hyp $T(n/2) \leq b\ n/2$

$T(n)= 2T(n/2)+ c\ n$

$\leq 2b(n/2)+ c\ n$ (by Hypothesis)

$= (b+c)n$

$\neq bn$ **So, the guess is wrong**

Prove via induction

$T(n)=2T(n/2)+ c\, n$

$T(2)=\text{constant}$

Thm: $T(n) \leq b\, n \log n$ A guess and we later find 'b'. Need to prove via induction.

Proof: (by induction)

Base Case: $T(2)=\text{constant} \leq b.2\log 2=2b$ [here, constant/2 $\leq$ b]

Ind Hyp: Assume statement is true for $k<n$, i.e., $T(k) \leq bk$. [Strong Induction]

Ind Step: To prove theorem for $T(n)$, using ind hyp $T(n/2) \leq b\, n/2 \log n/2$

$T(n)= 2T(n/2)+ c\, n$

$\leq 2\, b\, (n/2) \log n/2+ c\, n$ (by Hypothesis)

$= 2\, b\, n/2 \log (n/2) +c\, n = bn \log n - bn \log 2+cn = bn \log n - bn +cb$

$=bn\log n-(bn-cn) \leq bn\log n$

b =max{constant/2, c} – this guess will work (if $bn-cn>0$, $b>c$)

Not D&C Problem, but an interesting problem

Let given two sorted array A and B, find number of pairs (i, j) such that A[i] > B[j].

For eg. A= [3,6,7,9] B=[1, 2, 5, 8, 10]

Output= 2+3+3+4=12

Let given two sorted arrays A and B, find the number of pairs (i, j) such that A[i] > B[j].

For eg. A= {3,6,7,9} B={1, 2, 5, 8, 10}

Output= 2+3+3+4=12

Naïve approach: check for each element of A, traverse in B.

Merge and Count

A= {3,6,7,9} B={1, 2, 5, 8, 10}

Lets Merge to get sorted order

Big Sorted Array= [1,2,3,5,6,7,8,9,10]

Merge and Count

A= {3,6,7,9} B={1, 2, 5, 8, 10}

Lets Merge to get sorted order

Big Sorted Array= [1,2,3,5,6,7,8,9,10] – element in red came from array B.

Merge and Count

A= {3,6,7,9} B={1, 2, 5, 8, 10}

Lets Merge to get sorted order

Big Sorted Array= [1,2,3,5,6,7,8,9,10] – elements in red came from the array B.

Lets see the status of array A, when an element came from the array B in the big sorted array.

Merge and Count

A= {3,6,7,9} B={1, 2, 5, 8, 10}

Lets Merge to get sorted order

Big Sorted Array= [1,2,3,5,6,7,8,9,10] – element in red came from array B.

Lets see the status of array A, when element entered from B in the big sorted array.

When 1 entered– How many elements are there in array A which are not traversed fully by the pointer of A? 4

When 1 entered– How many elements are there in array A which are not traversed fully by the pointer of A? 4

When 2 entered– How many elements are there in array A which are not traversed fully by the pointer of A? 4

When 5 entered– How many elements are there in array A which are not traversed fully by the pointer of A? 3

When 8 entered– How many elements are there in array A which are not traversed fully by the pointer of A? 1

When 10 entered– How many elements are there in array A which are not traversed fully by the pointer of A? 0

12

Merge and Count

Array got sorted and we got the count in $O(n)$

P8: Counting Inversion

I/P: Given an array A of distinct integers.

O/P: The number of inversions of A is the number of pairs (i,j) of array A with i<j and A[i] > A[j].

A=[1, 3, 5, 4, 2]

0 + 1 + 2 + 1 + 0 = 4 pairs (3,2), (5,4) (5,2) (4,2)

Application of inversion

My web-series order:

Breaking Bad Prison Break Panchayat Game of Thrones Mirzapur

1 2 3 4 5

Friend's web-series order:

Breaking Bad, Panchayat, Mirzapur , Game of Thrones, Prison Break

1 3 5 4 2

Naïve approach

```
inversion=0
for i=1 to n-1
    for j=i+1 to n
        if A[i] > A[j] then
            inversion=inversiom +1
Return inversion
```

$O(n^2)$-algorithm.

Can we do better?

Using D&C. Any suggestion?

Divide problem in subproblems of size half and recurse, be smart to combine the solutions of subproblem.

problem of size n

L prob. of size n/2 | prob. of size n/2

$T(n)=2T(n/2)+$ time required to combine solutions of two subproblems

inversions in L = N_L # inversions in L = N_R

Total = $N_L + N_R$ + number of pairs (i,j) s.t. L[i] > R[j]

inversions in L = N_L # inversions in L = N_R

Total = N_L + N_R + $\underbrace{\text{number of pairs (i,j) s.t. L[i] > R[j]}}_{}$

This comes from combining solution

A way to combine solutions:

We need to solve the problem in O(nlogn), and we are traversing both halves, so we cannot go beyond O(n) to do extra work.

inversions in L = N_L # inversions in L = N_R

Total = $N_L + N_R$ + $\underbrace{\text{number of pairs (i,j) s.t. } L[i] > R[j]}$

This comes from combining solution

Even smarter way to combine solutions:

Call Merge and Count routine (Last problem)

$$T(n)=2T(n/2)+O(n)=O(n\log n)$$

P9:Integer Multiplication: two n-digits numbers

I/P: $A = a_1 a_2 a_3 \ldots. a_n$

$B = b_1 b_2 b_3 \ldots. b_n$

O/P: A*B

Eg:

```
      5678
      1234
   -------
    22 712
  17 034-
 11 356--
 56 78---
 --------
 7 0 0 6 5 2
```

2n operations are required: n for multiplications

n for carry additions.

for each row: 2n operations

n rows: thus $2n^2$ operations in total.

need to add each column (at most $2n^2$ operations)

D&C Approach

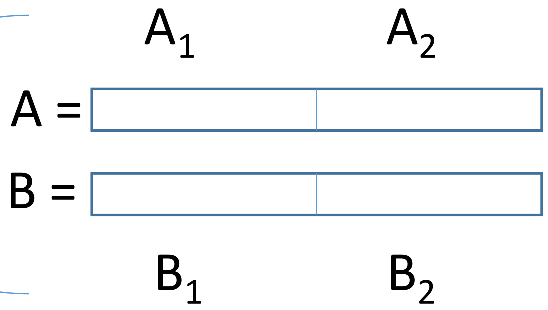
I/P: $A = a_1 a_2 a_3 \ldots a_n$

$B = b_1 b_2 b_3 \ldots b_n$

O/P: $A*B$

$A = a_1 a_2 a_3 \ldots a_n$

$B = b_1 b_2 b_3 \ldots b_n$



$5678 = 56*10^2 + 78$

$1234 = 12*10^2 + 34$

$A = A_1 * 10^{n/2} + A_2$

$B = B_1 * 10^{n/2} + B_2$

$A_1 \quad A_2$

$A = a_1a_2a_3 \ldots.a_n$ $\quad A = \boxed{ \mid }$

$B = b_1b_2b_3\ldots.b_n$ $\quad B = \boxed{ \mid }$

$B_1 \quad B_2$

$A = A_1 * 10^{n/2} + A_2$

$B = B_1 * 10^{n/2} + B_2$

$AB = A_1 * B_1 * 10^{n} + A_1 * B_2 * 10^{n/2} + A_2 * B_1 * 10^{n/2} + A_2 * B_2$

D&C Approach

$A = A_1 * 10^{n/2} + A_2$

$B = B_1 * 10^{n/2} + B_2$

$AB = \boxed{A_1 * B_1} * 10^n + \boxed{A_1 * B_2} * 10^{n/2} + \boxed{A_2 * B_1} * 10^{n/2} + \boxed{A_2 * B_2}$

T(n/2)

for shifting and adding

$T(n) = 4\,T(n/2) + c\,n$

shifting is easy

Another D&C Approach: Karatsuba Algo

$$AB = A_1 * B_1 * 10^n + A_1 * B_2 * 10^{n/2} + A_2 * B_1 * 10^{n/2} + A_2 * B_2$$

$$= A_1 * B_1 * 10^n + (A_1 * B_2 + A_2 * B_1) * 10^{n/2} + A_2 * B_2$$

$$(A_1 + A_2)(B_1 + B_2) = A_1B_1 + A_1B_2 + A_2B_1 + A_2B_2$$

$$A_1B_2 + A_2B_1 = (A_1 + A_2)(B_1 + B_2) - A_1B_1 - A_2B_2$$

Another D&C Approach: Karatsuba Algo

$$AB = A_1 * B_1 * 10^n + A_1 * B_2 * 10^{n/2} + A_2 * B_1 * 10^{n/2} + A_2 * B_2$$

$$= A_1 * B_1 * 10^n + (A_1 * B_2 + A_2 * B_1) * 10^{n/2} + A_2 * B_2 \quad \ldots\ldots.(1)$$

$$(A_1 + A_2)(B_1 + B_2) = A_1B_1 + A_1B_2 + A_2B_1 + A_2B_2$$

$$A_1B_2 + A_2B_1 = (A_1 + A_2)(B_1 + B_2) - A_1B_1 - A_2B_2 \quad \ldots..(2)$$

(by putting 2 in 1)

$$AB = A_1 * B_1 * 10^n + A_1 * B_2 * 10^{n/2} + A_2 * B_1 * 10^{n/2} + A_2 * B_2$$

$$= A_1 * B_1 * 10^n + ((A_1 + A_2)(B_1 + B_2) - A_1B_1 - A_2B_2) * 10^{n/2} + A_2 * B_2$$

Another D&C Approach: Karatsuba Algo

$$AB = A_1 * B_1 * 10^n + A_1 * B_2 * 10^{n/2} + A_2 * B_1 * 10^{n/2} + A_2 * B_2$$

$$= \boxed{A_1 * B_1} * 10^n + (\boxed{(A_1 + A_2)(B_1 + B_2)} - A_1B_1 - A_2B_2) * 10^{n/2} + \boxed{A_2 * B_2}$$

