3/12/18

UNIT-2

"Divide & conquer"

→ General Method (or) Abstraction Method of Divide and conquer
→ Defective chess board
→ Binary search
→ Quick Sort
→ Merge Sort
→ Finding Maximum and Minimum
→ performance Measurement
→ Randomized Sorting Algorithm. eg:- quick sort

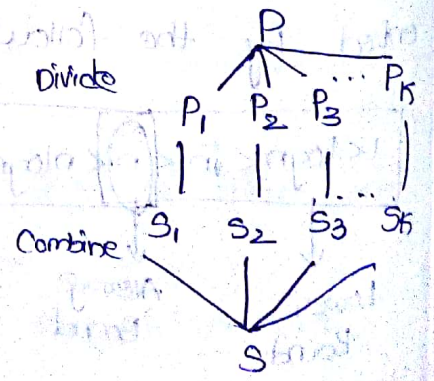
Divide and conquer (General Method):-

* If the given problem is a small problem then return a solution
* If the given problem is a large then apply divide and conquer method.
* If the given problem is large then it can be divided into possible number of sub problems.
* For each sub problem we have to find out solutions
* Finally all the solutions are combined

Explain Divide & Conquer Method Algorithm:-

```
DAC (p)
{
 If (small (p))
  {
   S(p);
  }
 else
  {
  divide P into P1, P2, P3, ··· Pk
  Apply DA(P1), DA(P2), DA(P3) ···· DA(Pk)
  Combine DA(P1), DA(P2), DA(P3) ···· DA (Pk)
  }
}
```

Tree Representation



Note:2

* All the divide and conquer algorithms are recursive hence we can find out the time complexity using Recurrence Relation

Binary Search:-

* In the binary search method the elements are sorted order that is ascending order.
* If we want search for the element say 'x'.
* Then we first divided the list at middle so that two sublists are created.
* If x greater than middle then right sub list consider and x is searched in the right sub list otherwise x is searched in the left sub list.

eg:- the given list of elements are 3, 6, 8, 12, 14, 17, 25, 29, 31, 36, 42, 47, 53, 55, 62. and key element is 42. Then find a key element 42 is available are not and also find time complexity

* Consider the given list of elements are

3	6	8	12	14	17	25	29	31	36	42	47	53	55	62
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

let us assume $l=1$

high $h=n=15$

then find mid value mid $(m) = \lfloor (l+h)/2 \rfloor$

if always low lessthan are equal to high then find mid

mid $(m) = \frac{(l+h)}{2}$.

if (key == mid) then $42 == 29$ hence it is false.

if (key < mid)

$42 < 29$ false

if (key > mid)

$42 > 29$ True

then low = mid+1
= 8+1
l = 9

now l=9, h=15, m = $\frac{9+15}{2} = \frac{24}{2} = 12$.

if (key == mid) then
42 == 47 false

if (key < mid) then
42 < 47 True.

high = mid - 1
= 12-1
h = 11

now h=11, l=9, m = $\frac{11+9}{2} = \frac{20}{2} = 10$

if (key == mid)
42 == 36 false

if (key < mid) then
42 < 36 false

if (key > mid) then
42 > 36 True
low = mid+1
= 10+1
l = 11

now l=11, h=11, m = $\frac{11+11}{2} = 11$

if (key == mid) then
42 == 42 True

the element is found.

~~now key is 47.~~

```
while (l<=h)
key == mid
    return m
key < mid
h = mid - 1;
key > mid
l = mid + 1;
else
return 0;
```

The Time complexity of binary Search is

	Best case	Average Case	Worst Case
Successful	O(1)	O(logn)	O(logn)
unSuccessful	O(logn)	O(logn)	O(logn)

Recursive Algorithm for Binary Search:- (Iterative)

```
int BinarySearch(A, n, key)
    {
     l=1, h=n
     while (l≤h)
     {
      mid = (l+h)/2;
      if (key==A[mid])
       return mid;
       if (key< A[mid])
         h=mid-1;
         }
         return 0;
       }
```

else
l=mid+1;

Recursive algorithm:-

```
Algorithm RecursiveBinarySearch(l, h, key)
{
 if (l==h)
 {
  if (A[l]==key)
  return l;
  else
     return 0;
   }
   else
     {
     mid = (l+h)/2;
     if (key==A[mid])
      return mid;
     if (key<A[mid])
     return RecursiveBinarySearch(l, mid-1, key);
     else
      return RecursiveBinarySearch(mid+1, h, key);
  }}
```

Recurrence Relation:

$$T(n) = \begin{cases} 1 & \text{if } n=1 \\ T(n/2)+1 & \text{if } n>1 \end{cases}$$

prove that the time complexity for average and worst case is $O(\log n)$ and unsuccessful search in three cases is also $O(\log n)$.

Consider $T(n) = T(n/2) + 1$ —①

Now substitute n is replaced by $n/2$ in equ ① then we get

$$T(n) = \left(T\left(n/2^2\right)+1\right)+1$$

$$= T\left(n/2^2\right) + 2 \rightarrow ②$$

$$= T\left(n/2^3\right) + 3 \rightarrow ③$$

$$= T\left(n/2^4\right) + 4 \rightarrow ④$$

Assume $\frac{n}{2^k} = 1$ then $n = 2^k$ then $k = \log_2 n$

$$T(n) = T\left(n/2^k\right) + k$$

$$= T(1) + \log n$$

$$= 1 + \log n$$

Hence,

* Time complexity is $O(\log n)$.

ex: 11, 22, 30, 33, 40, 44, 55, 60, 66, 77, 80, 88, 99, k=40,

l			m									h
11	22	30	33	40	44	55	60	66	77	80	88	99
1	2	3	4	5	6	7	8	9	10	11	12	13

① let us assume l=1 key k=40

high h=n=13

then find mid value mid (m) = $\left\lfloor \frac{(l+h)}{2} \right\rfloor$

if always low lessthan one equal to high then find mid

$$mid(m) = \frac{l+h}{2}$$

if (key == mid) then 40 == 55 hence it is false

if (key < mid)

40 < 55 true

h = mid - 1;

= 7 - 1;

h = 6

now h = 6, l = 1, m = $\frac{6+1}{2} = \frac{7}{2} = 3$

if (key == mid) then

40 == 30 false

if (key < mid) then

40 < 30 false

if (key > mid) then

40 > 30 true

l = mid + 1

= 3 + 1

= 4

now l = 4, h = 6, m = $\frac{4+6}{2} = \frac{10}{2} = 5$

if (key == mid) then

40 == 40 true

the element is found.

② now key k = 62

l = 1, h = 13, k = 62

consider if (l ≤ h)

mid = $\frac{l+h}{2}$ = $\frac{1+13}{2} = \frac{14}{2} = 7$

if (key == mid) then

62 == 55 false

if (key < mid) then

62 < 55 false

if (key > mid)

62 > 55 true

l = 7 + 1 = 8

now l = 8, h = 13, m = $\frac{8+13}{2} = \frac{21}{2} = 10$

if (l ≤ h)

mid = $(\frac{8+13}{2})$ = 10

if (key == mid)

62 == 77 false

if (key < mid)

62 < 77 True

high = mid - 1

= 10 - 1

h = 9

Finding Maximum and Minimum element

* Identify low index and High index.

* calculate Mid Value based on low and high indexes that is mid $m = \left[\frac{l+h}{2}\right]$

* Based on the mid element the given list can be divided into two sub lists.

