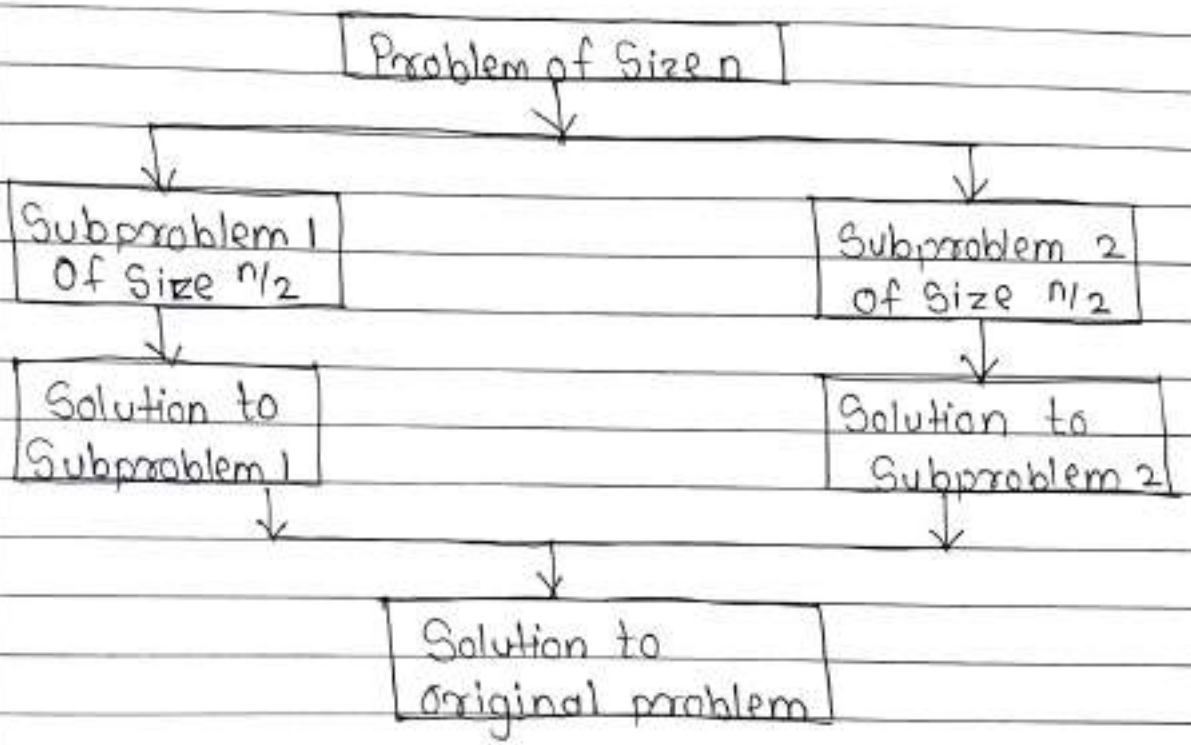
Date 30/01/25
Page

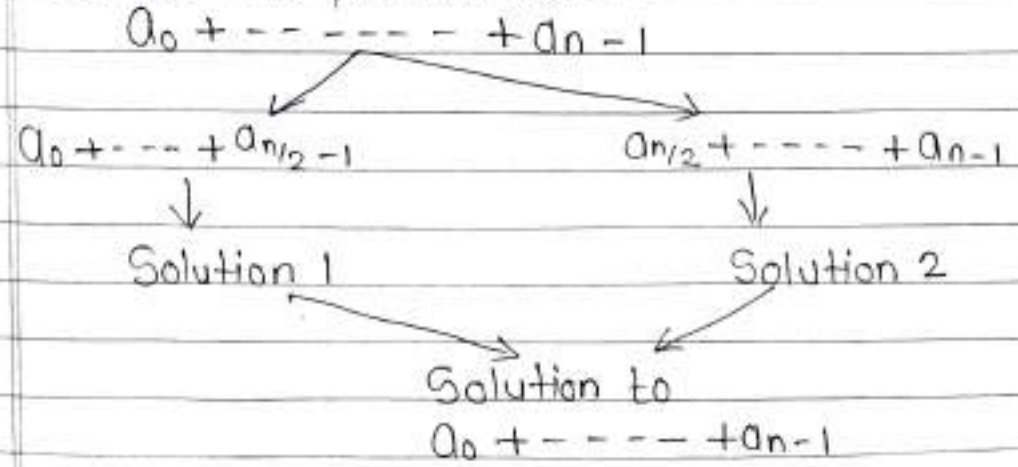
Module - II (Divide & conquer approach)

* Quick Sort :-

- Problem is divided into two sub-problems each sub-problem is sort independently them Combine the solutions of both problems into the solution of whole.



Suppose we Count Sum of n numbersthen by using divide & Conquer approach we can Solved the problem as



* Quick Sort :-

- Array is divide into two parts left sub list and right sub list.
- left sub list Contains the elements which are less than or equal to pivot.
- RSL Contains the elements which are greater than pivot.

Qe 1] Sort the following array by using quick Sort.

50 30 10 90 80 20 40 70

→

- 50 30 10 90 80 20 40 70
  pivot, i, j

  increment i pointer

- 50 30 10 90 80 20 40 70
  pivot, i, j

  increment i pointer

- 50 30 10 90 80 20 40 70
  pivot, i, j

  increment i pointer

- 50 30 10 90 80 20 40 70
  pivot, i, j

  Stop incrementating i, decrement j

- 50 30 10 90 80 20 40 70
  i, j

  Stop decrementing j
  Swap A[i] & A[j]

- 50 30 10 40 80 20 90 70
  i j
  increment i
- 50 30 10 40 80 20 90 70
  i j
  80 > 50 Stop increment i
  decrement j
- 50 30 10 40 80 20 90 70
  i j
  20 < 50 Stop decrementing j
  Swap A[i] & A[j]
- 50 30 10 40 20 80 90 70
  i j
  20 < 50 increment i
- 50 30 10 40 20 80 90 70
  i,j
  Stop incrementing 1, decrement j
- 50 30 10 40 20 80 90 70
  pivot j i
  J has crossed i
  So Swap A[j] & A[pivot]
- 20 30 10 40 50 80 90 70
  LSL pivot RSL

LSL:-

- 20 30 10 40
  pivot, j

  increment i pointer

- 20 30 10 40
  pivot i j

  Stop incrementing i & decrement j

- 20 30 10 40
  pivot i j

  Stop decrementing j Swap A[i] & A[j]

- 20 10 30 40
  i j

  increment i pointer

- 20 10 30 40
  i,j

  Stop incrementing i decrement j

- 20 10 30 40
  pivot j i

  j has cross i
  Swap A[j] & A[pivot]

- 10 20 30 40
  pivot

* RSL :-

80 90 70
pivot i j

increment i

80 90 70
i j

Stop incrementing i & stop decrementing j
Swap A[i] & A[j]

80 70 90
i j

increment i & decrement j

80 70 90
j i

j has crossed Swap A[j] & A[pivot]
70 80 90

• Final Sorted array
10 20 30 40 50 70 80 90.

