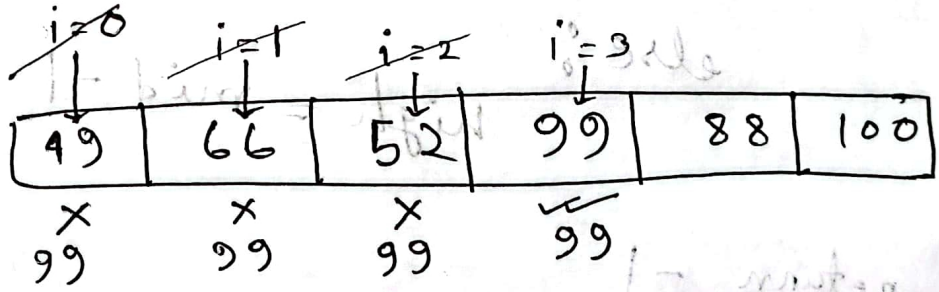
Searching

Linear Search

code:

```
def linear_search (arr, target):
    for i in range (len(arr)):
        if arr[i] == target:
            return i
    return -1
```

simulation: Find if 99 is in the array.



it will return the index 3.

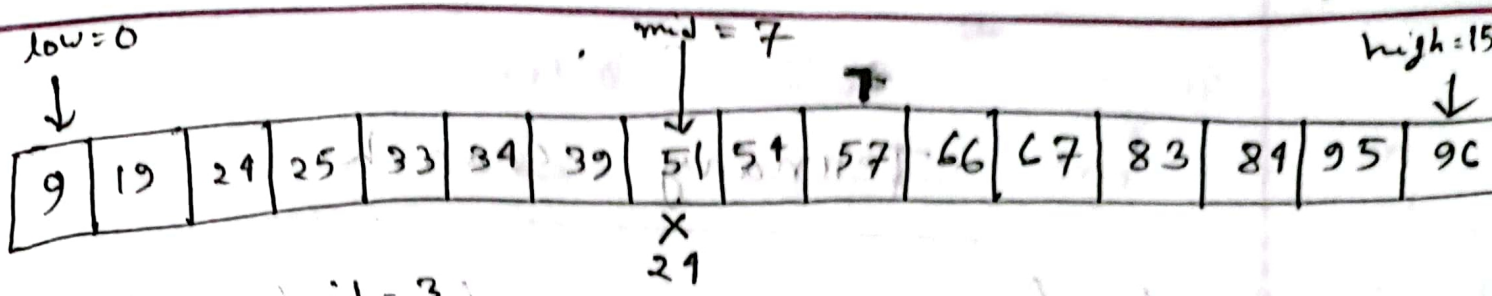
Binary Search

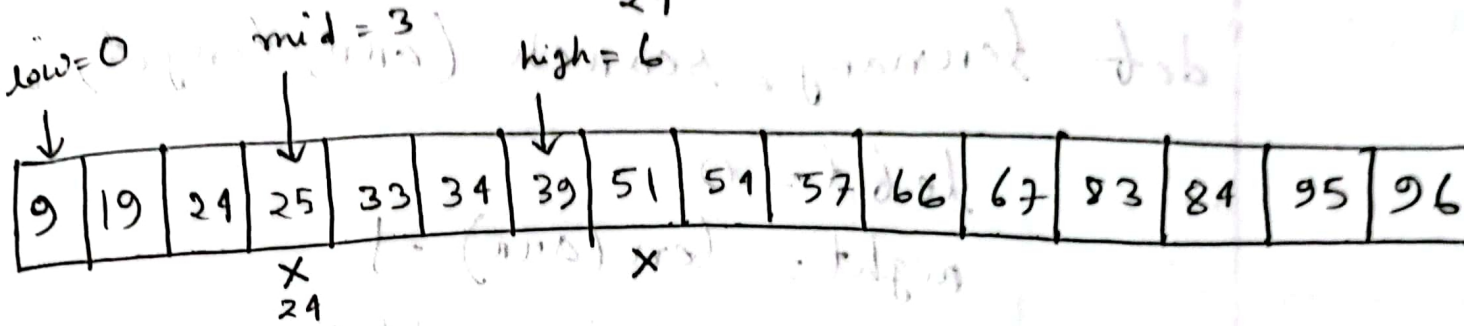
Code:

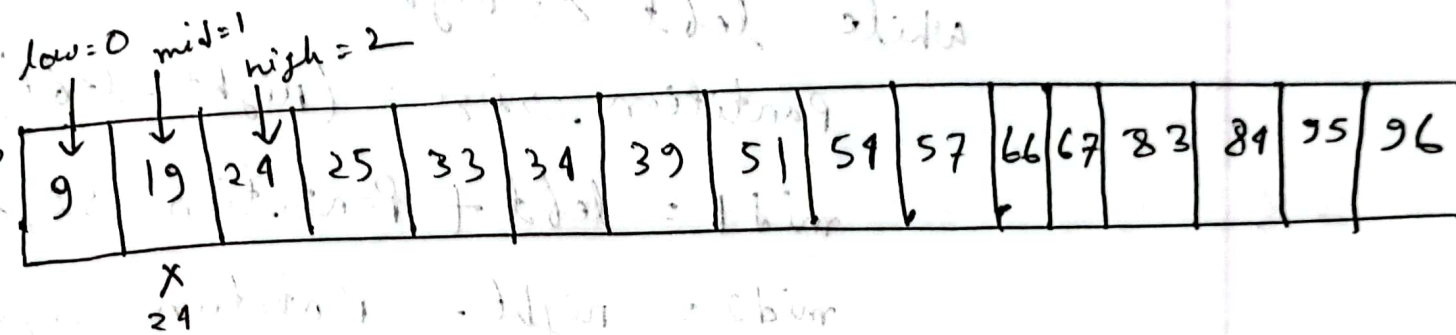
```
def binary_search(arr, target):
    low = 0
    high = len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return -1
```