1) Left sub list

2) Right sub list

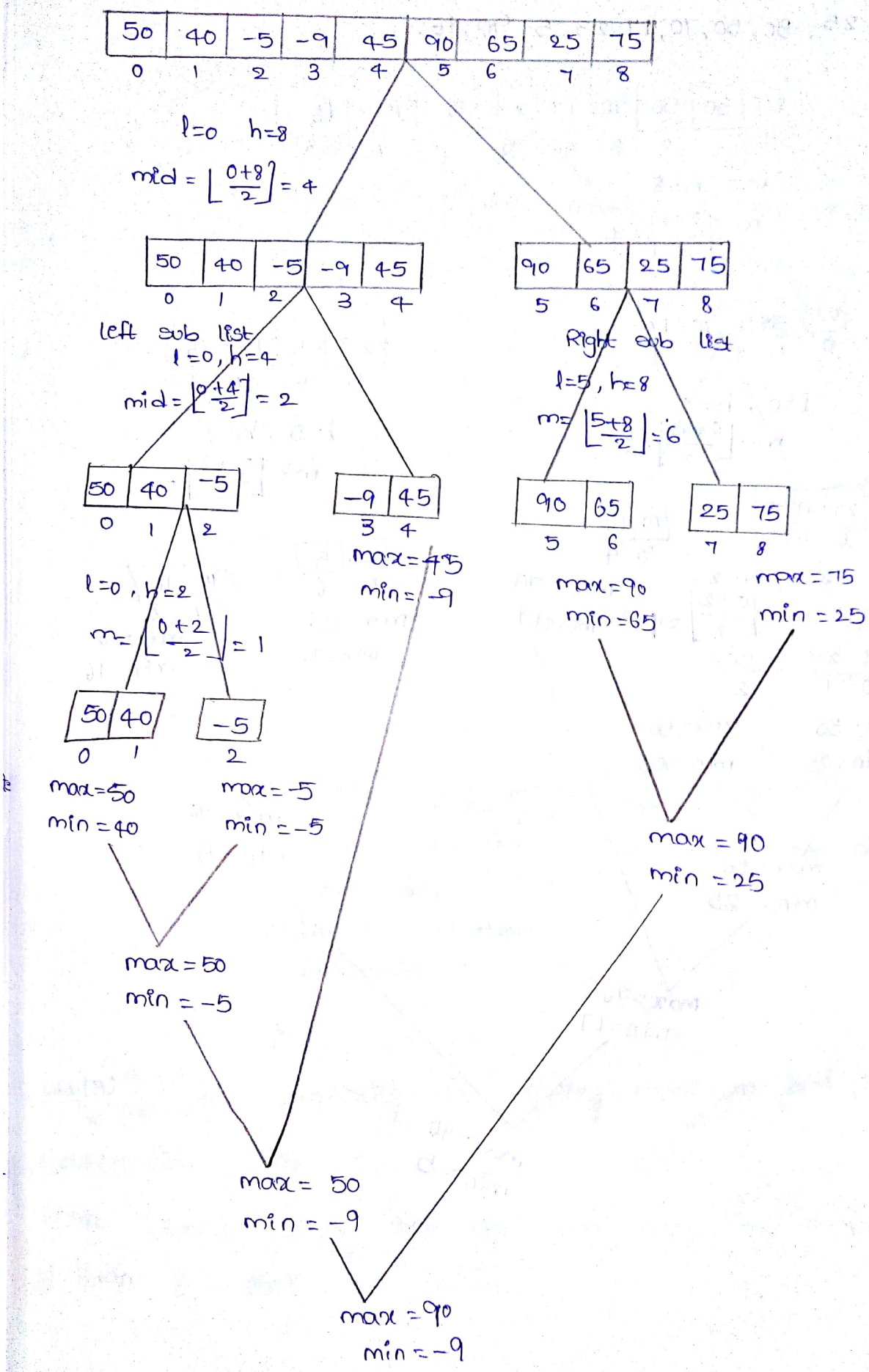
* consider left sub list, identify low, high indexes and calc mid element.

* until the left sub list contain one (or) two elements in the list.

* Consider Right sub list, identify low, high indexes and calculate mid element. until you get the list contain 1(or) 2 elements.

* compare left & Right sub lists identify maximum elem and minimum element.

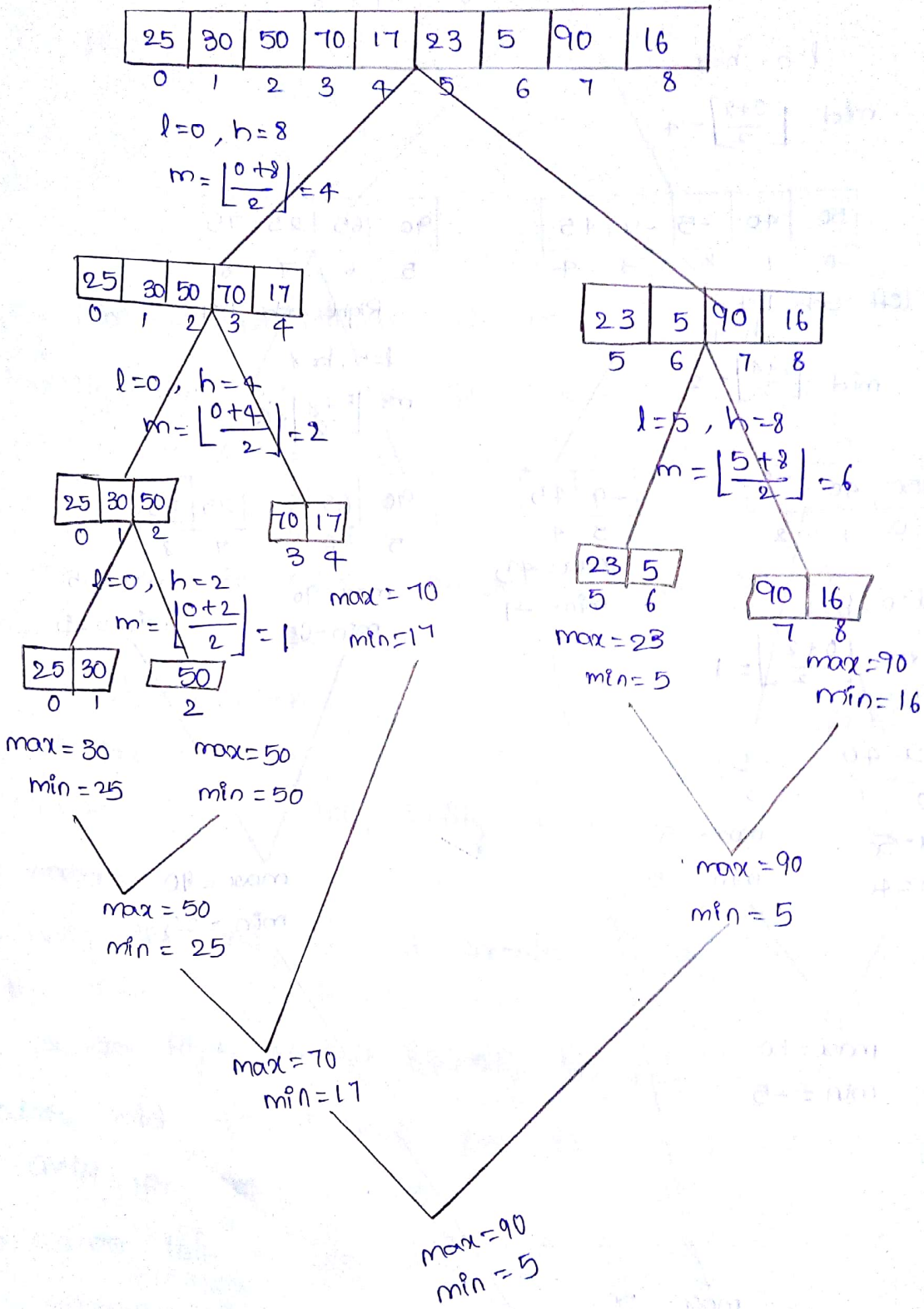
Ex:- Identify Maximum and minimum element from the following given list 50, 40, -5, -9, 45, 90, 65, 25, 75.



Hence the maximum number is 90

minimum number is -9

Ex:2 25, 30, 50, 70, 17, 23, 5, 90, 16.



```
Algorithm MaxMin(i, j, max, min)
{
 if (i = j) then
   max := min := a[i]; // list contain only one element
   else if (i = j - 1)
   {
    if (a[i] < a[j])
    {
     max := a[j];
     min := a[i];
     }
    else
      {
       max := a[i];     // list contains 2 elements
       min := a[j];
      }
    }
   else     mid = ⌊(i+j)/2⌋;   maxmin(i, mid, max, min)
      {                        maxmin(mid+1, j, max1, min1)
       if (max < max1) then    // more than one element.
       max := max1;
       if (min > min1) then
         min := min1;
      }
    }
```

Note(3)
* The time complexity for finding maximum and minimum number is $O(n)$

* The bestcase, average case and worst case time complexity for maxmin is $O(n)$.