Q.2] Sort the following array by using quick sort.
50 40 5 9 45 90 65 25 75

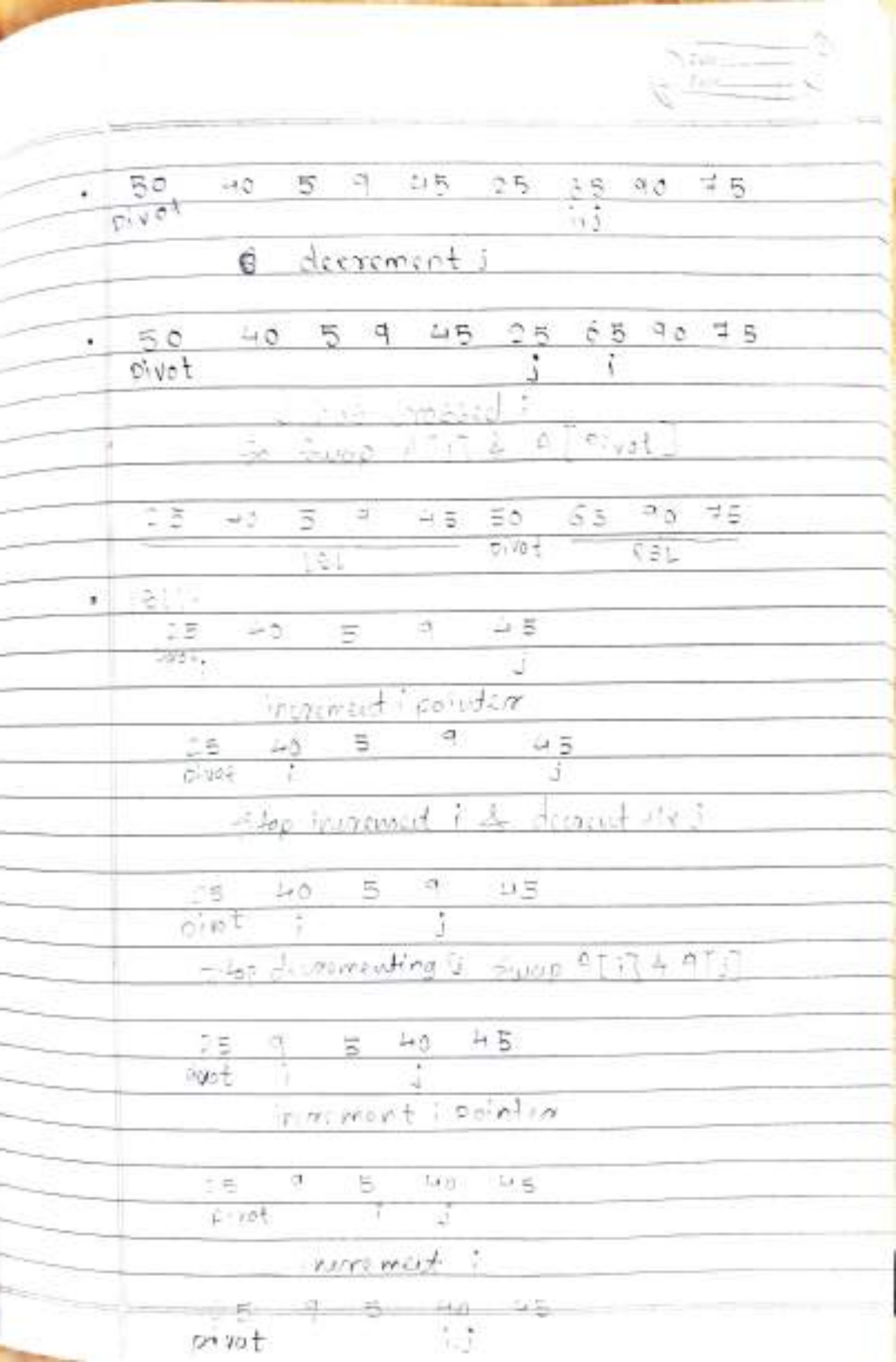
50 40 5 9 45 90 65 25 75
pivot i j

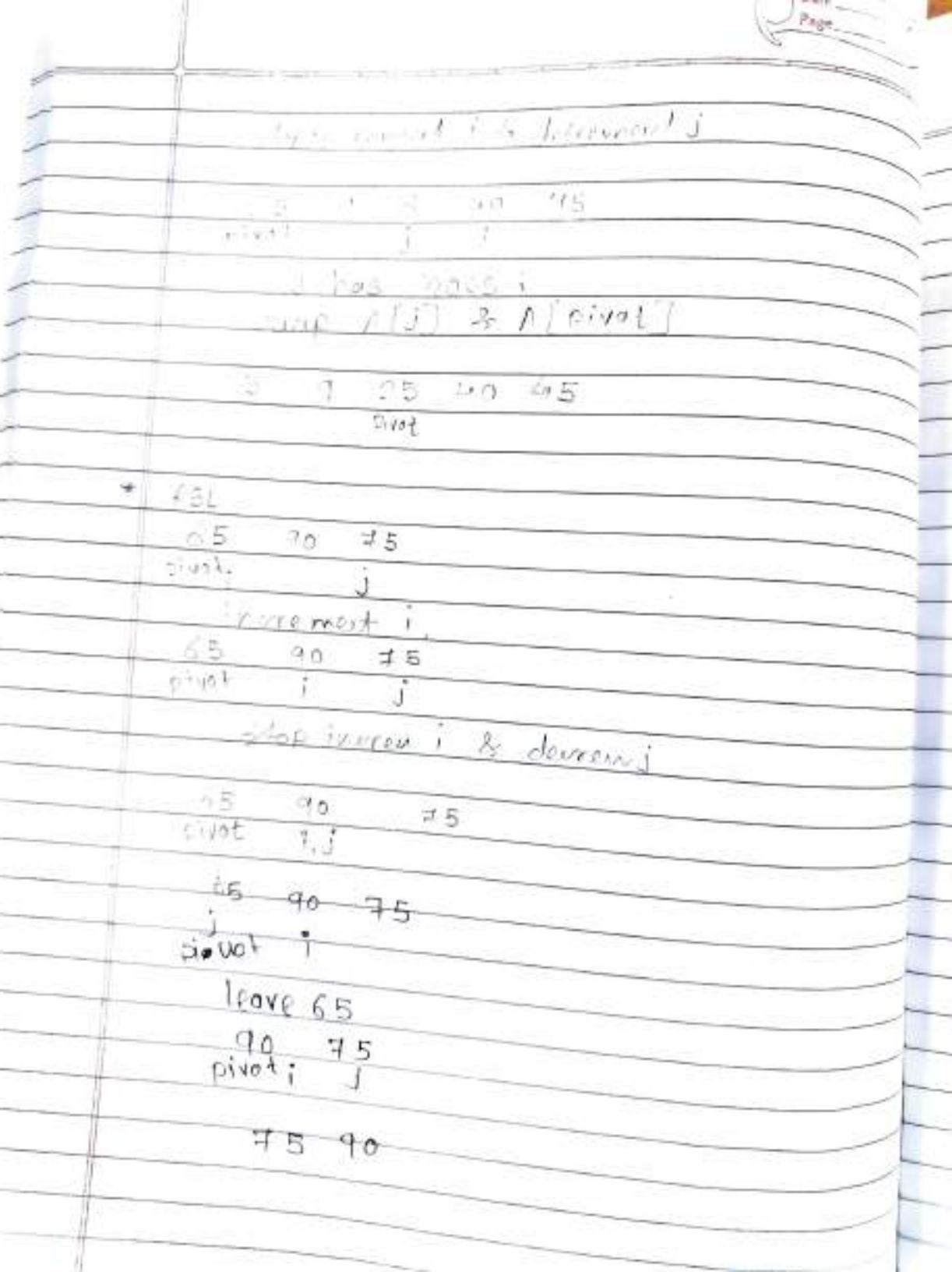
increment i pointer.