$T(n/2)$

sum of two n/2 digits can give you at most n/2+1 digits

Thus, we are multiplying two (n/2+1) digits number …

Another D&C Approach: Karatsuba Algo

$$AB = A_1 * B_1 * 10^n + A_1 * B_2 * 10^{n/2} + A_2 * B_1 * 10^{n/2} + A_2 * B_2$$

$$= \boxed{A_1 * B_1} * 10^n + (\boxed{(A_1 + A_2)(B_1 + B_2)} - A_1B_1 - A_2B_2) * 10^{n/2} + \boxed{A_2 * B_2}$$

$$T(n/2)$$

$$T(n) = 3T(n/2) + cn = O(n^{\log 3}) = O(n^{1.58})$$

Example

$A*B=1234*4321$

$A_1 = 12$ $\quad A_1B_1=12*43 \quad A_2B_2=34*21$

$A_2 = 34$ $\quad (A_1+A_2)(B_1+B_2)-A_1B_1-A_2B_2$

$B_1 = 43$ $\quad = (12+34)(43+21)-A_1B_1-A_2B_2$

$B_2 = 21$

Example

$A*B=1234*4321$

$A_1 = 12$ $A_2 = 34$ $B_1 = 43$ $B_2 = 21$

$A_1B_1=12*43$ $A_2B_2=34*21$

$(A_1+A_2)*(B_1+B_2)-A_1B_1-A_2B_2$

$= (12 +34)*(43+21)- A_1B_1-A_2B_2$

$= (46)*(64)- A_1B_1-A_2B_2$

$A_1B_1=12*43$

1 2

4 3

$1*4=4$

$2*3=6$

$(1+2)(4+3)-4-6=11$

$4*10^2+11*10+6=516$

$A_2B_2=34*21$ 3 4

2 1

$3*2=6$

$4*1=4$

$(3+4)(2+1)-6-4=11$

$6*10^2+11*10+4=714$

4 6

6 4

$4*6=24$

$6 *4=24$

$(4+6)(4+6)-24-24=52$

$24*10^2+52*10+24=2944$

Example

$A*B=1234*4321$

$A_1 = 12 \quad A_2 = 34 \quad B_1 = 43 \quad B_2 = 21$

$A_1B_1=12*43 \qquad A_2B_2=34*21$

$(A_1+A_2)*(B_1+B_2)-A_1B_1-A_2B_2$

$= (12 +34)*(43+21)- A_1B_1-A_2B_2$

$= (46)*(64)- A_1B_1-A_2B_2$

Final Solution:

$A_1B_1 10^4 +((A_1+ A_2)(B_1+ B_2) - A_1B_1 - A_2B_2)10^2 + \quad A_2 B_2$

$516 *10^4+(2944-714-516) *10^2+714$

$=5332114$

$A_1B_1=12*43$

1 2

4 3 $(A_1+A_2)*(B_1+B_2)$

$1*4=4$

$2*3=6$

$(1+2)(4+3)-4-6=11$

$4*10^2+11*10+6=516$

$A_2B_2=34*21$ 3 4

2 1

$3*2=6$

$4*1=4$

$(3+4)(2+1)-6-4=11$

$6*10^2+11*10+4=714$

4 6

6 4

$4*6=24$

$6 *4=24$

$(4+6)(4+6)-24-24=52$

$24*10^2+52*10+24=2944$

Prepared by Pawan K. Mishra

K-th smallest element

- I/P: Array of n numbers.
- OP: Select k-th smallest element

Naïve way: Sort it, and return it. O(nlogn)

Can we do better ? ---- in O(n)??

YES!!!!! ------"**Median of Median**"- algorithm

MANUEL BLUM, ROBERT W. FLOYD, VAUGHAN PRATT, RONALD L. RIVEST, AND ROBERT E. TARJAN

Let's try..Select(A,k)

Suppose we select a pivot element x of Array A (not randomly, just to explain).

A can be portioned in two sets... A_1={all elements of A < x}

A_2={all elements of A > x}

Select(A, k)

choose x;

If $|A_1|$= k-1, return x;

If k < $|A_1|$, Select(A[1], k)

Else Select(A[2], k-$|A_1|$-1)

We are throwing away something, but might not be in fractions

Unlucky in all the recursive call

$T(n) <= T(n-1)+ O(n)$ ----- $O(n^2)$

for partition

Let's try..Select(A,k)

Suppose we select a pivot element x of Array A (not randomly, just to explain).

A can be portioned in two sets… A_1={all elements of A < x}

A_2={all elements of A > x}

Select(A, k)

choose x;

If $|A_1|$= k-1, return x;

If k < $|A_1|$, Select(A[1], k)

Else Select(A[2], k-$|A_1|$-1)

in each recursive call

Suppose, the selected pivot throw away this much 0.3n=3n/10,

$T(n) <= (\ ) + T(7n/10) + O(n)$

for partion

Prepared by Pawan K. Mishra

T(n) <= (need to do something to get a good pivot)+ T(7n/10)+ O(n)

Divide array A into group of size 5.