Quick Sort:-

* In quick sort method we are going to identify i (low), j (high) and pivot.
* Always pivot = a[i]; {∵ a is array list of elements}
* i has to be incremented when a[i] ≤ pivot.
* j has to be decremented when a[j] ≥ pivot.
* Otherwise no change to be made if i & j has stop then swap a[i] & a[j].
* Then again continue the process.
* If i & j crosses each other then swap a[j], pivot.
* Finally, we can generate the list is sorted order i.e ascending order

eg: 50, 30, 10, 90, 80, 20, 40, 70.

i, j will be in same place that situation we can swap j, pivot

⇒ 50(P) 30(i) 10 90 80 20 40(j) 70

⇒ 50(P) 30 10(i) 90 80 20 40(j) 70

⇒ 50(P) 30 10 90(i) 80 20 40(j) 70

⇒ 50(P) 30 10 90(i) 80 20 40(j) 70 — swap

⇒ 50(P) 30 10 40(i) 80 20 90(j) 70

⇒ 50(P) 30 10 40 80(i) 20(j) 90 70 — swap

⇒ 50(P) 30 10 40 20(i) 80(j) 90 70

⇒ 50(P) 30 10 40 20(i) 80 90 70 — swap

⇒ 20 30 10 40 [50] 80 90 70

sublist1 < pivot sublist2 > pivot

⇒ 20(Pi) 30 10 40(j) [50] 80 90 70

⇒ 20(P) 30(i) 10 40(j) [50] 80 90 70 — swap

⇒ 20 10 30 40 [50] 80 90 70
P i j

⇒ 20 10 30 40 [50] 80 90 70
P swap j i

⇒ 10 20 30 40 [50] 80 90 70
P i j

⇒ 10 20 30 40 [50] 80 90 70
P i swap j

⇒ 10 20 30 40 [50] 80 70 90
P i j

⇒ 10 20 30 40 [50] 80 70 90
P swap i j

⇒ 10 20 30 40 [50] 70 80 90

ex:- 62, 71, 72, 80, 82, 60, 52, 51, 42.

⇒ 62 71 72 80 82 60 52 51 42
P i j

⇒ 62 71 72 80 82 60 52 51 42
P i swap j

⇒ 62 42 72 80 82 60 52 51 71
P i j

⇒ 62 42 72 80 82 60 52 51 71
P i swap j

⇒ 62 42 51 80 82 60 52 72 71
P i j

⇒ 62 42 51 80 82 60 52 72 71
P i swap j

⇒ 62 42 51 52 82 60 80 72 71
P i swap j

⇒ 62 42 51 52 60 82 80 72 71
P swap j i

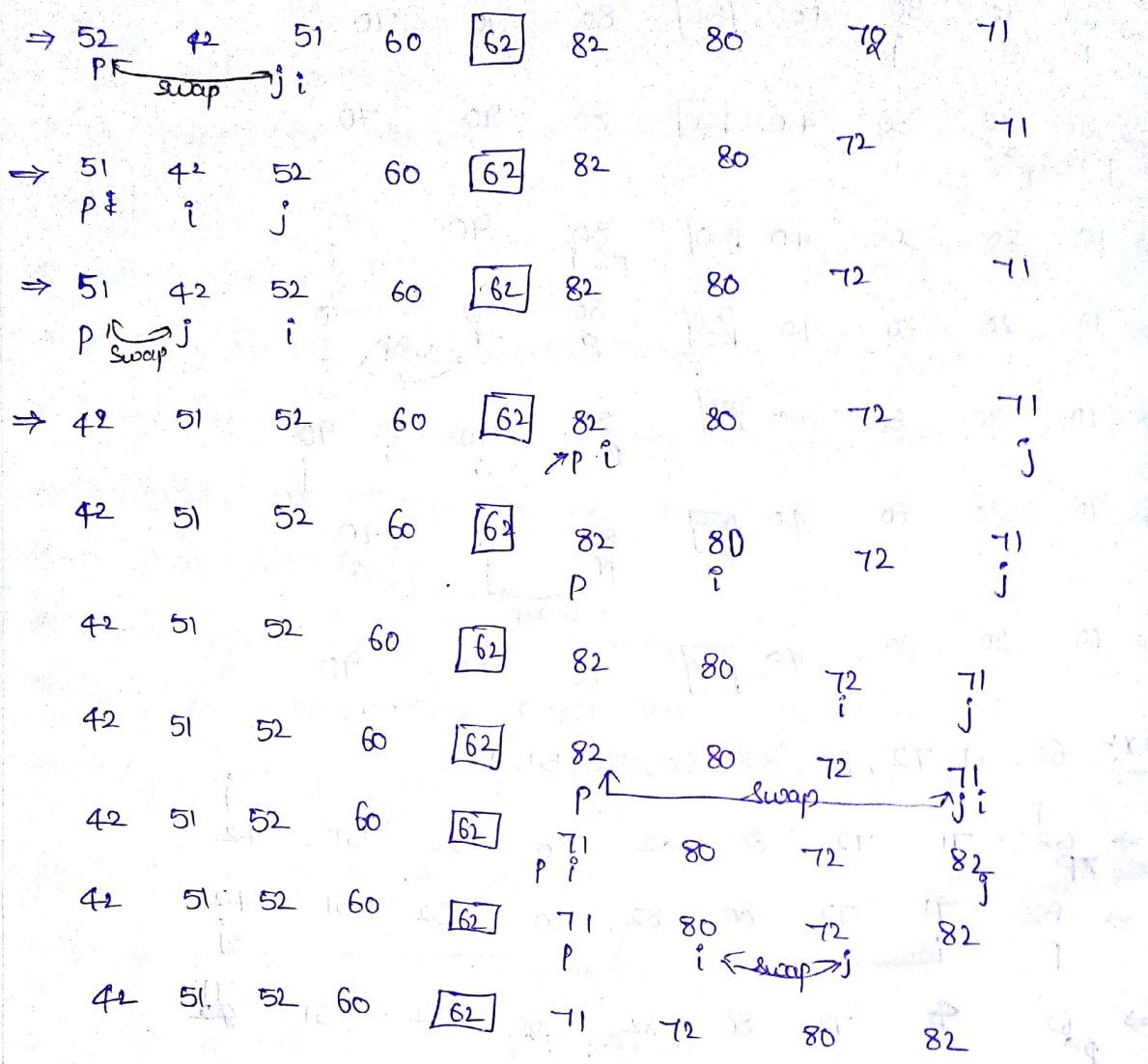
⇒ 60 42 51 52 [62] 82 80 72 71
P i j
sublist1 < pivot sublist2 > pivot

⇒ 60 42 51 52 [62] 82 80 72 71
P i j

⇒ 60 42 51 52 [62] 82 80 72 71
P i j

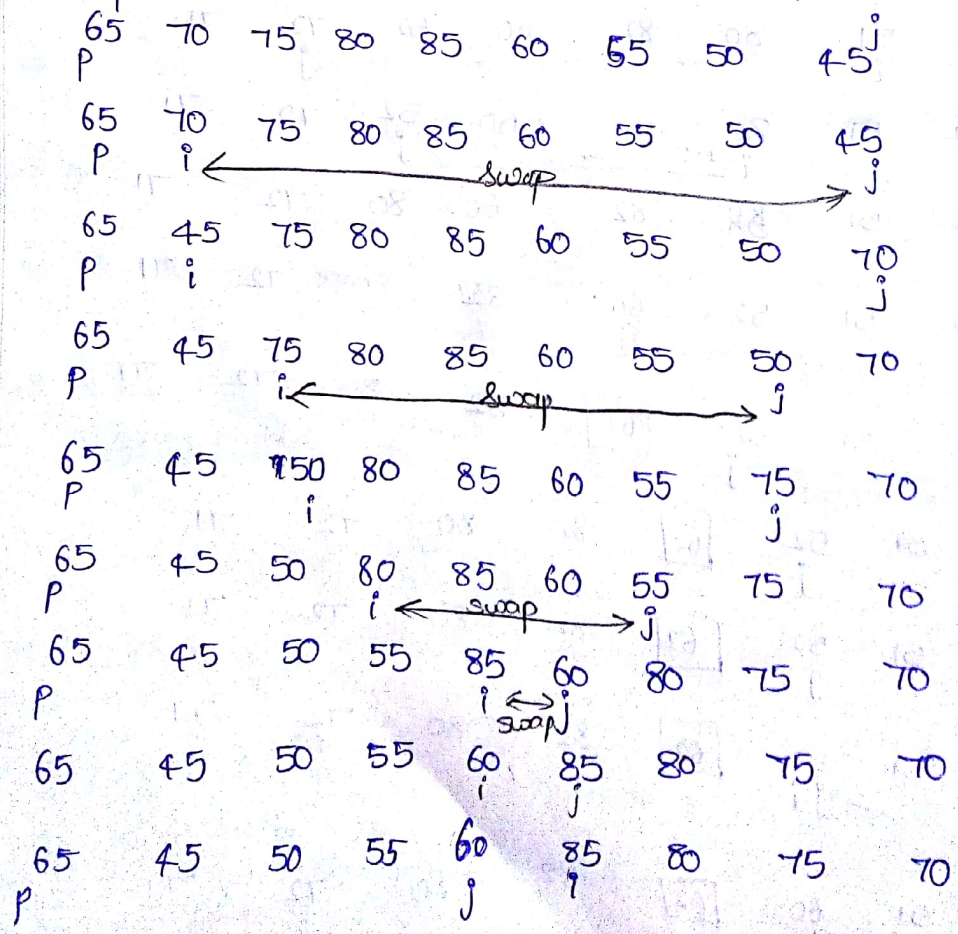
⇒ 60 42 51 52 [62] 82 80 72 71
P swap j i