50 40 5 9 45 90 65 25 75
pivot i j

increment i pointer.

- 50 40 5 9 45 90 65 25 75
  pivot i j
  increment i pointer
- 50 40 5 9 45 90 65 25 75
  pivot i j
  increment i pointer
- 50 40 5 9 45 90 65 25 75
  pivot i j
  increment i pointer
- 50 40 5 9 45 90 65 25 75
  pivot i j
  Stop incrementing i, decrement j
- 50 40 5 9 45 90 65 25 75
  pivot i j
  Stop decrementing j
  Swap A[i] & A[j]
- 50 40 5 9 45 25 65 90 75
  i j
  increment i
- 50 40 5 9 45 25 65 90 75
  i j
  65 > 50 Stop increment i
  decrement j





Date 23/02/25
Page

* Algorithm QuickSort (A, low, high)

```
if (low < high) then
q ← partition (A, low, high)
QS (A, low, q-1)  // left Sublist
QS (A, q+1, high) // right Sublist
 end
partition (A, low, high)
   x ← A[high]
    i = low -1
for j = low to high-1 do
    if A[j] ≤ x then
       i = i+1
Swap (A(i), A(j))
  end
  end
  Swap (A(i+1), A(high))
  return (i+1)
```

eg①:-

0	1	2	3	4	5	6
10	80	30	90	40	50	70

low = 0 high = 6

0 < 6

partition (A, 0, 6)

x = A[high]

x = A[6]

x = 70

i = low - 1

i = 0 - 1 = -1

j = low to high - 1

j = 0 to 5

A[0] ≤ 70

10 ≤ 70

i = i+1 = 0

Swap (A(0), A(0))

i & j are same

No change

j = 1

A[1] ≤ x

80 ≤ 70 false

j = 2

A[2] ≤ 70

30 ≤ 70 True

i = 0+1 = 1

Swap (A(1), A(2))

Swap (80, 30)

0	1	2	3	4	5	6
10	30	80	90	40	50	70

j = 3

A[3] ≤ 70

90 ≤ 70 false

j = 4

A[4] ≤ 70

40 ≤ 70 True

i = 1+1 = 2

Swap (A(2), A(4))

Swap (80, 40)

40, 80

0	1	2	3	4	5
10	30	40	90	80	50

j = 5

A[5] ≤ 70

50 ≤ 70 True

i = 1+1

i = 2+1 = 3

Swap (A(3), A(5))

Swap (90, 50)

50, 90

0	1	2	3	4	5	6
10	30	40	50	80	90	70

Swap (A(i+1), A(high))

Swap (A(4), A(6))

Swap (80, 70)

10	30	40	50	70	90	80
				pivot		

Return (i+1)

return (4)

q = 4

QS (A, 0, 3)

10, 30, 40, 50

QS (A, q+1, high)

QS (A, 5, 6)

90 80

- Best Case Complexity:- / & Average Case:-

- Best and Avg. case complexity occurs when the list is devided in middle.

$T(n) = T(n/2) + T(n/2) + n$ → (Time taken to combine left sublist and right sublist)

↓ Time take for the left sublist to get started

↓ Time taken by the right sublist to get sorted.

$$T(n) = 2T\left(\frac{n}{2}\right) + n \quad \text{— (I)}$$

Replace n by $n/2$

$$T(n/2) = 2T\left(\frac{n}{4}\right) + \frac{n}{2} \quad \text{— (II)}$$

Put $T(n/2)$ in eqn (I)

$$T(n) = 2\left(2T\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n$$

$$= 4T\left(\frac{n}{4}\right) + n + n$$

$$T(n) = 4T(n/4) + 2n \quad \text{— (III)}$$

Replace n by $n/4$ in eqn (I) or n by $n/2$ in eq^n (II)

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + \left(\frac{n}{4}\right) \quad \text{— (IV)}$$

put $T(n/4)$ in eq^n (III)

$$T(n) = 4\left(2T\left(\frac{n}{8}\right) + \frac{n}{4}\right) + 2n$$

$$= 8T\left(\frac{n}{8}\right) + n + 2n$$

$$T(n) = 8T(n/8) + 3n$$

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

$$\vdots$$

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

$n = 2^k$

$k = \log_2 n$

$T(n) = 2^k T(1) + kn$

$T(1) = 0$ No Comparison is required when the problem of size 1

$T(n) = 0 + kn$

$T(n) = n \log_2 n$

Complexity $= O(n \log_2 n)$

* Worst Case :-

Q.1]

- Worst Case occurs when the list is already Sorted.
- So Complexity of quick Sort is same as that of Selection Sort.
- After each iteration one element get ~~Sorted~~ and problem size is reduced by 1 So $T(n) = T(n-1) + n$ and inner loop iterates for n times.

$T(n) = T(n-1) + n$ —①

Replace n by $n-1$

$T(n-1) = T(n-2) + (n-1)$ — (ii)

put $T(n-1)$ in eqn (i)

$T(n) = T(n-2) + (n-1) + n$ — (iii)

Replace n by $n-2$ in eqn (i) or $n-1$ in eqn (ii)

$T(n-2) = T(n-3) + (n-2)$

$T(n) = T(n-3) + (n-2) + (n-1) + n$

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

put $n = k$

$T(n) = 1 + 2 + 3 + \cdots + n$

$= \leq n$

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

Complexity $= O(n^2)$

Q.1] Sort the following array using quick sort:
44, 22, 33, 77, 11, 55, 66

44	22	33	77	11	55	66
pivot i						j

increment i pointer

44	22	33	77	11	55	66
pivot	i					j

increment i pointer

44	22	33	77	11	55	66
pivot		i				j

increment i pointer

44 22 33 77 11 55 66
pivot i j

Stop increment i & decrement the j

44 22 33 77 11 55 66
pivot i j

Stop decrement j
Swap A[i] & A[j]

44 22 33 55 11 77 66
pivot i j

decrement j

44 22 33 55 11 77 66
pivot i j

Stop decrement j & Swap A[i] = A[j]

44 22 33 11 55 77 66
pivot i j

increment i pointer

44 22 33 11 55 77 66
pivot i,j

Stop increment i pointer
& decrement j

44 22 33 11 55 77 66
pivot j i

j has Crossed i
So Swap A[j] & A[pivot]