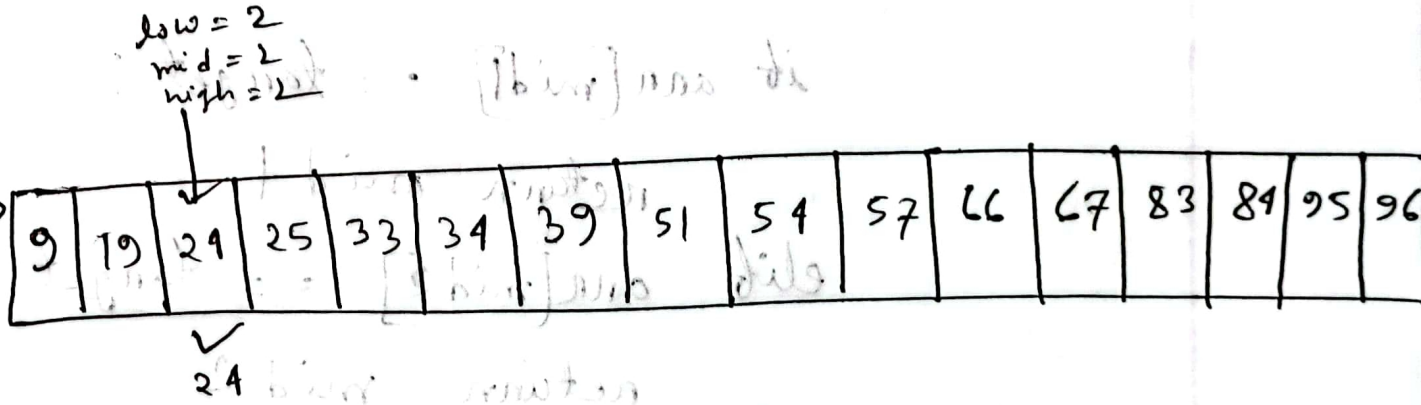
simulation: Find if 24 is in there.

9	19	24	25	33	34	39	51	54	57	66	67	83	84	95	96



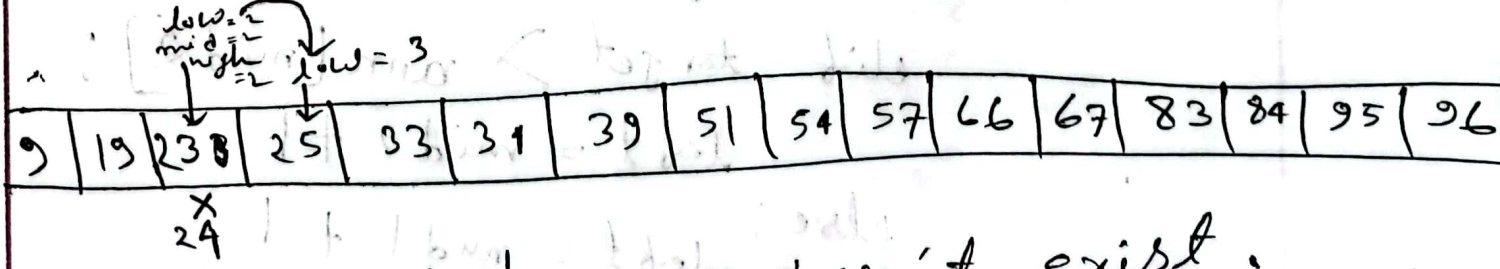






∴ 24 was found at index number 2.

⊛ suppose 24 এর জায়গায় 23 থাকলে,



as low > high ; 24 doesn't exist.

Ternary search

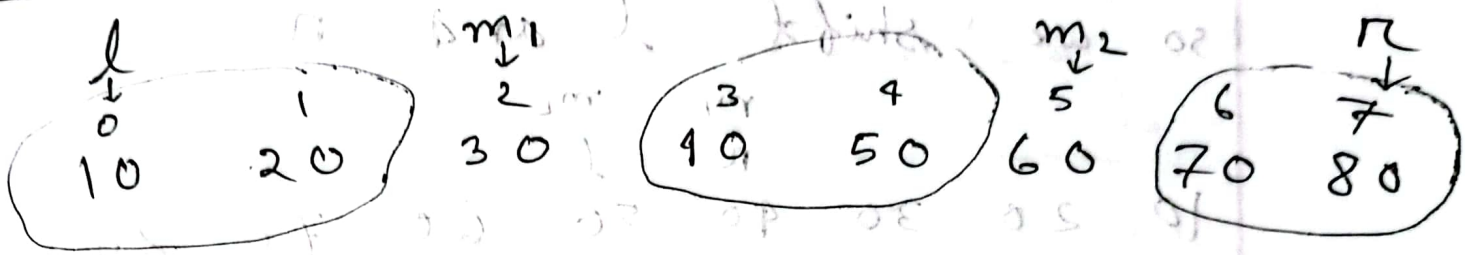
```
def ternary_search(arr, target):
    left = 0
    right = len(arr) - 1
    while left <= right:
        partition_size = (right - left) // 3
        mid1 = left + partition_size
        mid2 = right - partition_size

        if arr[mid1] == target:
            return mid1
        elif arr[mid2] == target:
            return mid2
        elif target < arr[mid1]:
            right = mid1 - 1
        elif target > arr[mid2]:
            left = mid2 + 1
        else:
            left = mid1 + 1
            right = mid2 - 1
    return -1
```

Partition = $\frac{7-0}{3} \approx 2$

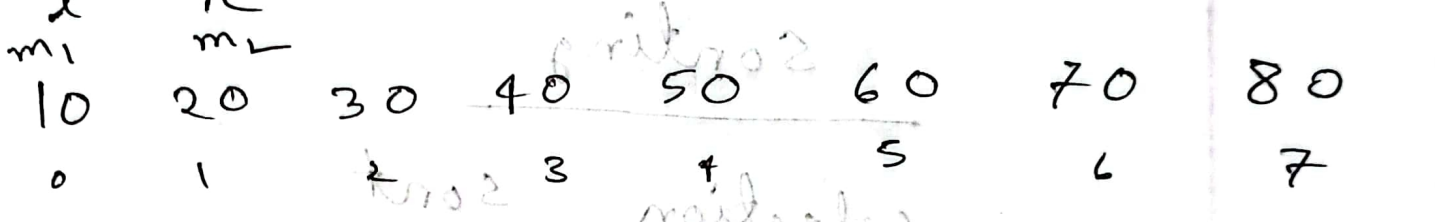
$m_1 = 0 + 2 = 2$

$mid_2 = 7 - 2 = 5$

simulation: Find 10



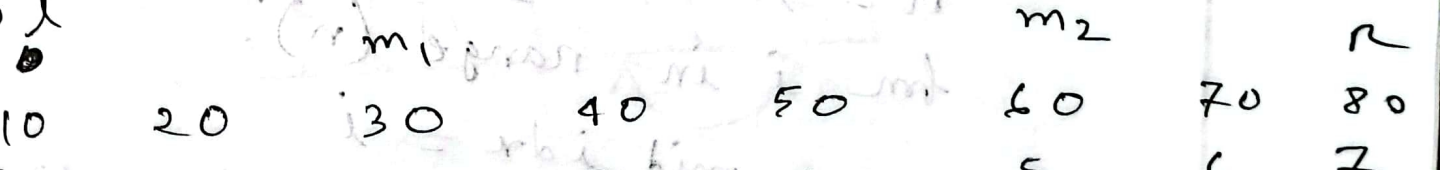
as $10 < 30$ on m_1, we move r to the first Part of left to m_1.