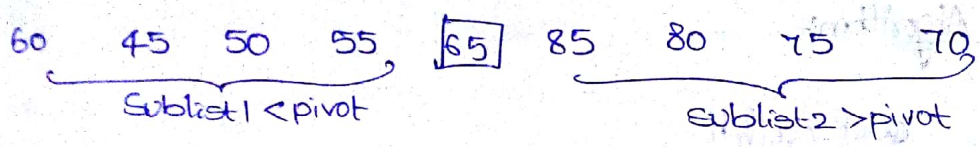
⇒ 52 42 51 60 [62] 82 80 72 71
P i j

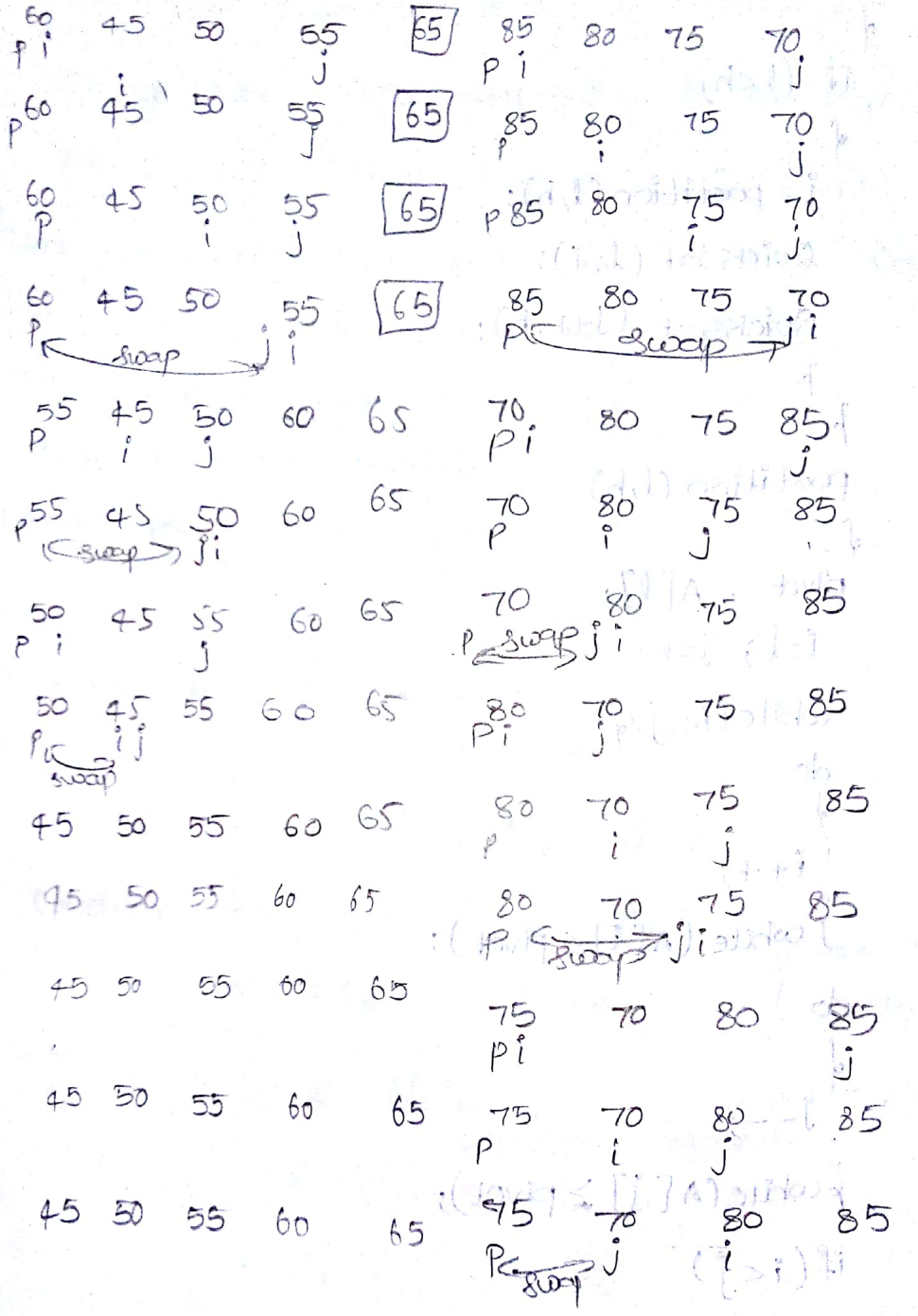


eg: 65, 70, 75, 80, 85, 60, 55, 50, 45.

a[i] ≤ pivot
a[j] ≥ pivot







45 50 55 60 65 70 75 80 85

10/12/18

Quick Sort Algorithm:-

```
QuickSort(l,h)
{
 if (l<h)
 {
  j = partition(l,h);
  QuickSort(l, j);
  QuickSort (j+1, h);
  }
}
partition (l,h)
{
 pivot = A[l];
  i=l; j=h;
 while(i<j){
 do
 {
  i++;
 }while (A[i]≤pivot);
do
{
  j--;
}while(A[j] ≥ pivot);
if(i<j)
swap (A[i], A[j]);
}
Swap(A[l], A[j]);
return j;
}
```

* Time complexity for QuickSort:

Note:

Best Case Time complexity is $O(n \log n)$

Average Case Time complexity is $O(n \log n)$

Worst case Time complexity is $O(n^2)$

** Time complexity for Best and Average Case:

To construct left sublist i.e $T(n/2)$

To construct Right sublist i.e $T(n/2)$

Time to build partition i.e n.

Hence total Time $T(n) = T(n/2) + T(n/2) + n$

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

Now the recurrence relation is

$$T(n) = \begin{cases} 1 & \text{if } n=1 \\ 2T\left(\frac{n}{2}\right)+n & \text{if } n>1 \end{cases}$$

Consider $T(n) = 2T(n/2) + n$ — (1)

$T(n) = 2\left[2T\left(\frac{n}{2^2}\right) + \frac{n}{2}\right] + n$

$= 2^2 T\left(\frac{n}{2^2}\right) + n + n$

$= 2^2 T\left(\frac{n}{2^2}\right) + 2n$ — (2)

$= 2^3 T\left(\frac{n}{2^3}\right) + 3n$ — (3)

$\vdots$

$= 2^K T\left(\frac{n}{2^K}\right) + Kn$ — (4)

{ Since $T(n) = 2T\left(\frac{n}{2}\right) + n$

$T\left(\frac{n}{2}\right) = 2 \cdot T\left(\frac{n}{2^2}\right) + \frac{n}{2}$

$T\left(\frac{n}{2^2}\right) = 2^3 T\left(\frac{n}{2^3}\right) + \frac{n}{2}$

Assume $\frac{n}{2^K} = 1$

$\therefore n = 2^K \Rightarrow K = \log_2 n$

$T(n) = n \cdot T\left(\frac{n}{n}\right) + \log_2 n \cdot n$

$T(n) = n \cdot T(1) + n \log_2 n$

$= n + n \log_2 n$

$= O(n \log_2 n)$

Time complexity for Worst Case:-

The recurrence relation $T(n) = \begin{cases} 1 & \text{if } n=0 \\ T(n-1)+n & \text{if } n>0 \end{cases}$

$T(n) = T(n-1) + n$ — ①

$T(n) = T(n-2) + (n-1) + n$ — ②

$T(n-3) + (n-2) + (n-1) + n$ — ③

$T(n-4) + (n-3) + (n-2) + (n-1) + n$ — ④

let assume $n-k=0$

$= T(n-k) + (n-(k-1)) + (n-(k-2)) + \cdots + (n-1) + n$

$= T(n-k) + (n-k+1) + (n-k+2) + \cdots + (n-1) + n$

$= T(n-n) + (n-n+1) + (n-n+2) \cdots + (n-1) + n$

$= T(0) + 1 + 2 + \cdots (n-1) + n$

$= T(0) + \frac{n(n+1)}{2}$

$= 1 + \frac{n(n+1)}{2}$

$= O(n^2)$

Note:-

Select pivot as middle element

Select Random element as pivot

** MERGE SORT:-

In merge sort based on the middle element the list can be divided into left sublist and Right sublist.