11 22 33 44 55 77 66

LSL pivot RSL

- RSL

55 77 66

pivot i j

increment i pointer

55 77 66

pivot i j

Stop increment i & decrement j

55 77 66

pivot i,j

55 77 66

j,pivot i

leave 55

77 66

pivot j

66 77

- Ans :-

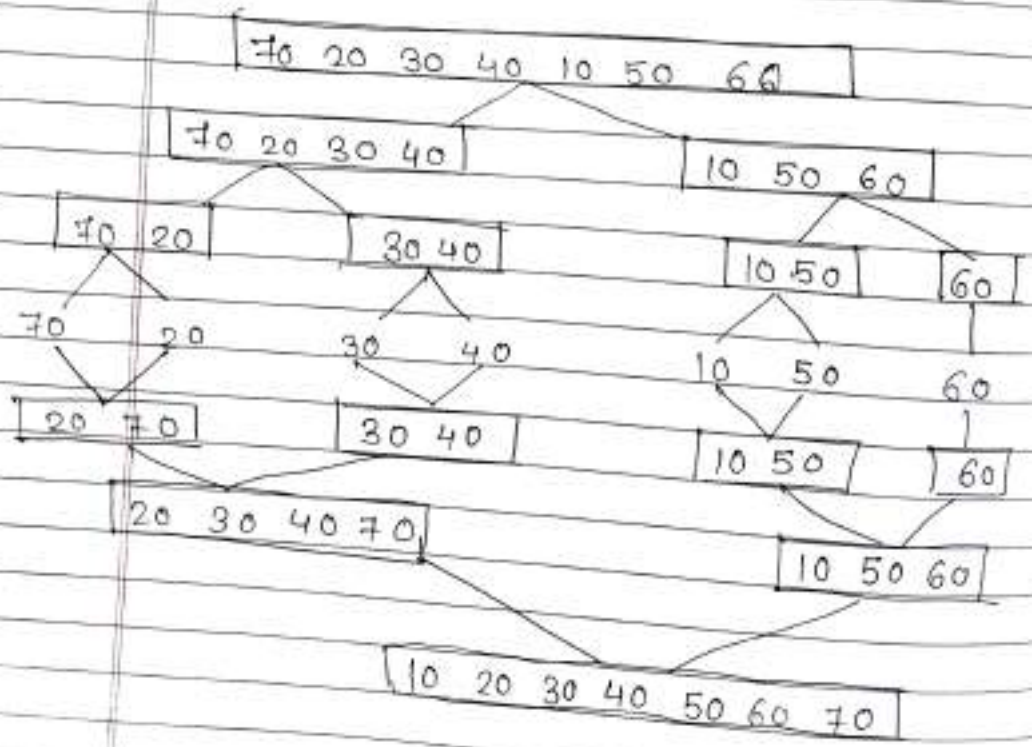
11 22 33 44 55 66 77 //

* Merge Sort :-

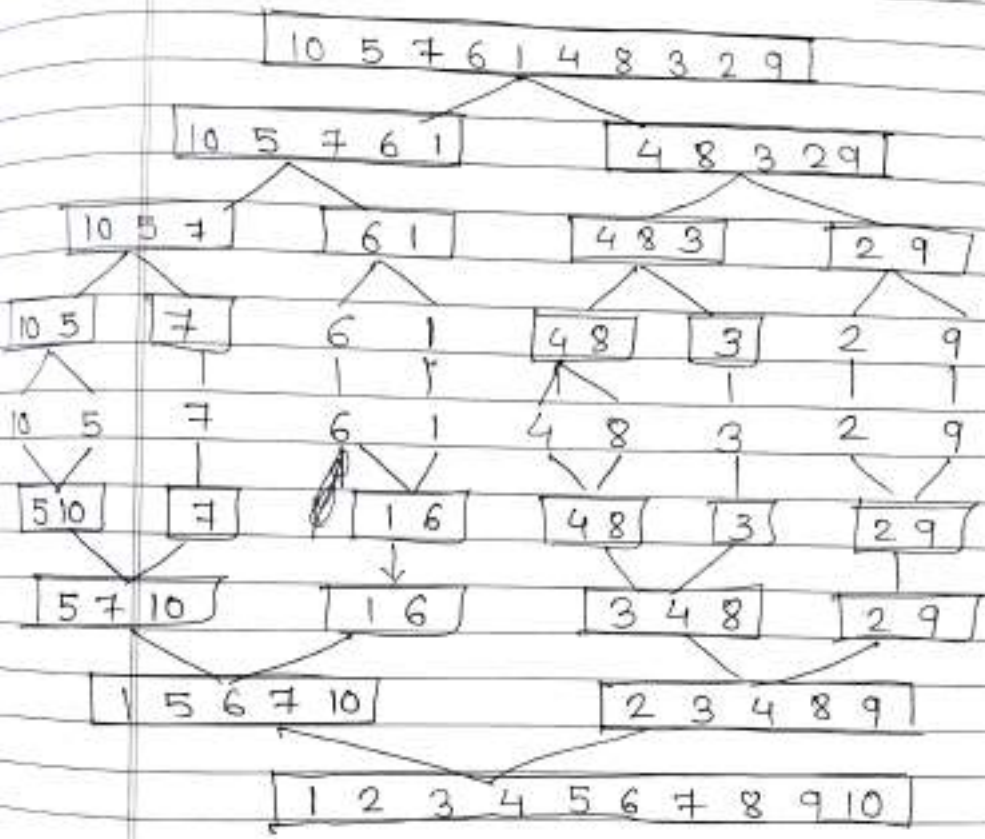
Array is divided into two parts each subpath is ~~sort~~ Solved independently. and combining the Solution of both the Subproblems into a Solution of whole.

Q.1] Sort the following array by using merge Sort:

70 20 30 40 10 50 60



Q02] Sort the following array by using merge Sort.
10, 5, 7, 6, 1, 4, 8, 3, 2, 9



- Algorithm:-

Algorithm MS(A, low, high)
low ← 0 & high ← n
if low < high then
mid ← floor (low + high)/2
MS (A, low, mid)
MS (A, mid+1, high)
Combine (A, low, mid, high)

```
end
Combine (A, low, mid, high)
  l1 = mid - low + 1
  l2 = high - mid
for i = 1 to l1 do
left [i] = A [low + i - 1]
end
  for j = 1 to l2 do
  Right [j] = A [mid + j]
  end
  left [l1 + 1] → ∞
  Right [l2 + 1] → ∞
  i ← 1, j ← 1
for k = low to high do
if left [i] ≤ Right [j]
  B [k] ← left [i]
  i ← i + 1
else
  B [k] ← Right [j]
    j ← j + 1
end
end
```

Eg:-

0	1	2	3	4	5	6
77	22	33	44	11	55	66

MS (A, 0, 6)

low = 0, high = 6

mid = $\frac{low + high}{2}$ = 3

MS (A, 0, 3)