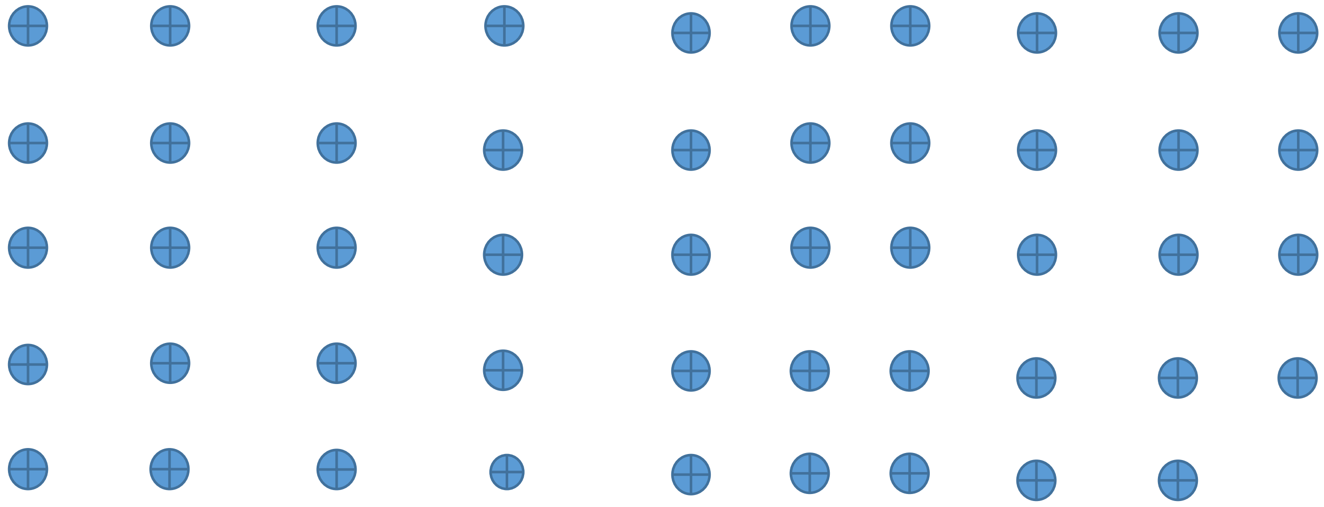
n/5 groups

Find the median of each group.

3rd element in each group– create the list of all selected -- B

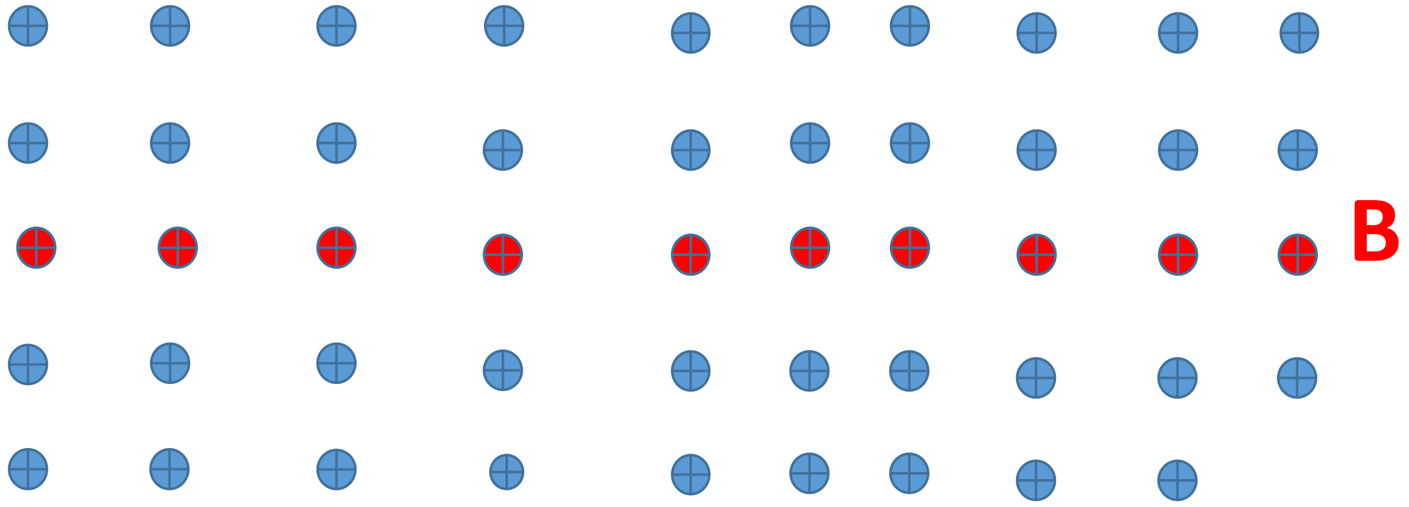
Find median of medians using the same Select(B, n/10)

Array A divided into group of 5, n/5 groups….