Again consider left sublist findout mid value again it can be divide into left sublist 1 & Right sublist 1, untill we get 1 element in the list.

Then combine (apply merge sort) all the elements

In the same way right sublist also divide the list untill we get 1 element in the list.

Finally combined left & Right sort list.

Eg:- | 20 | 50 | 30 | 10 | 40 | 60 |

Consider the list, Now find out middle element $m = \frac{l+h}{2}$

$\frac{1+6}{2} = 3.5 = 3$

The list 20, 50 & 30 | 20 | 50 | 30 | (1 2 3)

find out mid value $m = \left\lfloor \frac{l+h}{2} \right\rfloor = \frac{1+3}{2} = \frac{4}{2} = 2$

Now the mid element is 2

Hence | 20 | 50 | (1 2) | 30 | (3)

Now again find out mid element $m = \left\lfloor \frac{l+h}{2} \right\rfloor = \frac{1+2}{2} = 1$

The final elements are | 20 | (1) | 50 | (2) | 30 | (3)

Now combine first two elements, hence the list is | 20 | 50 |.

Now combine index 1, 2 and 3 | 20 | 30 | 50 | (1 2 3).

The list | 10 | 40 | 60 | (4 5 6)

find out middle value $m = \left\lfloor \frac{l+h}{2} \right\rfloor = \frac{4+6}{2} = \frac{10}{2} = 5$

Now the mid element is 5

Hence | 10 | 40 | (4 5) | 60 | (6)

Now again find out mid element $m = \left\lfloor \frac{l+h}{2} \right\rfloor = \left\lfloor \frac{4+5}{2} \right\rfloor = \frac{9}{2} = 4$

the final elements are | 10 | (4) | 40 | (5) | 60 | (6)

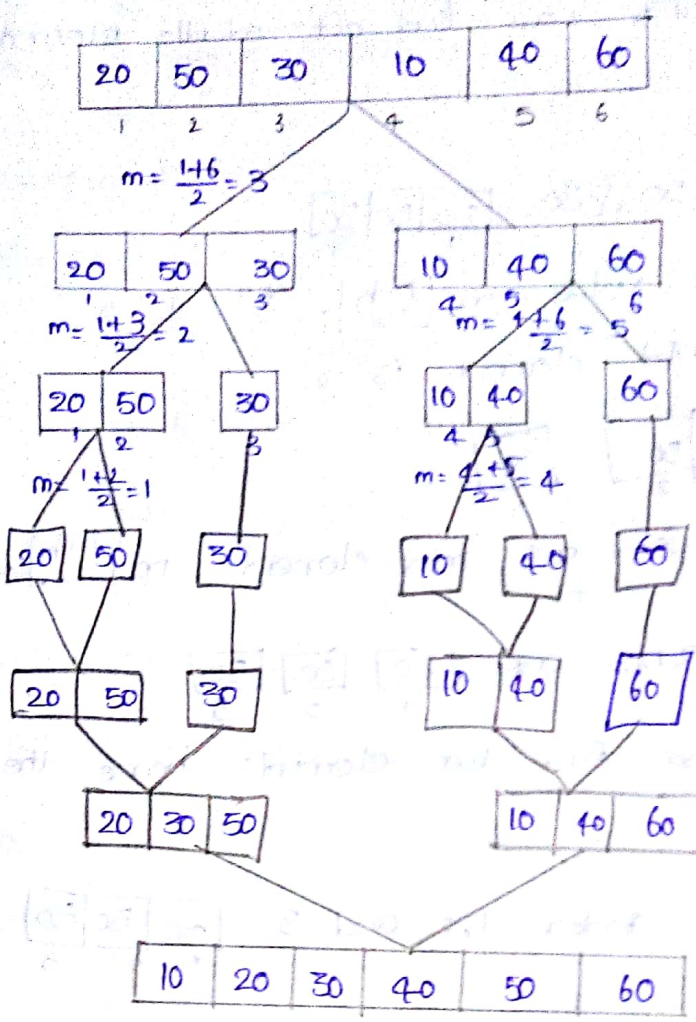
Now combine first two elements hence the list is | 10 | 40 | (4 5)

Now combine index 4, 5 and 6 | 10 | 40 | 60 | (4 5 6)

Now combine the indexes [1 → 3] and [4 → 6]

| 20 | 30 | 50 | (1 2 3) | 10 | 40 | 60 | (4 5 6)

Hence the final sorted list is | 10 | 20 | 30 | 40 | 50 | 60 |



eg: 65, 70, 60, 75, 85, 80, 55, 50, 45, 57

consider the list

65	70	60	75	85	80	55	50	45	57
1	2	3	4	5	6	7	8	9	10

Find out middle element $m = \frac{l+h}{2} = \frac{1+10}{2} = \frac{11}{2} = 5$

the list

65	70	60	75	85
1	2	3	4	5

Find out mid value $m = \frac{l+h}{2} = \frac{1+5}{2} = 3$

Now the mid element is 3

Hence

65	70	60
1	2	3

75	85
4	5

Now again find out mid element $m = \frac{1+3}{2} = 2$.