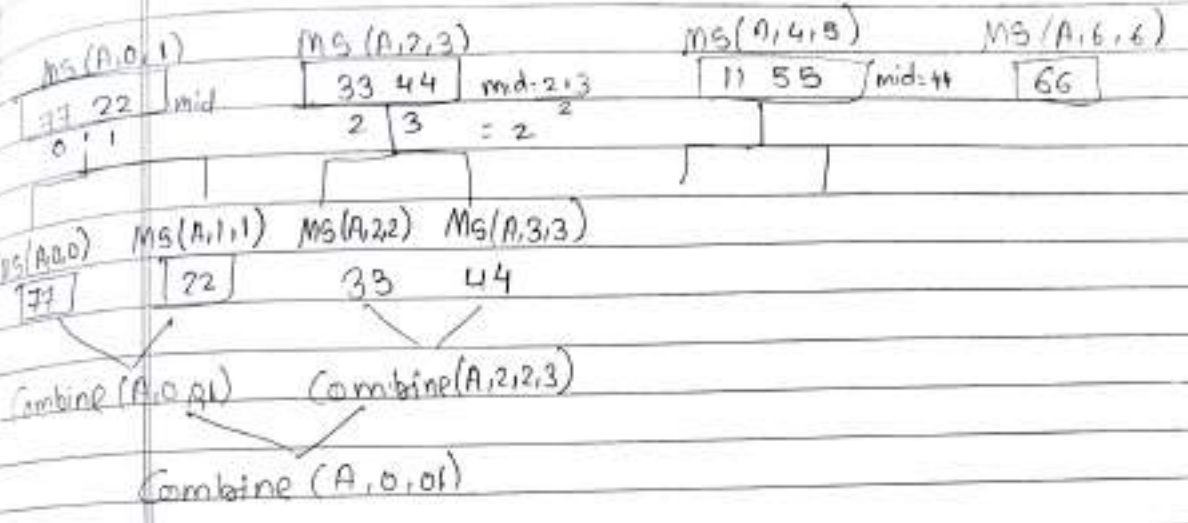
0	1	2	3
77	22	33	44

mid = $\frac{0+3}{2}$ = 1

MS (A, 4, 6)

4	5	6
11	55	66

mid = $\frac{4+6}{2}$ = 5



Combine (A,0,0,1)

low = 0, mid = 0, high = 1

l_1 = mid − low + 1

= 0 − 0 + 1 = 1

l_2 = high − mid

= 1 − 0 = 1

for i = 1 to 1

left [i] = A [0+1−1]

* Best Case, average Case & worst Case Complexity

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

- $T(n/2)$: Time taken by left Sublist to get Sorted
- $T(n/2)$: Time taken by right Sublist to get Sorted.
- n: time taken to combine two parts

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

Same as that of Bestcase average case of quick Sort algorithm.

Complexity $= O(n \log_2 n)$

06/03/25

* Binary Search:-

An element which is to be search from list of elements sorted in an array is called key elements. let A(m) is mid element then there are three conditions

i) key = A(m) then desired element is found in the list.

ii) key > A(m) } we need to
low = mid+1 } Search right Sublist

iii) key < A(m) } we need to search
high = mid-1 } left Sublist

Q.1] Find the element 60 from the following array by using binary Search.

10 20 30 40 50 60 70

→

Step I:- Arrange the elements in ascending order

10	20	30	40	50	60	70
1	2	3	4	5	6	7

low = 1 , high = 7

$mid = \frac{low + high}{2}$

$mid = \frac{1+7}{2} = 4$

A(mid) = 60 = key
desired element is found in the list.

A(mid) = 40

key > A(mid) True

low = mid+1

low = 4+1 = 5

$mid = \frac{5+7}{2} = 6$

11	22	33	44	55	66	77	88
1	2	3	4	5	6	7	8

low = 1 high = 8

$$mid = \frac{1+8}{2}$$

$$mid = \frac{9}{2} = 4$$

A(mid) = 44

key < A(mid) True

high = mid - 1

= 4 - 1

= 3

$$mid = \frac{1+3}{2}$$

mid = 2

A(mid) = 22

key > A(mid)

low = mid + 1

= 2 + 1

low = 3

$$mid = \frac{3+3}{2}$$

mid = 3

A(mid) = 33

* Algorithm_binary Search
low ← 1
high ← n
While low < high do
mid = $\frac{low + high}{2}$
if A[mid] = key then
return mid
else if A(mid) < key then
low ← mid + 1
else
high ← mid - 1
end
end

* Time Complexity:-

In every iteration binary search does one comparision and creates the new problem of size n/2.

$$T(n) = T\left(\frac{n}{2}\right) + 1 \quad \text{— (i)}$$

$T(n) = 1$ if $n = 1$

Replace n by n/2 in eqn (i)

$$T(n/2) = T\left(\frac{n}{4}\right) + 1 \quad \text{— (ii)}$$

put $T(n/2)$ in eqn (i)

$$T(n) = T\left(\frac{n}{4}\right) + 1 + 1$$

$$T(n) = T\left(\frac{n}{4}\right) + 2 \quad \text{— (iii)}$$

Replace n by n/4 in eqn (1) & n by n/2 in eqn (2)

$T\left(\frac{n}{4}\right) = T\left(\frac{n}{8}\right) + 1$

put $T(n/4)$ in eqn (2)

$T(n) = T\left(\frac{n}{8}\right) + 1 + 2$

$= T\left(\frac{n}{8}\right) + 3$

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

$\vdots$

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

Assume $n = 2^k$, $k = \log_2 n$

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

Complexity $= O(\log_2 n)$

Q.] Find the element 75 from the following array by using binary Search.

50, 40, 5, 9, 45, 90, 65, 25, 75.

Step I:- Arrange the array in Secnding order

5	9	25	40	45	50	65	75	90
1	2	3	4	5	6	7	8	9

low = 1 high = 9

$mid = \frac{9+1}{2} = 5$

A(mid) = 45

key > A(mid) True

low = mid + 1

= 5 + 1

low = 6

$mid = \frac{6+9}{2}$

mid = 7

A(mid) = 65

key > A(mid) True

low = mid + 1

= 7 + 1

low = 8

$mid = \frac{8+9}{2}$

mid = 8

A(mid) = 75

key = A(mid)