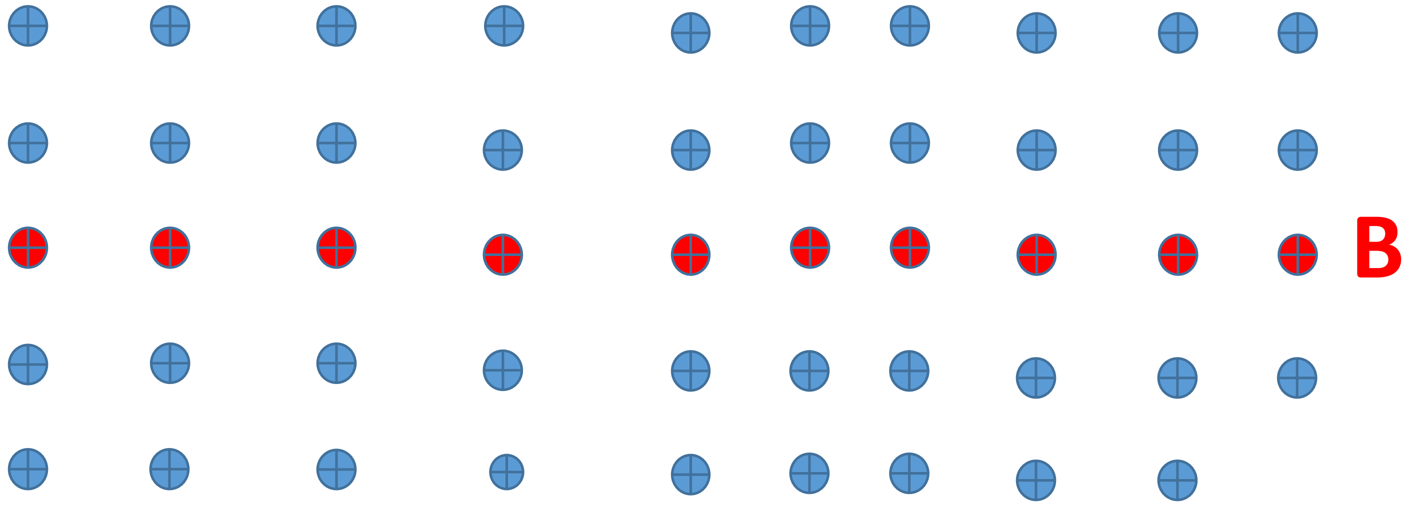


Prepared by *Pawan K. Mishra*

Array A divided into group of 5, n/5 groups….



Array A divided into group of 5, n/5 groups….

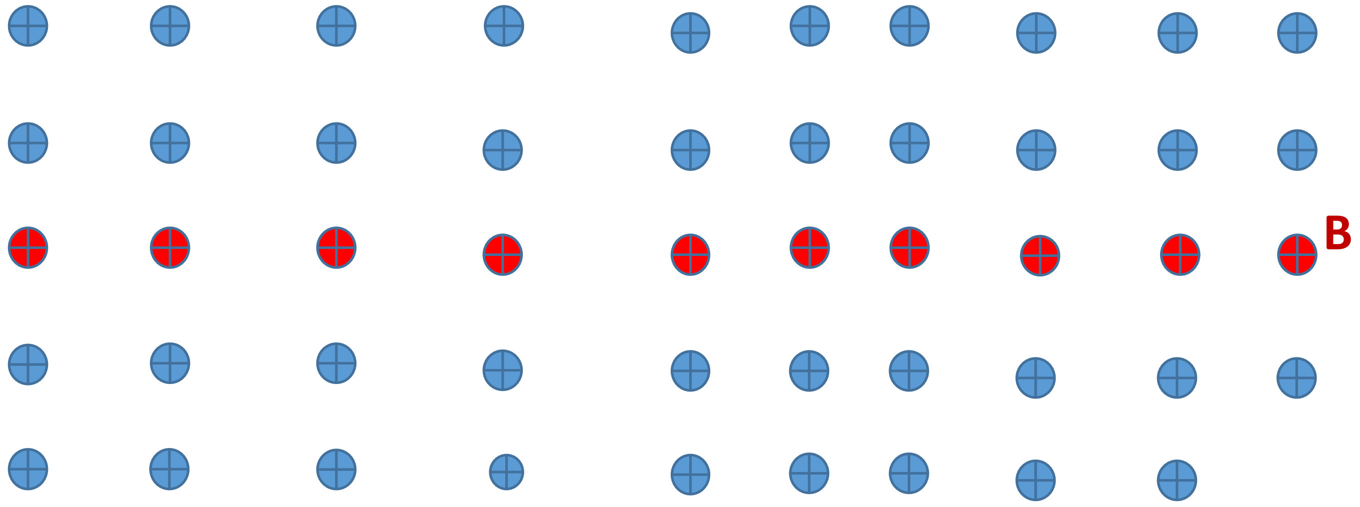


Prepared by *Pawan K. Mishra*

.

A divided into group of 5, n/5 groups….