65	70
1	2

60
3

75	85
4	5

Now again find out mid element $= \frac{1+2}{2} = 1$

65	70	60	75	85
1	2	3	4	5

Now again find out mid value $m = \frac{4+5}{2} = \frac{9}{2} = 4$

the mid element is 4

65	70	60	75	80
1	2	3	4	5

Now combine first two elements hence the list is

65	70
1	2

Now combine indexes 1, 2 & 3 | 60 | 65 | 70 |

Now combine indexes 1, 2, 3 & 4 | 60 | 65 | 70 | 75 |

Now combine indexes 1, 2, 3, 4 & 5

60	65	70	75	80
1	2	3	4	5

the list

80	55	50	45	57
6	7	8	9	10

Find out middle value $m = \frac{l+h}{2} = \frac{6+10}{2} = \frac{16}{2} = 8$

Now the middle element is 8, hence | 80 | 55 | 50 | | 45 | 57 |

Now again find out mid element $m = \frac{6+8}{2} = \frac{14}{2} = 7$

Now the middle element is 7, hence | 80 | 55 | | 50 | | 45 | 57 |

again find out middle element $m = \frac{6+7}{2} = \frac{13}{2} = 6$

Now the mid value is 6, | 80 | | 55 | | 50 | | 45 | 57 |

Again find out middle value $m = \frac{9+10}{2} = \frac{19}{2} = 9$

Now the mid value is 9, | 80 | | 55 | | 50 | | 45 | | 57 |

Now combine first two elements hence the list is

55	80
6	7

Now combine indexes 6, 7 & 8 | 50 | 55 | 80 |

Now combine indexes 6, 7, 8 & 9 | 45 | 50 | 55 | 80 |

Now combine indexes 6, 7, 8, 9 & 10 | 45 | 50 | 55 | 57 | 80 |

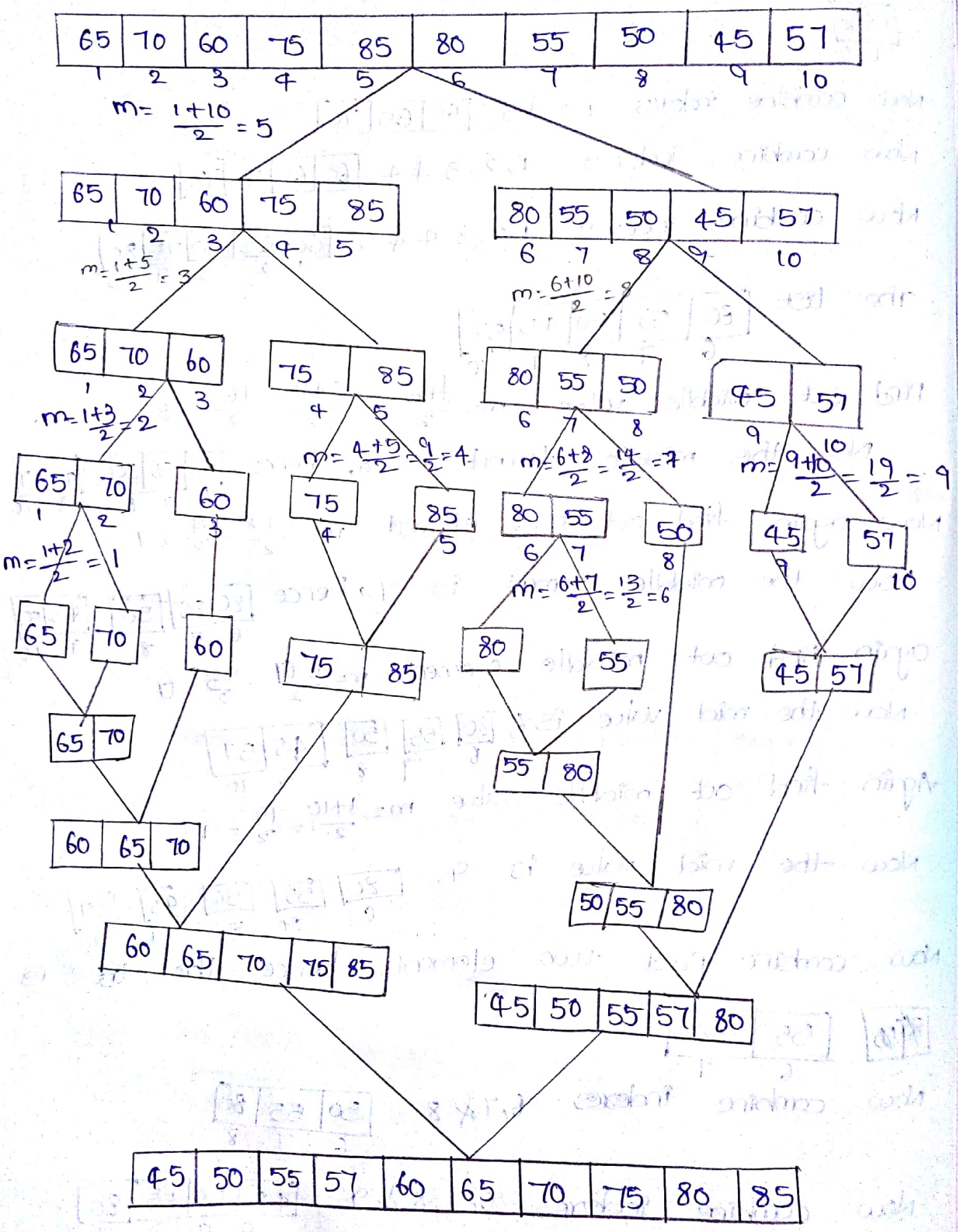
Now combine indexes [1-5] & [6-10]

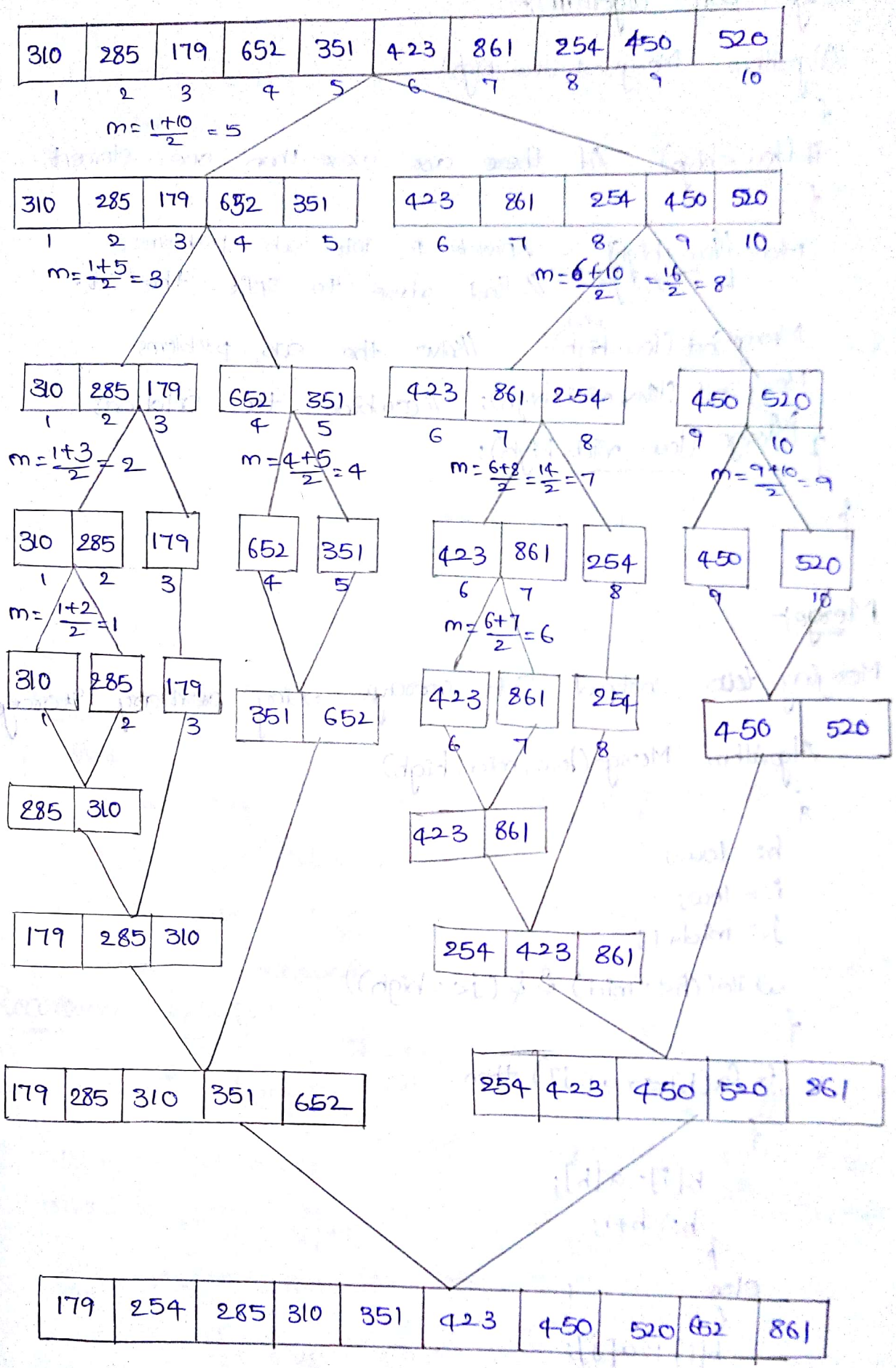
60	65	70	75	85
1	2	3	4	5

45	50	55	57	80
6	7	8	9	10

Hence the final sorted list is

45	50	55	57	60	65	70	75	80	85
1	2	3	4	5	6	7	8	9	10




310 285 179 652 351 423 861 254 450 520
1 2 3 4 5 6 7 8 9 10
m = 1+10/2 = 5
310 285 179 652 351
1 2 3 4 5
m = 1+5/2 = 3
423 861 254 450 520
6 7 8 9 10
m = 6+10/2 = 16/2 = 8
310 285 179
1 2 3
m = 1+3/2 = 2
652 351
4 5
m = 4+5/2 = 4
423 861 254
6 7 8
m = 6+8/2 = 14/2 = 7
450 520
9 10
m = 9+10/2 = 9
310 285
1 2
m = 1+2/2 = 1
179
3
652 351
4 5
423 861
6 7
m = 6+7/2 = 6
254
8
450 520
9 10
310 285
1 2
179
3
351 652
423 861
6 7
254
8
450 520
285 310
423 861
179 285 310
254 423 861
179 285 310 351 652
254 423 450 520 861
179 254 285 310 351 423 450 520 652 861

Algorithm:-

Merge Sort algorithm:-

```
Algorithm MergeSort (low, high)
{
if (low<high)  // if there are more than one element
{
 mid = ⌊(low+high)/2⌋;  // Divide P into sub problems
                        // -find where to split the set.
 MergeSort (low, mid, high);  // Solve the sub problems
 Mergesort (mid+1, high);  // combine the solutions
 Merge (low, mid, high);
}
}
```

Merge:-

Merging two sorted sub arrays using auxilary storage

```
Algorithm Merge(low, mid, high)
{
  h:= low;
  i:= low;
  j:= mid+1;
  while((h<=mid) && (j<=high))
  {
   if (a[h]<=a[j]) then
    {
      b[i]= a[h];
      h:= h+1;
    }
   else
    {
     b[i] :=a[j];
     j:=j+1;
    }
    i:=i+1;
  }
```

```
for k:= j to high do
{
  b[i]:=a[k];
   i:= i+1;
  }
else
  {
  for k:= j to mid do
  {
   b[i]:=a[k];
     i=i+1;
  }
  for k:=low to high do
     a[k]:= b[k];
  }
```

storage

Time Complexity:-

Note:- In merge sort the time complexity is in Best

Best case $O(n\log n)$

Average Case $O(n\log n)$

Worst Case $O(n\log n)$

Recurrence Relation:-

$$T(n) = \begin{cases} 1 & \text{if } n=1 \\ 2T(\frac{n}{2})+n & \text{if } n>1 \end{cases}$$