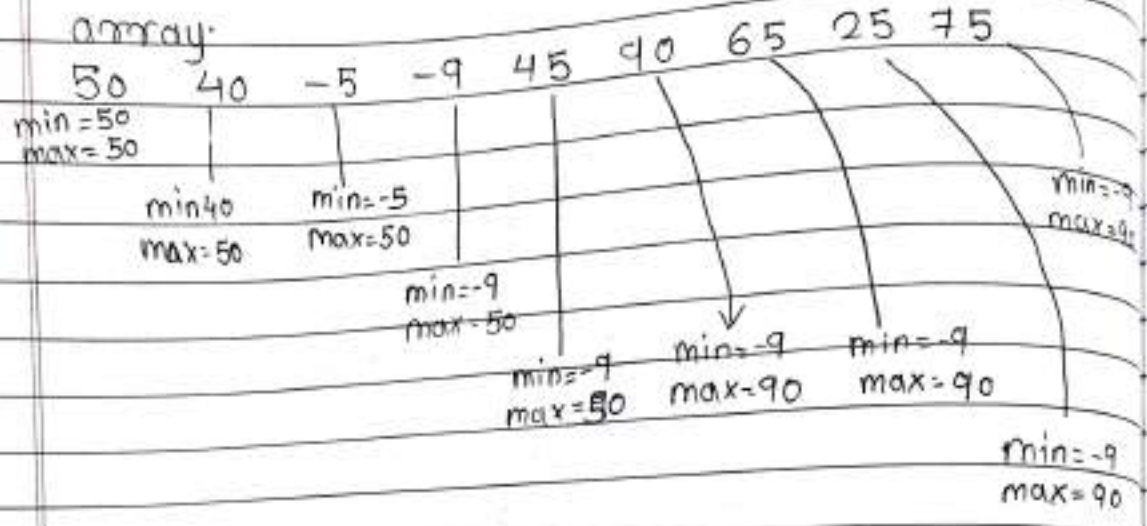
Element is found in the list.

Date 12/02/25
Page

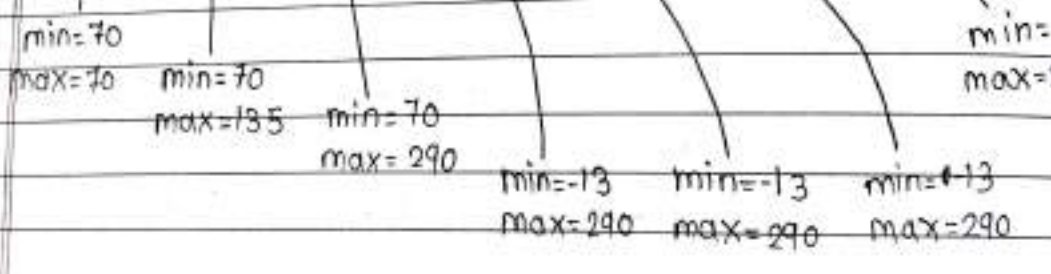
* Min / Max Algorithm :-

Q.1] Find min and max value from the following array:

50 40 -5 -9 45 90 65 25 75



(ii) 70 135 290 -13 -2 -12 65




```
Algorithm Max-Min (A[1....n], max, min)
    max ← min ← A[1]
    for (i=2 to n) do
    {
    if (A[i] > max) then
        max ← A[i]
    if (A[i] < min) then
        min ← A[i]
    }
}
```

* Time Complexity:-
problem is divided into two subproblems of size $\frac{n}{2}$ and two comparisons are needed to combine the result.

$$T(n) = 2T\left(\frac{n}{2}\right) + 2 \quad \text{if } n > 2$$
$$T(2) = 1$$
$$T(1) = 0$$
$$T(n) = 2T\left(\frac{n}{2}\right) + 2$$

Replace n by $\frac{n}{2}$ in eqn (i)

$$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + 2 \quad \text{— (ii)}$$

Put $T\left(\frac{n}{2}\right)$ in eqn (i)

$$T(n) = 2\left(2T\left(\frac{n}{4}\right) + 2\right) + 2$$

$$T(n) = 4T\left(\frac{n}{4}\right) + 4 + 2 \quad \text{— (iii)}$$

Replace n by $\frac{n}{4}$ in eqn (i) or n by $\frac{n}{2}$ in eqn (ii)

$$T\left(\frac{n}{4}\right) = 2T\left(\frac{n}{8}\right) + 2$$

Put $T\left(\frac{n}{4}\right)$ in eqn (iii)

$$T(n) = 8T\left(\frac{n}{8}\right) + 8 + 4 + 2$$

$$T(n) = 2^3 T\left(\frac{n}{2^3}\right) + 2^3 + 2^2 + 2^1$$

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + 2^1 + 2^2 + 2^3 + \cdots + 2^k$$

$$\frac{n}{2^k} = 2 \qquad \frac{n}{2} = 2^k \qquad n = 2^{k+1}$$

$$T(n) = 2^k T(2) + \frac{2(2^k - 1)}{2 - 1}$$

$$T(n) = 2^k + \frac{(2^{k+1} - 2)}{1}$$

$$= \frac{n}{2} + n - 2$$

$$= \frac{3n}{2} - 2$$

$$T(n) = 1.5n - 2$$

Complexity = O(n)

* Binary Search

Q.1] Search an element 55 in an array

37, 43 75 80 83 61 53 45 22 17

17 22 37 43 45 53 61 75 80 83

low = 1 high = 10

$$mid = \frac{low + high}{2}$$

$$= \frac{1+10}{2}$$

mid = 5

A(mid) = 45

Key > A(mid)

low = mid + 1

= 5 + 1

low = 6

$$mid = \frac{6+10}{2} = 8$$

A(mid) = 75

Key < A(mid)

high = mid - 1

= 8 - 1

high = 7

$mid = \frac{6+7}{2} = 6$

A(mid) = 53

key > A(mid)

low = mid + 1

low = 6 + 1 = 7

$mid = \frac{7+7}{2} = 7$

A(mid) = 61 Element is not found in the list.

Q02] Sort the following array by using quick Sort.

50, 31, 71, 38, 77, 81, 12, 33

pivot i j

incrementing i pointer

50 31 71 38 77 81 12 33

pivot i j

increment i

50 31 71 38 77 81 12 33

pivot i stop j

Stop & increment i & decrement j

Swap A[i] & A[j]

50 31 33 38 77 81 12 71
pivot i j

increment i

50 31 33 38 77 81 12 71
i j

increment i

50 31 33 38 77 81 12 71
i j

Stop incrementing i, & decrement j

50 31 33 38 77 81 12 71
i j

Stop decrementing j
Swap A[i] & A[j]

50 31 33 38 12 81 77 71
i j

increment i

50 31 33 38 12 81 77 71
i j

Stop incrementing i, decrement j

50 31 33 38 12 81 77 71
i,j

decrement j

50 31 33 38 12 81 77 71
j i

j has crossed i, Swap A[i] & A[Pivot]

12 31 33 38 50 81 77 71
Pivot

Left Sublist is already Sorted.

Right Sublist

81 77 71
pivot i j

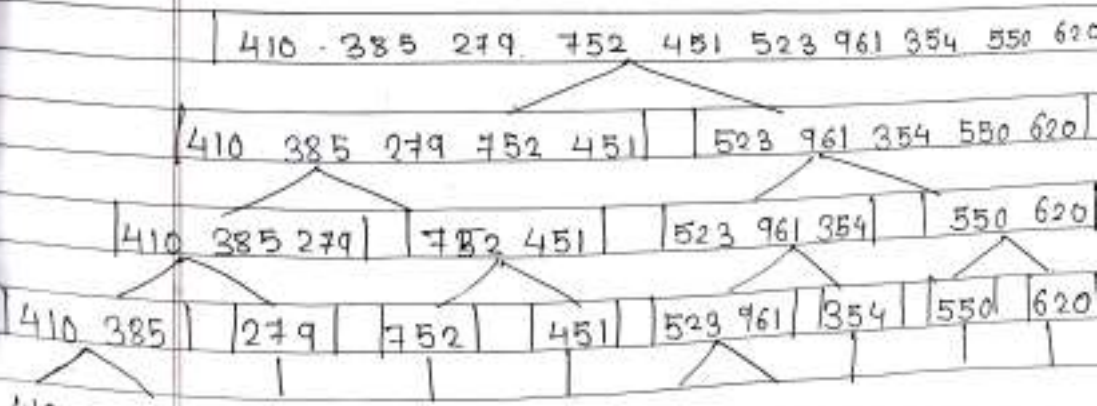
81 77 71
pivot i,j

Swap A[pivot] & A[i,j]

71, 77, 81

12 31 33 38 50 71 77 81.