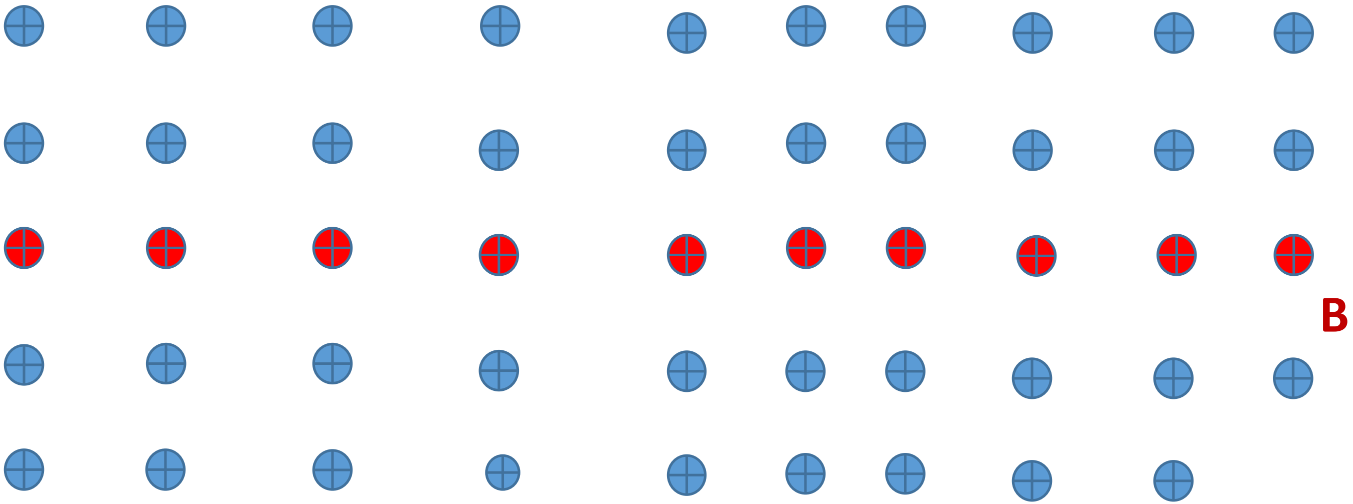
Median in constant time for each group, as each group contain 5 elements.



.

A divided into group of 5, n/5 groups….

Median in constant time for each group, as each group contain 5 elements.   So, overall linear time. Add to extra work O(n).

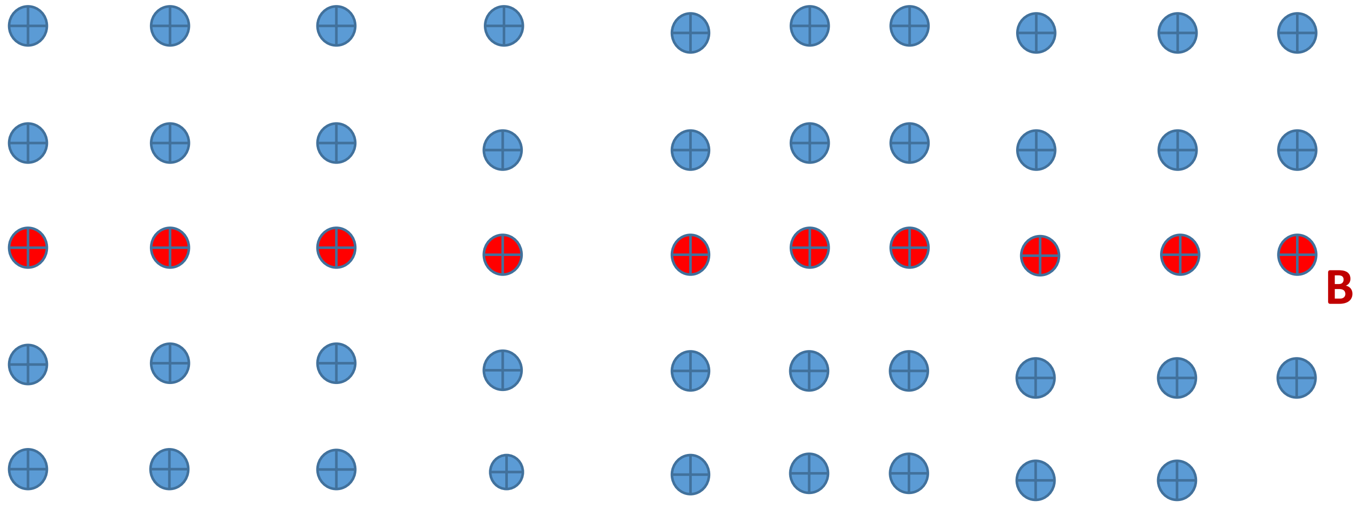


.

A divided into group of 5, n/5 groups….

Median in constant time for each group, as each group contain 5 elements. So, overall linear time. Add to extra work O(n).

Median of Medians- Select(B, n/10)