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

$$T(n) = 2\left[2T\left(\frac{n}{2^2}\right)+\frac{n}{2}\right]+n$$

$$= 2^2T\left(\frac{n}{2^2}\right)+n+n$$

$$= 2^2T\left(\frac{n}{2^3}\right)+2n \quad \text{---(2)}$$

$$= 2^3T\left(\frac{n}{2^3}\right)+3n \quad \text{---(3)}$$

$$\vdots$$

$$= 2^kT\left(\frac{n}{2^k}\right)+kn \quad \text{---(4)}$$

$\because T(n) = 2T(\frac{n}{2})+n$

$T(\frac{n}{2}) = 2T(\frac{n}{2^2})+\frac{n}{2}$

$T(\frac{n}{2^2}) = 2^2T(\frac{n}{2^3})+\frac{n}{2}$

Assume $\frac{n}{2^k} = 1 \Rightarrow n = 2^k$

$k = \log_2 n$

$$T(n) = n \cdot T\left(\frac{n}{n}\right) + \log_2 n \cdot n$$

$$= n\,T(1) + n\log_2 n$$

$$= n + n\log_2 n$$

$$= O(n\log_2 n)$$

Strassen's Matrix Multiplication:-

* In general matrix multiplication is the no.of columns in the 1st matrix = the no.of rows in the 2nd matrix.

* Let us consider $A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ $B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$

* the resultant matrix $C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}$ where

$$c_{11} = a_{11} * b_{11} + a_{12} * b_{21}$$

$$c_{12} = a_{11} * b_{12} + a_{12} * b_{22}$$

$$c_{21} = a_{21} * b_{11} + a_{22} * b_{21}$$

$$c_{22} = a_{21} * b_{12} + a_{22} * b_{22}.$$

* Here we have to compute 8 multiplications hence the time complexity of matrix multiplication is $O(n^3)$

* In case of 4X4 matrix we are going to compute large no.of multiplications needed.

* Let us assume $A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$ and

$$B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix}$$

* Now the above matrix can be divided into 4 equal 2X2 matrices

$$A = \left(\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right) \quad B = \left(\begin{array}{cc|cc} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ \hline b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{array}\right)$$

A_{11} A_{12} A_{21} A_{22} — B_{11} B_{12} B_{21} B_{22}

* Now finally $C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$

* The recurrence relation for general matrix multiplication is

$$T(n) = \begin{cases} 1 & \text{if } n \le 2 \\ 8T(n/2) + n^2 & \text{if } n > 2 \end{cases}$$

* Now the strassen's reduce the time complexity is

$$T(n) = \begin{cases} 1 & \text{if } n \le 2 \\ 7T(n/2) + n^2 & \text{if } n > 2 \end{cases}$$

* Instead of using 8 multifications strassen's matrix multifications use only 7 multifications.

* Hence the time complexity is $O(n^{2.81})$

* Strassen's Introduce 7 different formula's are

$$P = (A_{11} + A_{22}) \cdot (B_{11} + B_{22})$$
$$Q = B_{11} (A_{21} + A_{22})$$
$$R = A_{11} (B_{12} - B_{22})$$
$$S = A_{22} (B_{21} - B_{11})$$
$$T = B_{22} (A_{11} + A_{12})$$
$$U = (A_{21} - A_{11}) \cdot (B_{11} + B_{12})$$
$$V = (B_{21} + B_{22}) \cdot (A_{12} - A_{22})$$

$\therefore \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$

T(+)

S(-) 11 12 21 22 R(-)

Q(+)

B_{11} A_{11} A_{22} B_{22}

$C_{11} = P+S-T+V$

$C_{12} = R+T$

$C_{21} = Q+S$

$C_{22} = P+R-Q+U$

RAT
Gadd

Eg:-

$A = \begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix}$ $B = \begin{pmatrix} 6 & 7 \\ 3 & 8 \end{pmatrix}$ find C

Here consider $A_{11} = 1$, $A_{12} = 3$, $A_{21} = 7$, $A_{22} = 5$

$B_{11} = 6$ $B_{12} = 7$ $B_{21} = 3$ $B_{22} = 8$

$P = (A_{11} + A_{22}) \cdot (B_{11} + B_{22})$

$= (1+5) \cdot (6+8) = (6) \cdot (14) = 84$

$Q = B_{11} \cdot (A_{21} + A_{22})$

$= 6(7+5)$

$= 6(12)$

$= 72$

$R = A_{11} \cdot (B_{12} - B_{22})$

$= 1(7-8) = -1$

$S = A_{22}(B_{21} - B_{11})$

$= 5(3-6) = 5(-3) = -15$

$T = B_{22}(A_{11} + A_{12})$

$= 8(1+3) = 8(4) = 32$

$U = (A_{21} - A_{11})(B_{11} + B_{12})$

$= (7-1)(6+7) = (6)(13) = 78$

$V = (B_{21} + B_{22}) \cdot (A_{12} - A_{22})$

$= (3+8)(3-5) = (11)(-2) = -22$

$C_{11} = P+S-T+V$

$= 84 + (-15) - 32 + (-22)$

$= 84 - 15 - 32 - 22$

$= 15$

$C_{12} = R+T = -1 + 32 = 31$

$C_{21} = Q+S = 72 + (-15) = 57$

$C_{22} = P+R-Q+U$

$= 84 + (-1) - 72 + 78$

$= 89$

Hence the resultant matrix $C = \begin{pmatrix} 15 & 31 \\ 57 & 89 \end{pmatrix}$

Eg:- $A = \begin{pmatrix} 4 & 5 \\ 6 & 9 \end{pmatrix}$ $B = \begin{pmatrix} 12 & 20 \\ 3 & 14 \end{pmatrix}$ Then find $A \times B = C$.

Here consider $A_{11}=4$ $A_{12}=5$ $A_{21}=6$ $A_{22}=9$

$B_{11}=12$ $B_{12}=20$ $B_{21}=3$ $B_{22}=14$

$P = (A_{11}+A_{22}) \cdot (B_{11}+B_{22})$

$= (4+9)(12+14) = (13)(26) = 338$

$Q = B_{11} \cdot (A_{21}+A_{22}) = 12(6+9) = 12(15) = 180$

$R = A_{11} \cdot (B_{12}-B_{22}) = 4(20-14) = 4(6) = 24$

$S = A_{22}(B_{21}-B_{11}) = 9(3-12) = 9(-9) = -81$

$T = B_{22}(A_{11}+A_{12}) = 14(4+5) = 14(9) = 126$

$U = (A_{21}-A_{11})(B_{11}+B_{12}) = (6-4)(12+20) = (2)(32) = 64$

$V = (B_{21}+B_{22})(A_{12}-A_{22}) = (3+14)(5-9) = (17)(-4) = -68$

$C_{11} = P+S-T+V$

$= 338 - 81 - 126 - 68 = 338 - 275 = 63$

$C_{12} = R+T = 24+126 = 150$

$C_{21} = Q+S = 180 - 81 = 99$

$C_{22} = P+R-Q+U = 338 + 24 - 180 + 64 = 394 - 180 = 246$

Hence the resultant matrix $C = \begin{pmatrix} 63 & 150 \\ 99 & 246 \end{pmatrix}$

Algorithm:

```
Algorithm MatrixMul (A, B, n)
{
 if(n<=2)
 {
 C = 4 formula's
 }
 else
 {
 mid ... n/2
 MM(A11, B11, n/2) + MM(A12, B21, n/2);
 MM(A11, B12, n/2) + MM(A12, B22, n/2);
 Mm(A21, B11, n/2) + Mm(A22, B21, n/2);
 Mm(A21, B12, n/2) + Mm(A22, B22, n/2);
 }
}
```

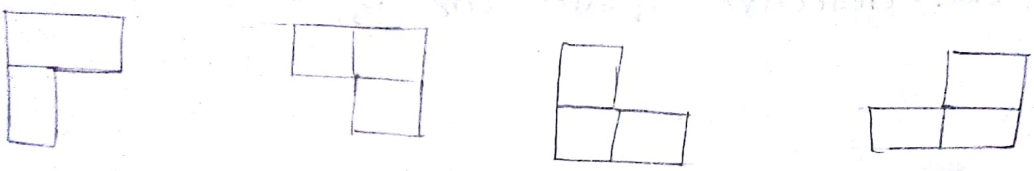
Time complexity:- $O(n^{2.81})$

24/12/18.

* Defective chess board Algorithm problem :-

→ Given chessboard of size nxn i.e $n = 2^k$.