Q03] Sort the following array by using Merged Sort.



Date 13/02/25
Page

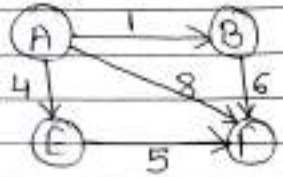
* Single Source Shortest path :- (D&A)

Divide & Conquer approach	Greedy Method.
i) It is used to find Solution to the problem	i) It is used to find optimal Solution.
ii) In this approach problem is divide into two sub-problems each subproblem is solved independently. Combinning the solutions of both the sub-problems into a Solution of	ii) physical Solution is generated and optimal Solution is picked up.
iii) In this approach is less effecient because of rework of Solutions.	iii) Greedy method is more efficeint.
iv) eg:- Quick Sort, merged Sort, binary Search & min max algorithm.	iv) eg:- Single Source Shortest path, Job Sequencing with deadline and minimum Spanning tree algorithm knapstack problem.

- Dijkstra's Algorithm :- (SSSP).
- It is used to find shortest path from Source to destination.

Algorithm Dijkstra's (G, S, t) :-

```
S → Source Vertex
t → target Vertex
dist [S] ← 0
  P [S] ← nil
for each Vertex V
   if V ≠ S then
     dist [V] ← ∞
       P [V] ← nil
  end
  end
U → Unprocessed Vertex in Q
 having minimum distance
    if U == t then
    break
for each adjacent node V of U do
    Val [V] ← dist [U] + weight (U, V)
    if  Val [V] < dist [V] then
        dist [V] ← Val [V]
          P [V] ← U
  end
  end
```



$S \leftarrow A$

$t \leftarrow f$

$dist[A] \leftarrow 0$

$P[A] \leftarrow nil$

$dist(B) = dist(E) = dist(F) = \infty$

$P(B) = P(E) = P(F) = nil$

Vertex u	A	B	E	f
dis[u]	0	∞	∞	∞
P[u]	nil	nil	nil	nil

u → Unprocessed vertexing having minimum distance.

$u = A$

Adjacent of $A = B, E, F$

$Val(B) = dist(A) + wt(A,B)$

$= 0 + 1 = 1$

$Val(B) < dist(B)$

$1 < \infty$

$dist(B) = 1$

$P(B) = A$

$Val(E) = dist(A) + wt(A,E)$

$= 0 + 4 = 4$

$Val(E) < dist(E)$

$dist(E) = 4$

$P(E) = A$

Val (F) = dist(A) + wt (A, F)
= 0 + 8 = 8
Val (F) < dis (F)
dis (F) = 8
P(F) = A

Vertex u	A	B	E	F
dist[u]	0	1	4	8
p(u)	nil	A	A	A

u ← unprocessed vertex in Q having minimum distance.
u = B
Adjacent of B = E
Val (E) = dist(B) + wt (B, E)
= 1 + 6
= 7
Val (E) < dist (E)
7 < 8
dist (E) = 7

Vertex u	A	B	E	F
dist[u]	0	1	4	7
p(u)	nil	A	A	B

u = E
Adjust of E = F
Val (F) = dist (E) + wt (E, F)
= 4 + 5
= 9
Val (F) > dist (F)
9 > 7
No change in the table.

Q

u = f

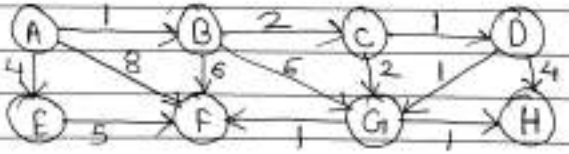
Adjacent of f = { }

final path

A → B → f

Value = 7

Q.01] Suppose dijkstra Algorithm is run on the following graph starting at node A. Draw a table showing the intermediate distance values of all the node at each iteration of the algorithm and show the final shortest path of tree.



dist [A] = 0

P(A) = nil

dist(E) = dist(F) = dist(G) = dist(H) = ∞

P(E) = P(F) = P(G) = P(H) = P(B) = P(C) = P(D) = P(E) = nil

Vertex [u]	A	B	C	D	E	F	G	H
dist [u]	0	∞	∞	∞	∞	∞	∞	∞
P(u)	——— nil ———							

Adjacent of A = B, E, F

dist [B] = 1 , dist [E] = 4 , dist (F) = 8

Vertex[V]	A	B	C	D	E	F	G	H
dist[u]	0	1	∞	∞	2	3	∞	∞
P(u)	nil	A	nil	nil	A	A	nil	nil

Adjacent of B = C, F, G

dist[C] = 3 dist[F] = 7 dist(G) = 7

Vertex u	A	B	C	D	E	F	G	H
dis[u]	0	1	3	∞	4	7	7	∞
P[u]	nil	A	B	nil	A	B	B	nil

Adjacent of C = D, G

dist[D] = 4 dist[G] = 5

Vertex u	A	B	C	D	E	F	G	H
dis[u]	0	1	3	4	4	7	5	∞
P[u]	nil	A	B	C	A	B	C	nil

Adjacent of D = G, H

val(G) = 5

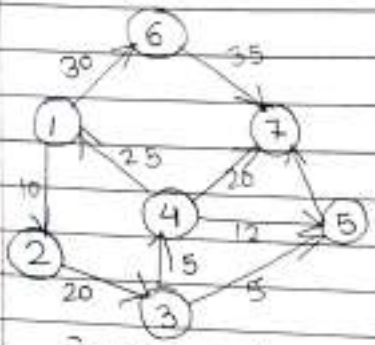
val(G) = dis(G)

dist(G) =

No change.

dist(H) = 8

Q.2] Find minimum distance from vertex 1 to 7



→ Source = 1
destination = 7
dist[1] = 0
P(1) = nil
dist(2) = dist(3) = dist(4) = dist[5] = dist(6) = dist(7) = ∞
P(2) = P(3) = P(4) = P(5) = P(6) = P(7) = nil

Vertex u	1	2	3	4	5	6	7
dist(u)	0	∞	∞	∞	∞	∞	∞
P(u)	nil						

u = 1
Adjacent of 1 = 2, 6
Valu(2) = dis(1) + wt(1,2)
= 0 + 10 = 10
Val(2) < dist(2)
10 < ∞
dist(2) = 10
P(2) = 1
dist(6) = 30
P(6) = 1