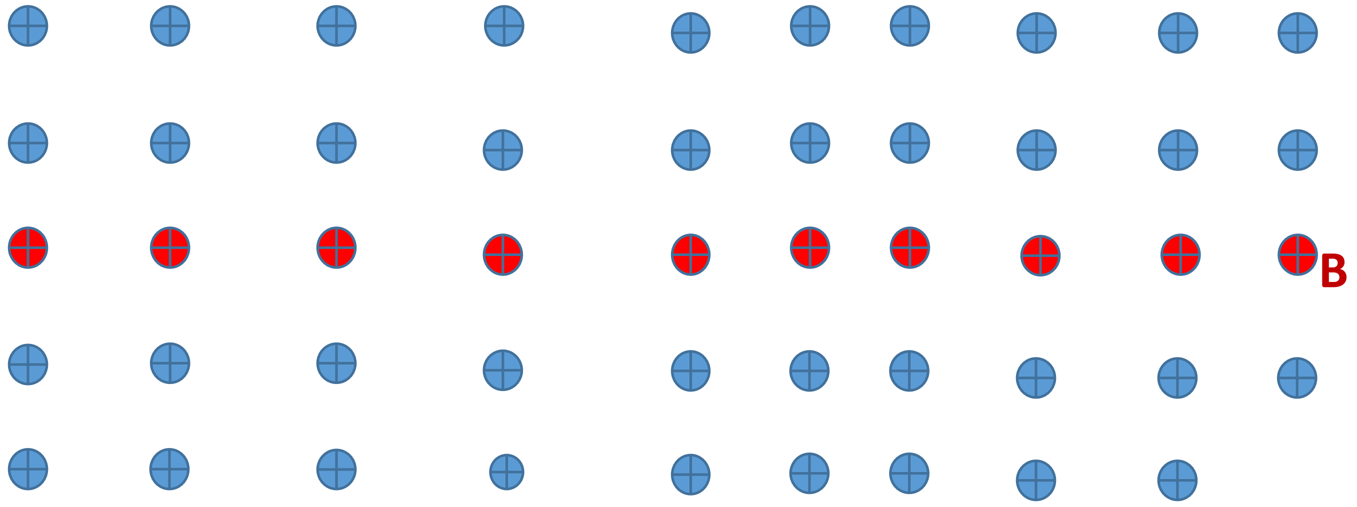
.

A divided into group of 5, n/5 groups….

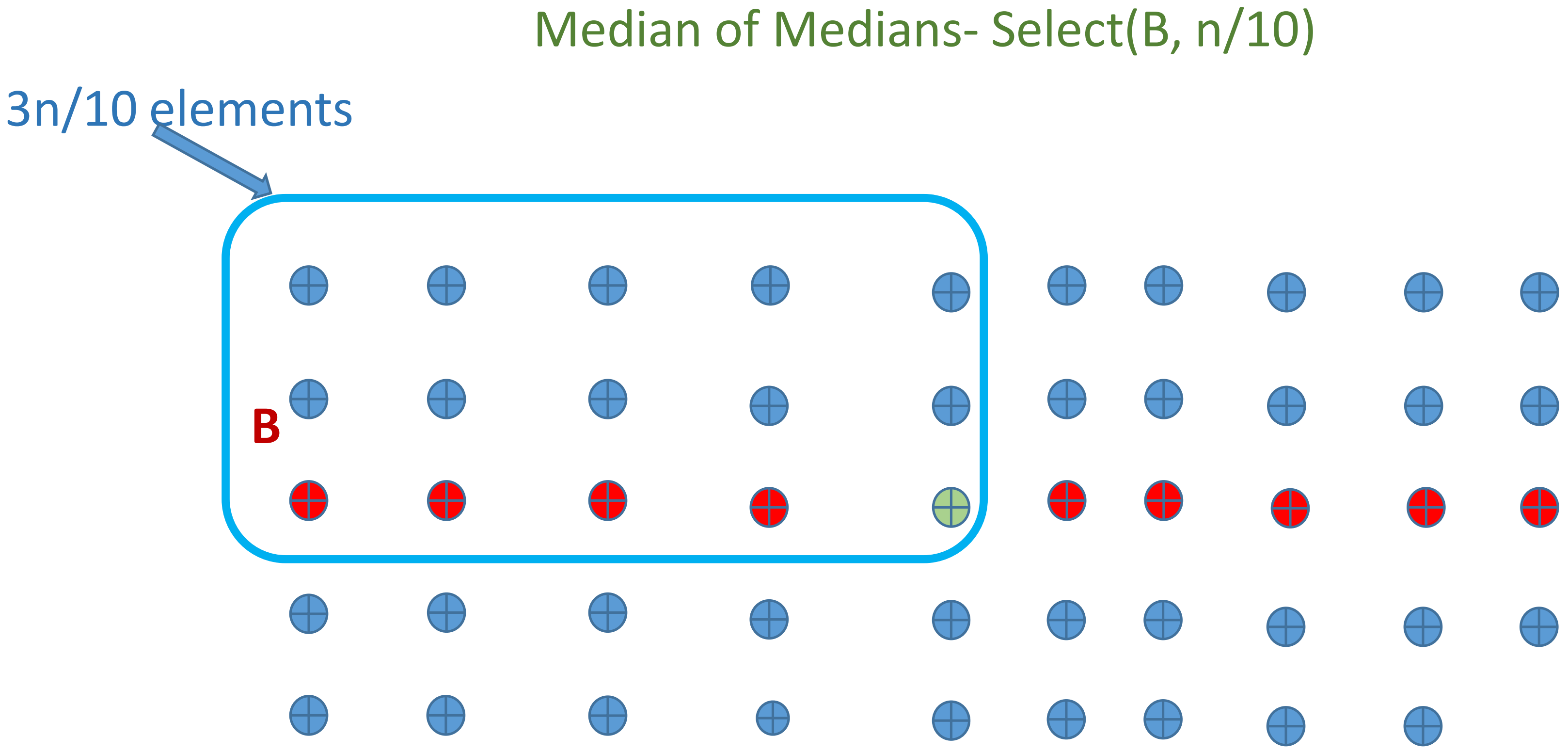
Median in constant time for each group, as each group contain 5 elements. So, overall linear time. Add to extra work O(n).