A chessboard contain exactly one defective i.e indicated by Square, (or) add to fill chessboard with help of L-shaped tiles. These L-shaped tails are called Triamines.



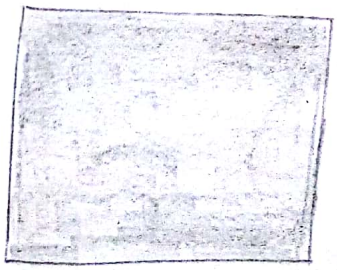
→ In a chessboard total no. of Squares = nxn, where $n = 2^k \Rightarrow$ i.e $2^k x 2^k \Rightarrow 2^{2k}$.

→ In a chsbrd, no. of non-defective squares = $2^{2k} - 1$

→ This number is always divisible by 3.

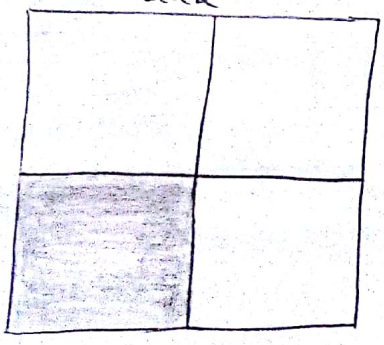
→ In 1x1 chsbrd only one defective square.

→ In 1x1 chsbrd the no. of Non-defective chsboard is 0.

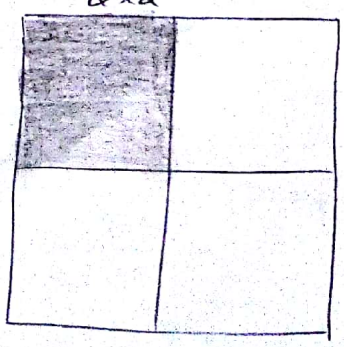


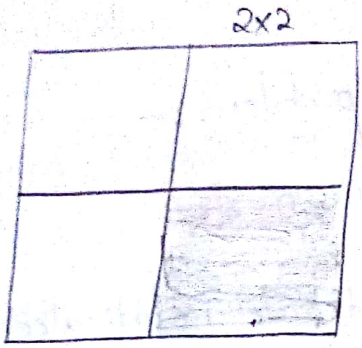
→ In 2x2 chsboard, exactly one defective square & remaining three are comes under to Non-defective Squares.

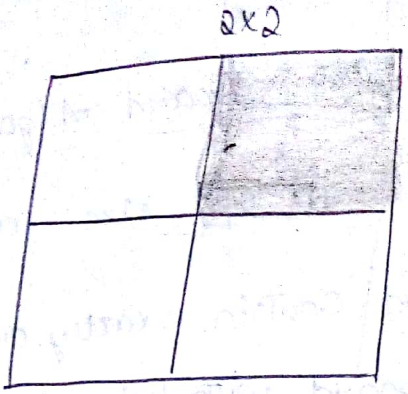
2x2



2x2

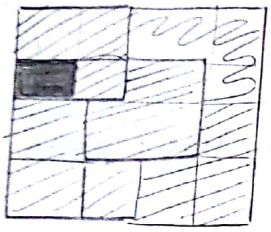


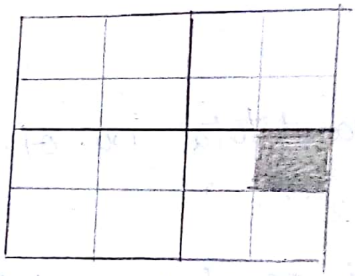




→ In 4x4 chessboard, exactly one defective square.

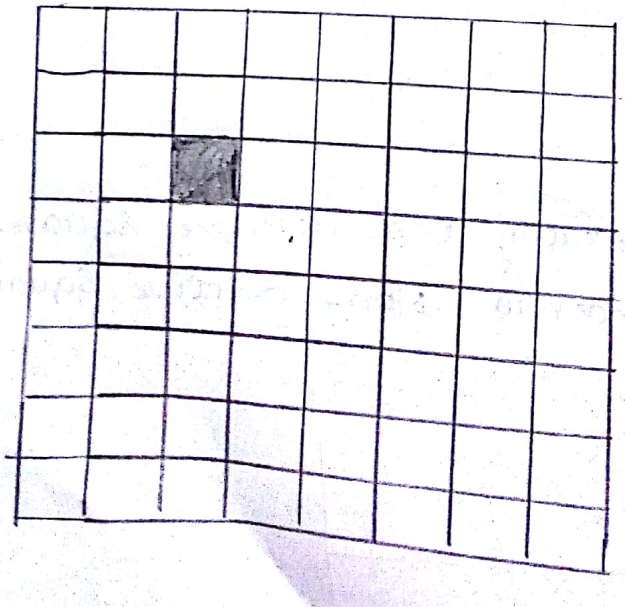
→ The no. of non-defective squares are 15.





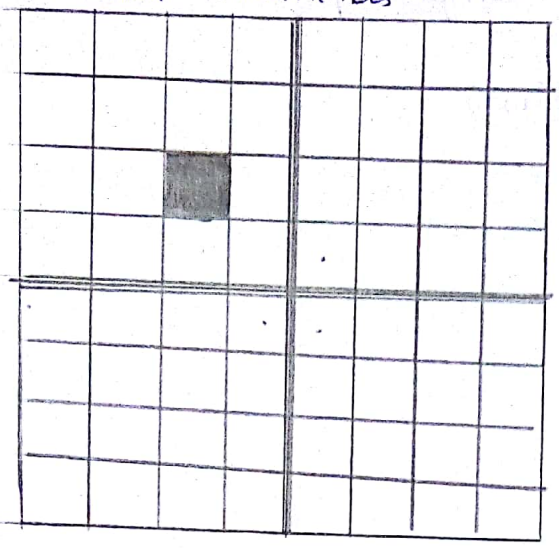
→ size k=1, i.e 2x2 problem has a simple solution, if the problem is very large then it can be divided into smaller ones finally combine solutions to smaller problems.

Ex:- Tilling 8x8 Defective chessboard.



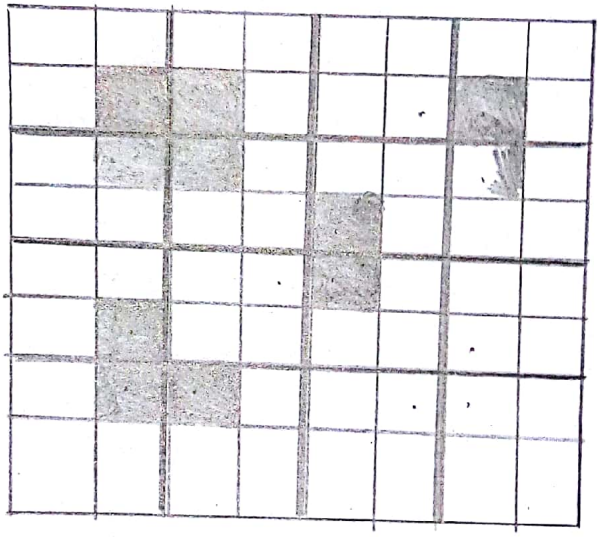
→ Now, the 8x8 chessboard can be divided into 4x4 chessboard.

→ Now, introduce, artificial traiminoes.



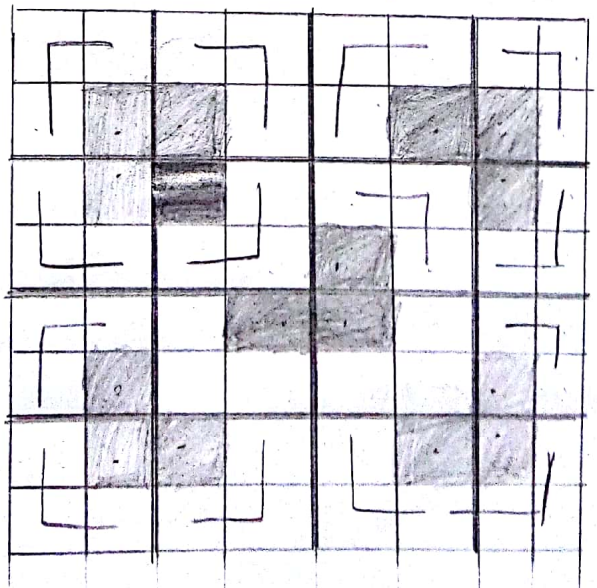
4x4

→ Now, again each 4x4 can be divided into possible no. of 2x2 chessbrd



→ Now add triaminos for each & every 2x2 chessboard.

→ Now Combine all the solutions



Note :- The Time Complexity the detective chessboard is $O(n^2)$ i.e, $\Theta(n^2)$