Vertex u	1	2	3	4	5	6	7
dist[u]	0	10	∞	∞	∞	30	∞
P(u)	nil	1	——	nil	——	1	nil

u = 2
adjacent of 2 = 3
dist (3) = 30
P(3) = 2

Vertex u	1	2	3	4	5	6	7
dist[u]	0	10	30	∞	∞	30	∞
P(u)	nil	1	2	nil	nil	1	nil

u = 3
adjacent of 3 = 4, 5
dist (4) = 45
dist (5) = 35
P(4) = 3 , P(5) = 3

Vertex u	1	2	3	4	5	6	7
dist[u]	0	10	30	45	35	30	∞
P(u)	nil	1	2	3	3	1	nil

u = 6
adjacent of 6 = 7
dist (7) = 65

Vertex u	1	2	3	4	5	6	7
dist[u]	0	10	30	45	35	30	42
P(u)	nil	1	2	3	3	1	5

u = 5
adjacent of 5 = 7
Val (7) = 42

Val(7) < dist(7)
dist(7) = 42
u = 7
adjacent of 7 = {3}

u = 4
adjacent of 4 = 1, 7, 5
Val(1) = 70
No change.
Val(7) = 65
No change
Val(5) = 57
No change
Final table shortest path
1 - 2 - 3 - 5 - 7

* Job Sequencing with deadline:-

- Objective of job sequencing with deadline is to schedule the jobs will be completed within given deadline gives max profit.

- Algorithm:-

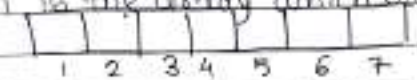
Algorithm Job-Sequencing (J, D, P)
// Input J: array of N jobs
// D: array of deadline for each job
// P: array of profit associated with each job
// Sort all jobs in decreasing ordered of profit

S ← ϕ // S is set of Schedule jobs, initially
// it is empty
SP ← 0 // Sum of the profit
for i=1 to N do.
if job J(i) is feasible then Schedule the job in latest possible free slot meeting its deadline.
S ← S ∪ J(i) Union operation
SP ← SP + P[i]
end
end

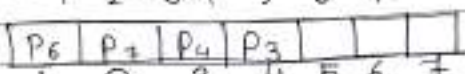
Q.1] Solve the following instances of job Sequencing with deadline problem.
n=7
profit $(P_1, P_2, P_3, P_4, P_5, P_6, P_7) = (3, 5, 20, 18, 1, 6, 30)$
& deadline $(d_1, d_2, d_3, d_4, d_5, d_6, d_7) = (1, 3, 4, 3, 2, 1, 2)$

Profit:	30	20	18	6	5	3	1
Job:	P_7	P_3	P_4	P_6	P_2	P_1	P_5
deadline:	2	4	3	1	3	1	2

S_1 is the array which stores N jobs



← initially it is empty.



optimal sequence
$P_6 - P_7 - P_4 - P_3$
profit = 6+30+18+20 = 74

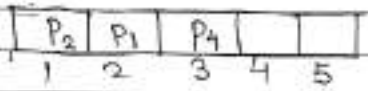
deadline for P_2 is 3
But 3 is already occupied So Search empty location less than Current location. But 1, 2 are also occupied.
So discard P_2

Q.1] Solve the following instances of Job Sequencing with deadline problem.

n = 5
$P_1, P_2, P_3, P_4, P_5 = 20, 15, 10, 5, 1$
$d_1, d_2, d_3, d_4, d_5 = 2, 2, 1, 3, 3$

→

profit:	20	15	10	5	1
Job:	P_1	P_2	P_3	P_4	P_5
deadline:	2	2	1	3	3



discard P_3
discard P_5
optimal Sequence :- $P_2 \rightarrow P_1 \rightarrow P_4$
profit :- 15 + 20 + 5 = 40

* Knapsack problem :-

Consider that there are n objects each objects has weight and profit associated with it. Objective is to choose only those objects that gives maximum profit.

fractional-knapsack (weight (1....n), profit (1-n), capacity)

weight = 0
profit = 0

// find ratio = $\frac{Pi}{wi}$

// Arrange Pi/wi in descending order.

```
for i=1 to n
{
  if weight + w[i] ≤ w then   (Capacity)
  {
    x[i] = 1
    weight = weight + w[i]
    profit = profit + p[i]
  }
  else
  {
    x[i] = (w - weight)/w(i)
    weight = weight + x[i] * w[i]
    profit = profit + x[i] * p[i]
  }
  weight = w
  break
  return x
}
```

Q.1] Find the optimal solution for the following knapsack instances using greedy method.

No. of items = 4

Capacity of knapsack = 60

item	A	B	C	D
profit	280	100	120	120
weight	40	10	20	24

item	A	B	C	D
profit	280	100	120	120
weight	40	10	20	24
$\frac{Pi}{wi}$	7	10	6	5

item	B	A	C	D
Profit	100	280	120	120
weight	10	40	20	24
$\frac{Pi}{wi}$	10	7	6	5

for i = 1 to 4 Capacity = 60

$0 + 10 \leq 60$ — True

$x[1] = 1$

weight = 0 + 10 = 10

profit = 0 + 100 = 100

weight == w

10 == 60 No

i = 2

$10 + 40 \leq 60$ — True

$x[2] = 1$

weight = 10 + 40 = 50

profit = 100 + 280 = 380

50 == 60 No

i = 3

$50 + 20 \leq 60$ - false

$x[3] = \frac{60 - 50}{20} = \frac{10}{20}$

Weight = Weight + x[i] * w[i]

$= 50 + \frac{10}{20} \times 20$

$= 60$

profit $= 380 + \frac{10}{20} \times 120$

$= 380 + 60$

$= 440$

Weight = = 60

60 = = 60

Break

return x

Q02] No. of objects n = 5

Capacity of knapstack = 100

profit = 10, 20, 30, 40, 50

items = I1, I2, I3, I4, I5

weights = 20, 30, 66, 40, 60

items	I1	I2	I3	I4	I5
profit	10	20	30	40	50
weights	20	30	66	40	60
$\frac{Pi}{wi}$	$\frac{1}{2}$	$\frac{2}{3}$	0.45	1	$\frac{5}{6}$

Items	profit	weight	pi/wi
I4	40	40	1
I5	50	60	0.83
I2	20	30	0.67
I1	10	20	0.5
I3	30	66	0.45

Capacity = 100

for i = 1 to 5

0 + 40 ≤ 100 True

x[i] = 1

Weight = 0 + 40 = 40

profit = 0 + 40 = 40

Weight = = w

40 = = 10 No

i = 2

40 + 60 ≤ 100 True

Weight = 40 + 60 = 100

profit = 40 + 50 = 90

Weight = = w

100 = = 100 yes

Break

Knapsack is full no more items can be added

Weight = 100

profit = 90

* Time Complexity:-

- Pi|wi is arranged in desending order, if we used merged sort or quick sort then the Complexity is $O(n \log_2 n)$.
and for loop runs for n times so overall Complexity is $O(n \log_2 n)$