Median of Medians- Select(B, n/10)

T(n/5) is required….

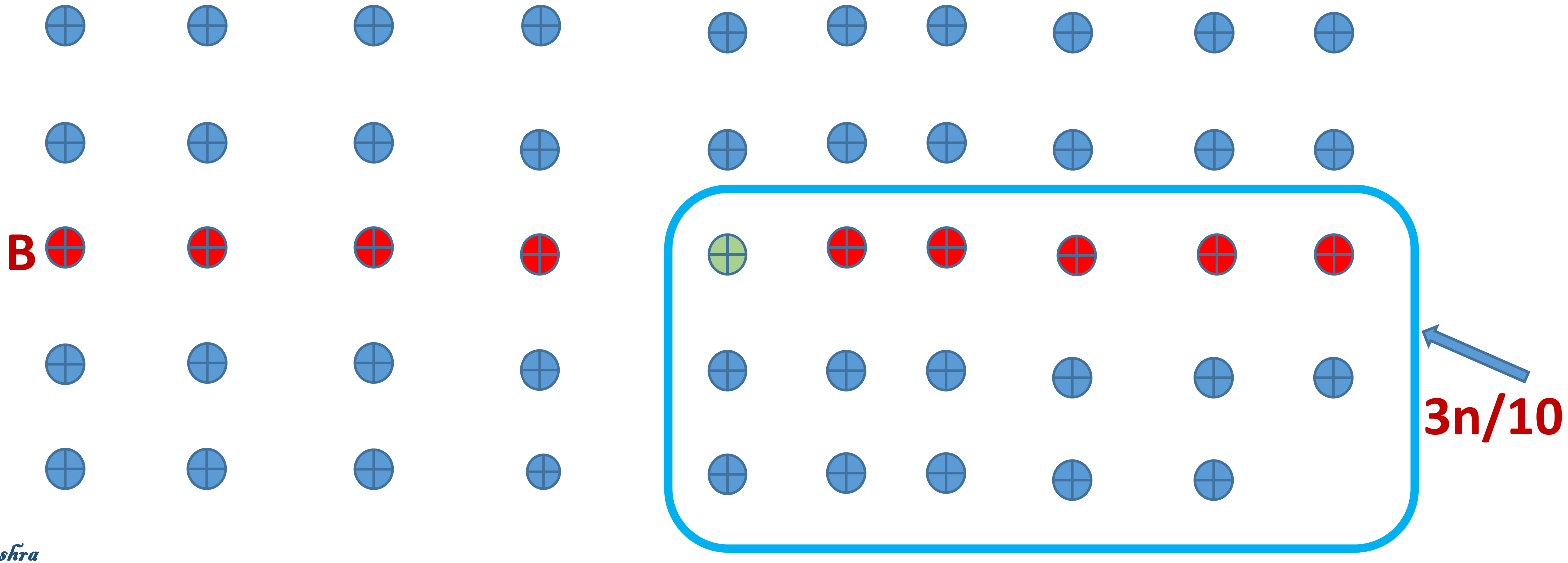


Claim: The median of median will be a good pivot, i.e., it throw away 3n/10 elements.



Claim: The median of median will be a good pivot, i.e., it throw away 3n/10 elements.

Median of Medians- Select(B, n/10)



Prepared by *Pawan K. Mishra*

Claim: The median of median will be a good pivot, i.e., it throw away 3n/10 elements.

Select(A, k) choose x; ← median of medians

Create partition A_1 and A_2 based on median of medians

If $|A_1| = k-1$, return x;

If $k < |A_1|$, Select(A[1], k) ← because of x, there are at least 3n/10 elements in A_1

Else

Select(A[2], $k-|A_1|-1$) ← because of x, there are at least 3n/10 elements in A_2

Since, $|A_1|$ => 3n/10, therefore $|A_2| < 7n/10$

and

Since, $|A_2|$ => 3n/10 , therefore $|A_1| < 7n/10$

Thus both A_1 and A_2 are bounded by 7n/10 elements.

Claim: The median of median will be a good pivot, i.e., it throw away 3n/10 elements.

$|A_1|$ => 3n/10 implies $|A_2|$ < 7n/10

Select(A, k)
choose x; ← median of medians
Create partition A_1 and A_2 based on median of medians
If $|A_1|$= k-1, return x;
If k < $|A_1|$, Select(A[1], k) ← because of x, there are at least 3n/10 elements in A_1
Else
Select(A[2], k-$|A_1|$-1) ← because of x, there are at least 3n/10 elements in A_2

$|A_2|$ => 3n/10 implies $|A_1|$ < 7n/10

Time Complexity Analysis.

- $T(n)=T(n/5)+T(7n/10)+cn$

To prove $T(n)= O(n)$

Guess: $T(n) <=Rn$ (for some R, we will later fix this R)

Ind Hyp: $T(n)<=Rn$ is true for all $k<n$.

- Ind step: $T(n)=T(n/5)+T(7n/10)+cn <= Rn/5+ R7n/10+cn$

$=(0.2R+0.7R+c)n=(0.9R+c)n$

If $10\ c = R$ $\quad =Rn$