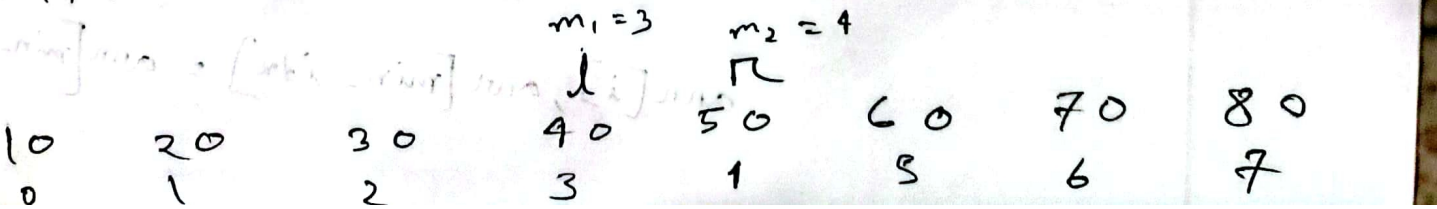
as $m_1 = 10$, we found 10.

(*) Find 45.

⇒



as $45 > m_1$ and $45 < m_2$ we shift l to m_1+1 and r to m_2-1.



Eg: $m_1 < 45 < m_2$?

so we shift l and r.

		m_1		m_2			
		r		l			
10	20	30	40	50	60	70	80
0	1	2	3	4	5	6	7

but here $l > r$, so 45 doesn't exist.

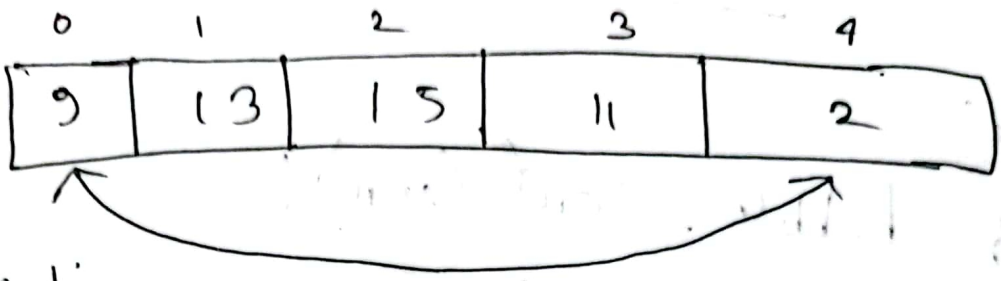
Sorting

selection sort

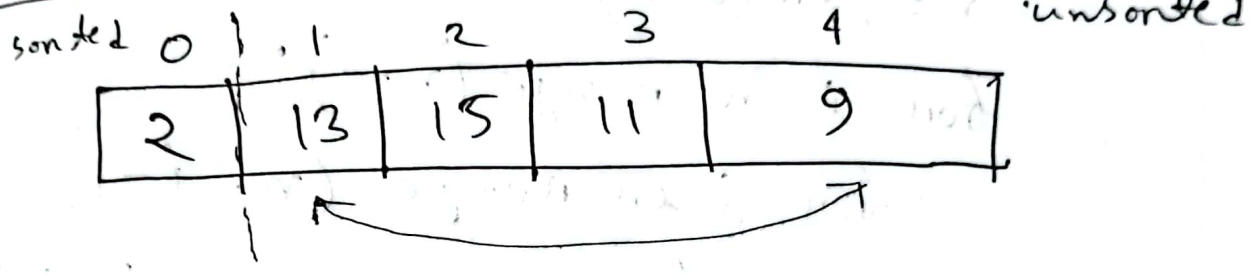
Code:

```
def selection_sort(arr):
    n = len(arr)
    for i in range(n):
        min_idx = i
        for j in range(i+1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
```

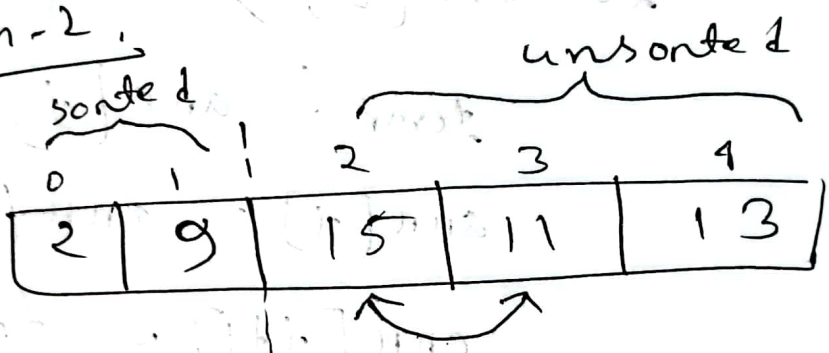
simulation :



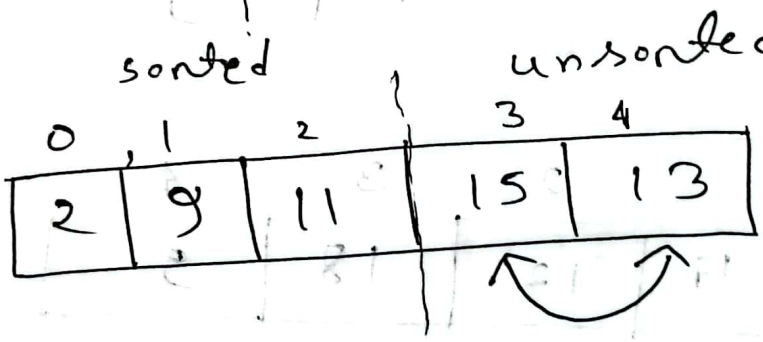
iteration-1 :



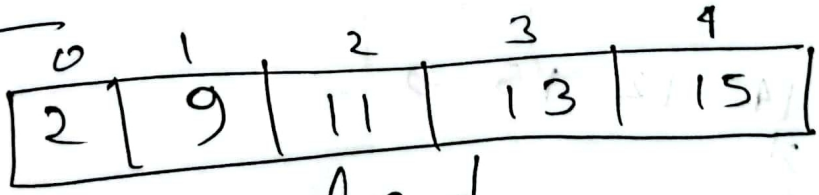
iteration-2 :



it-3 :



it-4 :



sorted

Bubble sort

Code:

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n-1):
        for j in range(n-i-1):
            if arr[j] > arr[j+1]:
                temp = arr[j]
                arr[j] = arr[j+1]
                arr[j+1] = temp
```

simulation:

0	1	2	3	4
22	14	12	18	9

it-1:

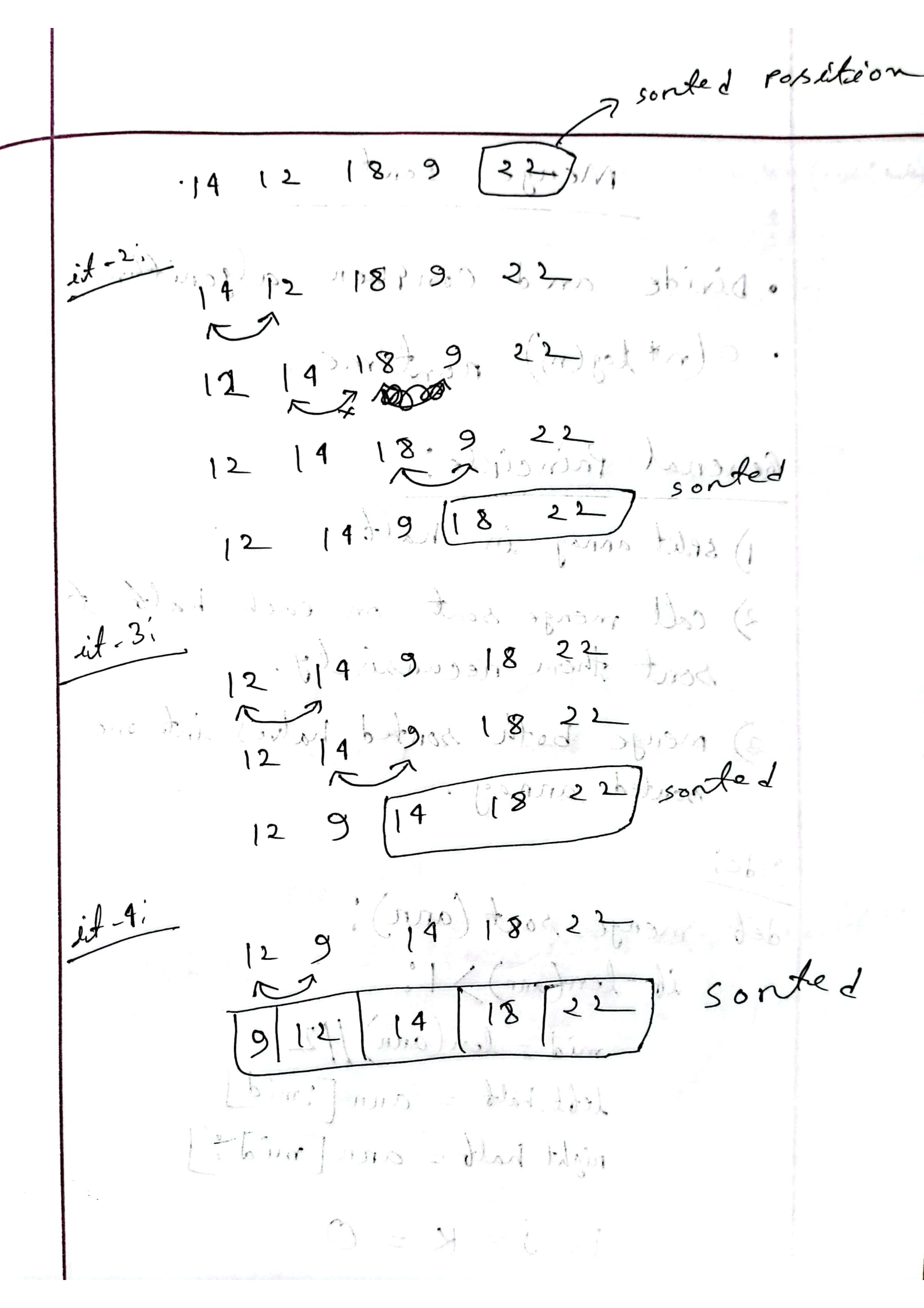
22 14 12 18 9

as 22>14

14 22 12 18 9

14 12 22 18 9

14 12 18 22 9


sorted position
14 12 18 9 22
it-2:
14 12 18 9 22
12 14 18 9 22
12 14 18 9 22
12 14 9 18 22
sorted
it-3:
12 14 9 18 22
12 14 9 18 22
12 9 14 18 22
sorted
it-4:
12 9 14 18 22
9 12 14 18 22
sorted

Merge sort

- Divide and conquer algorithm.
- O(n* log(n)) runtime

General Principle:

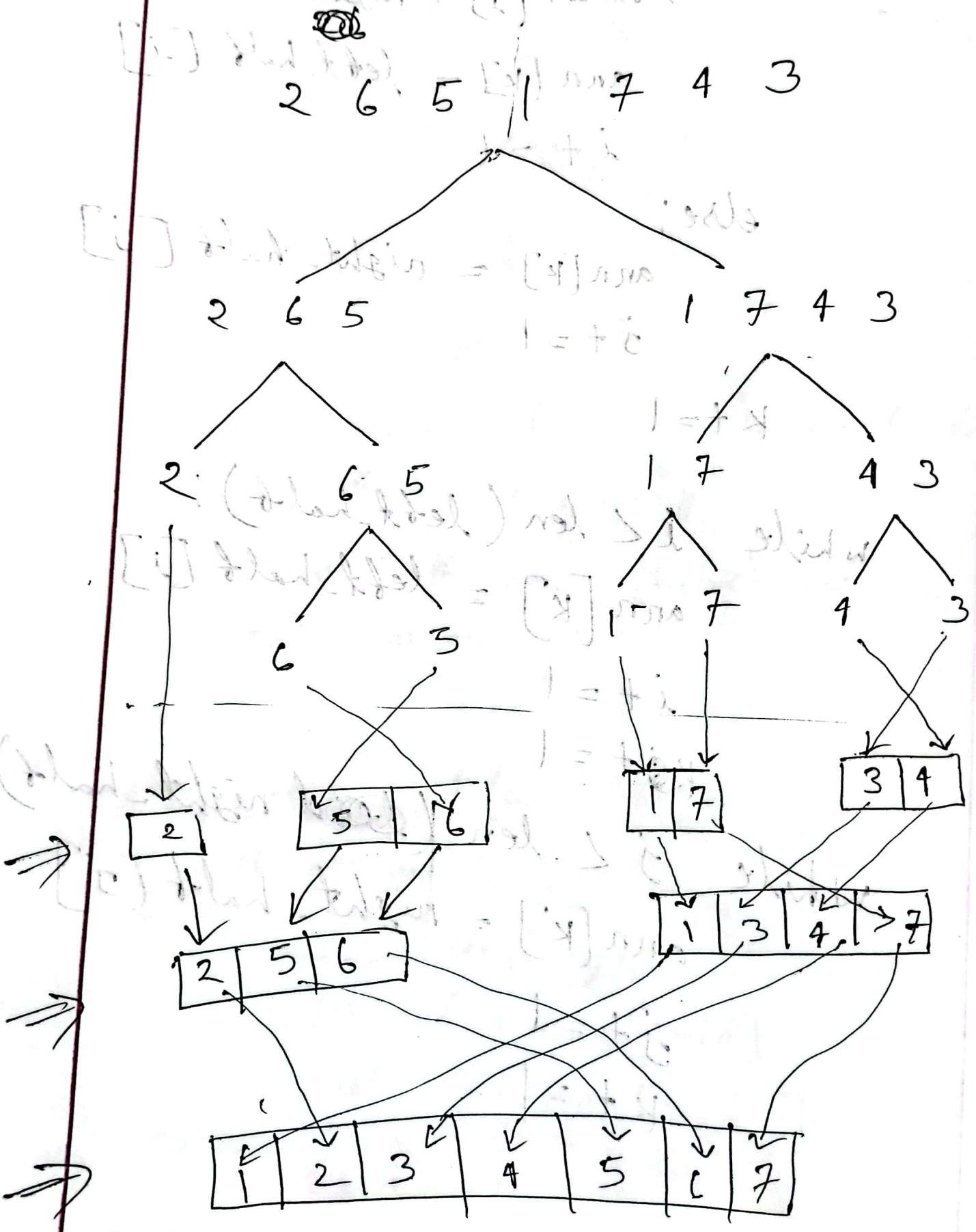
1) split array in half
2) call merge_sort on each half to sort them recursively.
3) merge both sorted halves into one sorted array.

Code:

```
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr)//2
        left_half = arr[:mid]
        right_half = arr[mid:]
        merge_sort(left_half)
        merge_sort(right_half)
        i = j = k = 0
```

```
while i < len(left_half) and j < len(right_half):
    if left_half[i] < right_half[j]:
        arr[k] = left_half[i]
        i += 1
    else:
        arr[k] = right_half[j]
        j += 1
    k += 1

while i < len(left_half):
    arr[k] = left_half[i]
    i += 1
    k += 1

while j < len(right_half):
    arr[k] = right_half[j]
    j += 1
    k += 1
```

Simulation:



Quick sort

(*) uses Divide and conquer algo.

(*) In worst case run time $O(n^2)$

In best/average " " $O(n \log n)$.

General Principle:

1) Choose pivot element (Generally last element or random).

2) stores elements less than pivot in left subarray.
stores elements greater than pivot in right subarray.

3) Call quicksort recursively on left subarray.
" " " right "

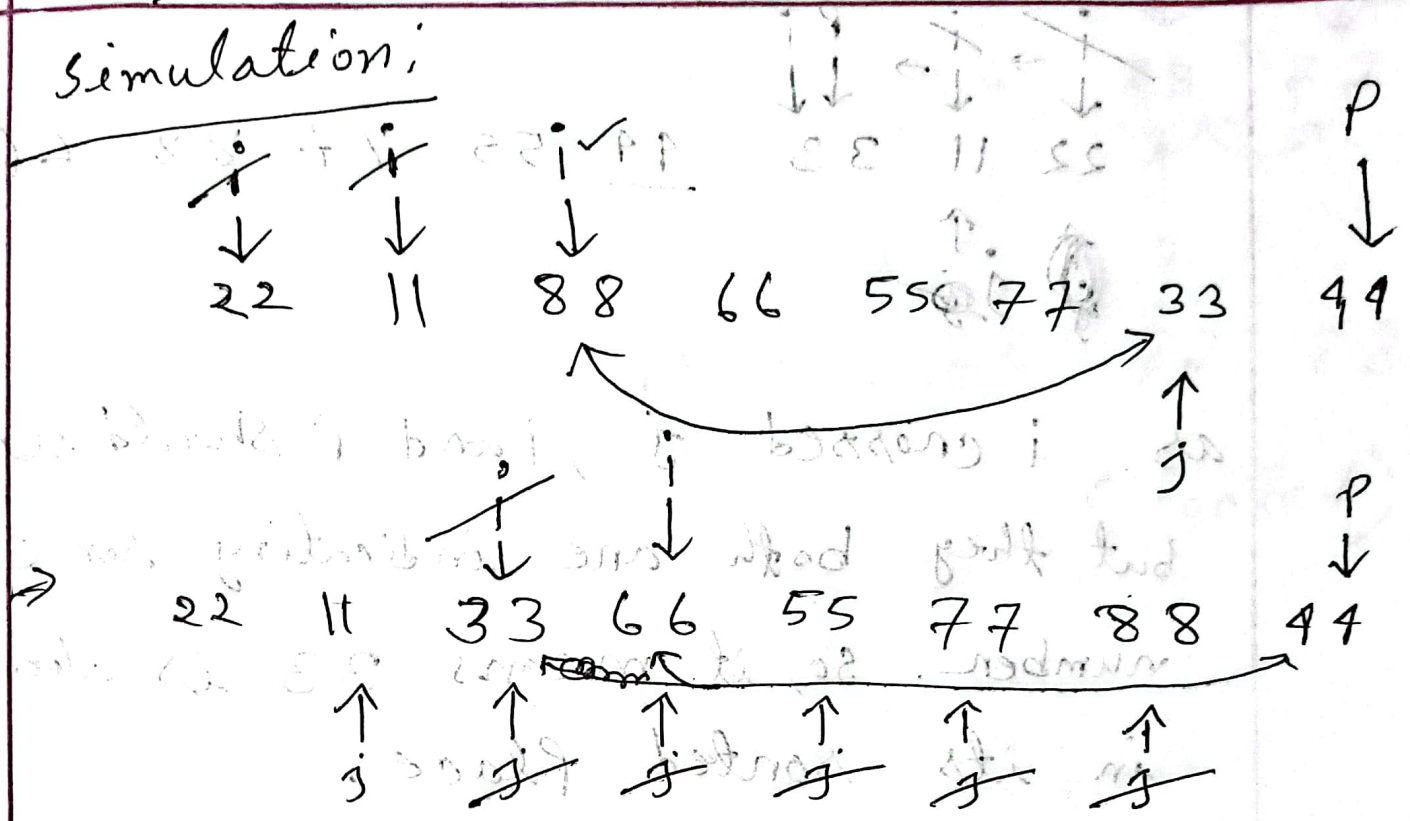
Code:

```
def Partition(arr, low, high):
    pivot = arr[high]  [: choosing the rightmost element]

    i = low - 1
    for j in range(low, high):
        if arr[j] <= pivot:
            i += 1
            arr[i], arr[j] = arr[j], arr[i]

    arr[i+1], arr[high] = arr[high], arr[i+1]

    return i + 1

def quicksort(arr, low, high):
    if low < high:
        pivot_index = Partition(arr, low, high)
        quicksort(arr, low, pivot_index - 1)
        quicksort(arr, pivot_index + 1, high)
```

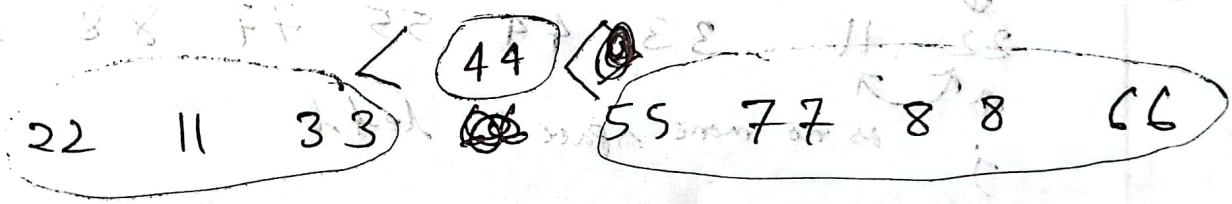
**

we will move i until ann[i] > Pivot
" " " j " ann[j] < Pivot
then we will swap

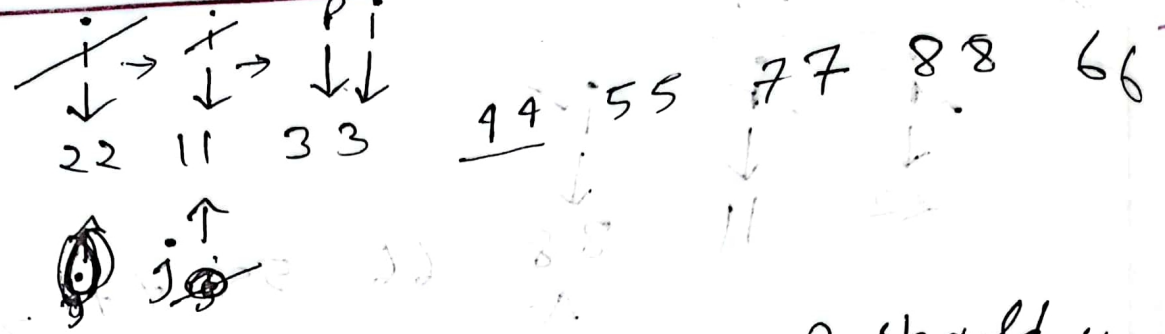
Simulation;



as j crossed i, we will swap i and P.

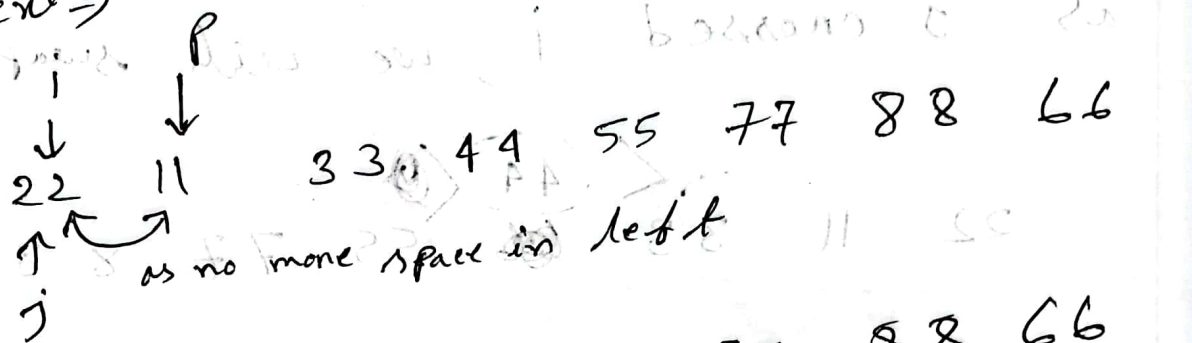


** after first iteration Pivot, P=44 is now at sorted position as every number left to it is smaller than 44 and numbers right to it is bigger.
Now there is 2 sub ann. left and right side. Let us work with left side first



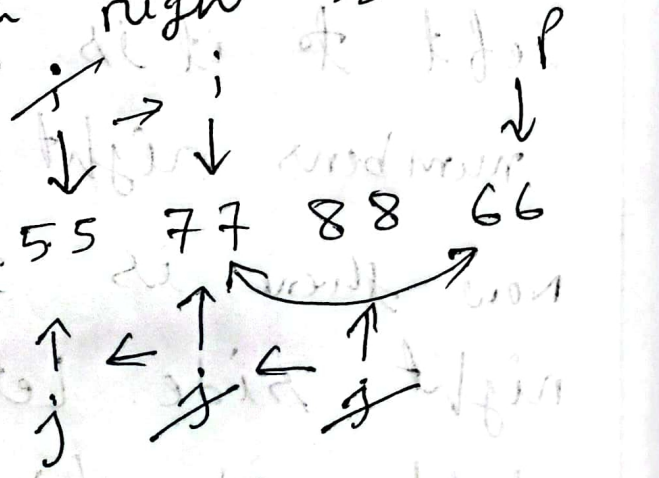
as i crossed j, i and p should swap but they both are indicating same number. so, it means 33 is already in its sorted place.

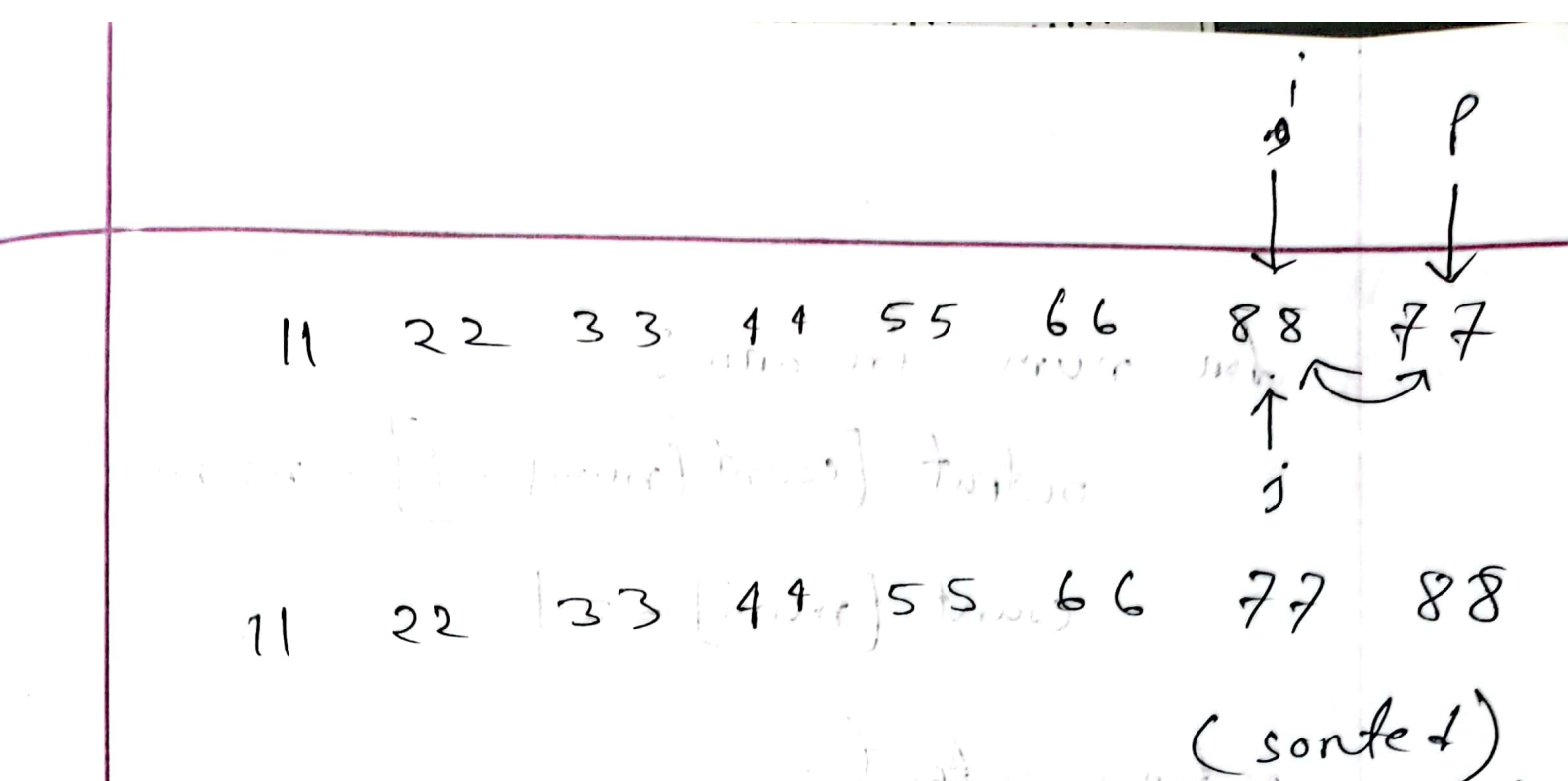
next ⇒



⇒ 11 22 33 44 55 77 88 66

Now lets work with right sub array.





(sorted).

Count sort

Code:

```
def count_sort(arr):
    max_elem = max(arr)
    count = [0] * (max_elem + 1)

    for num in arr:
        count[num] += 1
    for i in range(1, len(count)):
        count[i] += count[i-1]
    output = [0] * len(arr)
```

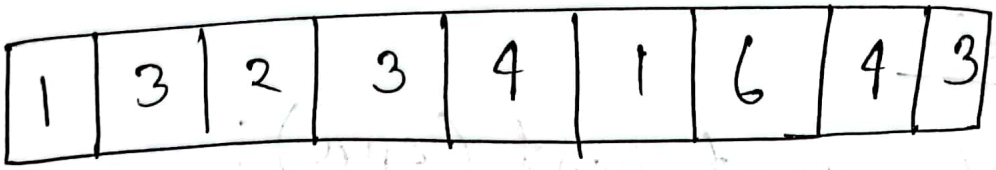
for num in arr:

output [count[num] - 1] = num

count [num] - = 1

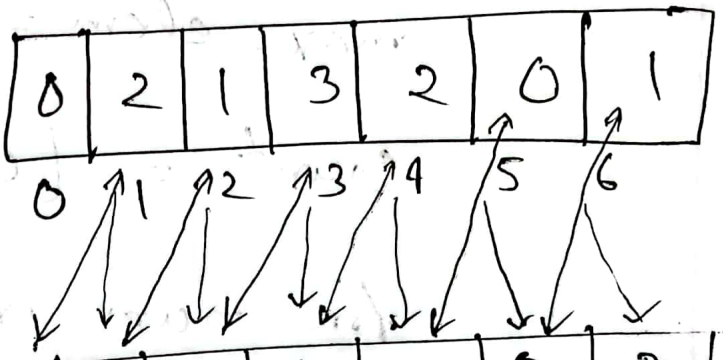
return output.

Simulation:

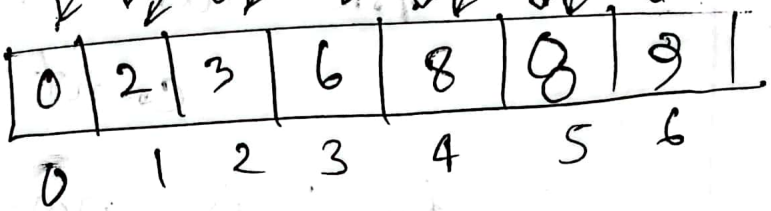


⇒ count array →

as max elem = 6
count. array size = 6+1 = 7



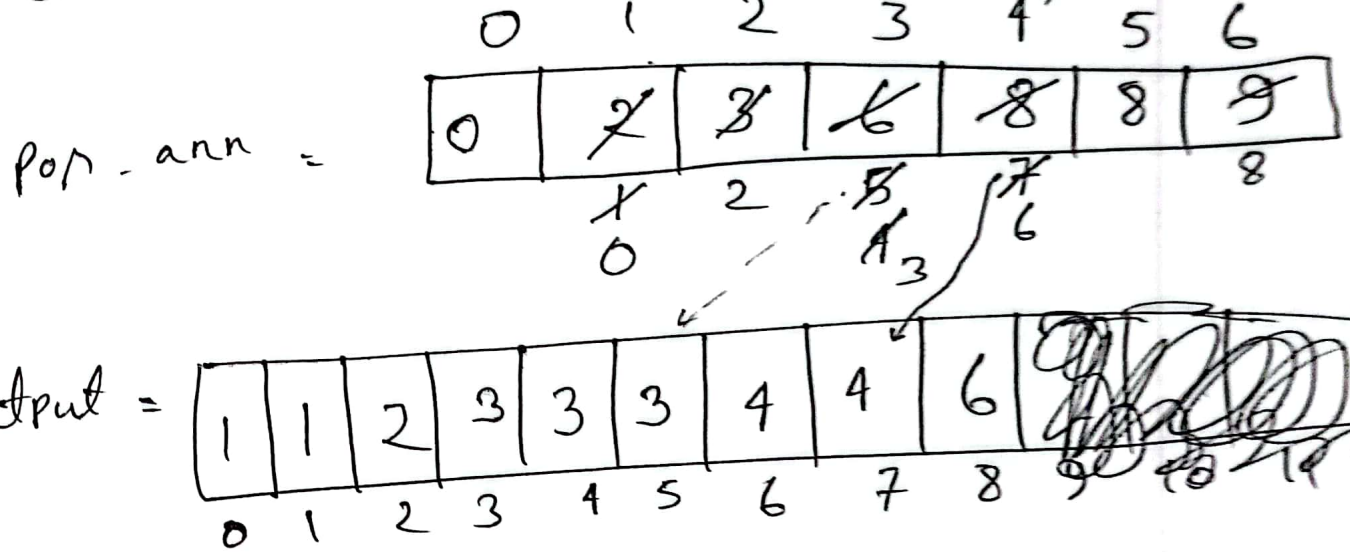
⇒ position array →
(adding with next elem)



Now for output value will will go reverse of the main array and will decrease 1 from position array of the

1, 3, 2, 3, 9, 1, 6, 9, 3

index and the result we will get after decreasing will be the index where we will put the element



Time complexcity

Algorithm	Best case	Avg. case	worst ca.
Linear search →	$O(1)$ →	$O(n)$ →	$O(n)$
Binary " →	$O(1)$ →	$O(\log n)$ →	$O(\log(n))$
Ternary " →	$O(1)$ →	$O(\log_3 n)$ →	$O(\log_3^n)$
selection sort →	$O(n^2)$ →	$O(n^2)$ →	$O(n^2)$
Bubble " →	$O(n)$ →	$O(n^2)$ →	$O(n^2)$
Quick " →	$O(n \log n)$ →	$O(n \log n)$ →	$O(n^2)$

Algo.	Best	Avg	Worst
Merge sort	$O(n \lg n)$	$O(n \log n)$	$O(n \log n)$
Count Sort	$O(n+k)$	$O(n+k)$	$O(n+k)$

—— o ——

Solving Recurrences

1. Iteration method
2. Recursion tree
3. master Theorem

merge sort : Running Time

$$T(n) = \underbrace{T(n/2)}_{\text{left\_half}} + \underbrace{T(n/2)}_{\text{right\_half}} + \underbrace{b}_{\text{constant}}\, \underbrace{n}_{\text{merging}}$$

$$\therefore T(n) = \begin{cases} b & ; \ n = 1 \\ 2T\left(\frac{n}{2}\right) + bn & ; \ n > 1 \end{cases}$$

Now,

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

$$\therefore T\left(\frac{n}{2}\right) = 2T\left(\frac{n/2}{2}\right) + b\left(\frac{n}{2}\right)$$

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

$$\therefore T(n) = 2 \times \left[2T(n/4) + b\left(\frac{n}{2}\right)\right] + bn$$

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

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

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

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

$$\vdots$$

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

Now,

$$T(1) = b$$

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

$$\Rightarrow \frac{n}{2^i} = 1$$

$\Rightarrow n = 2^i$

$\Rightarrow \log_2 n = \log_2 2^i$

$\Rightarrow \boxed{i = \log_2 n}$

$\therefore T(n) = bn + bn \log n$

(*) Master node Theorem দ্বারা এর মান বের করা যায়।

Master Theorem

General expression –

$$T(n) = aT(n/b) + f(n) \quad ; \quad a \geq 1, \; b > 1$$

and $f(n) = \Theta(n^K \log^P n)$

To solve any recursive functions time complexity we have to find (at first) –

(1) $\log_b a$ and (2) K

Formulas

Case-1: if $\log_b a > K$ then $\theta(n^{\log_b a})$

Case-2: if $\log_b a = K$;

if $P > -1$ then $\theta(n^K \log^{P+1} n)$

if $P = -1$ then $\theta(n^K \log\log n)$

if $P < -1$ then $\theta(n^K)$

Case-3: if $\log_b a < K$;

if $P \geq 0$ then $\theta(n^K \log^P n)$

if $P < 0$ then $O(n^K)$ → big O; not θ.

Examples

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

⇒ here;

$a = 2$, $b = 2$, $f(n) = \theta(1) = \theta(n^0 \log^0 n)$

$\therefore K = 0$; $P = 0$

$\therefore \log_b a = \log_2 2 = 1 > K = 0$;

$\therefore$ case-1, Thus the complexity $= \theta(n^{\log_b a}) = \theta(n^1) = \theta(n)$ (—)

(*) $T(n) = 4T\left(\frac{n}{2}\right) + n$ ($k \to 1$, $p \to 0$)

$\Rightarrow a = 4, b = 2, f(n) = n = n^1 \log^0 n$

$\therefore \log_b{}^a = \log_2{}^4 = 2 > k = 1$;

$\therefore$ complexity $= \theta(n^{\log_b{}^a})$

$= \theta(n^{\log_2{}^4})$

$= \theta(n^2)$

(*) Shortcut: For case 1 answer will be $\theta(n^{\log_b{}^a})$; we just have to see is $k < \log_b{}^a$.

Like $T(n) = 8T\left(\frac{n}{2}\right) + n \to \theta(n^3)$ ($\log_2 8 = 3$)

(*) $T(n) = 8T\left(\frac{n}{2}\right) + n^{①}\log n \Rightarrow \theta(n^3)$

second case;

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

$\Rightarrow a = 2, b = 2, f(n) = \theta(n) = \theta(n^1 \log^0 n)$

$\therefore \log_2{}^2 = 1 = k$ and $p = 0$

$\therefore$ case-2, as $p=0$, answer is $= \theta(n^k \log^{p+1} n)$

$= \theta(n^1 \log^{0+1} n)$

$= \theta(n \log n)$

④ mainly if $\log_b a = k$ and $p > -1$; we just have to multiply $\log n$ with $f(n)$.

for example —

$$T(n) = 2T\left(\frac{n}{2}\right) + n \longrightarrow \theta(n \log n)$$

$$T(n) = 4T\left(\frac{n}{2}\right) + n^2 \longrightarrow \theta(n^2 \log n)$$

$$T(n) = 4T\left(\frac{n}{2}\right) + n^2 \log n \rightarrow \theta(n^2 \log^2 n)$$

$$T(n) = 4T\left(\frac{n}{2}\right) + n^2 \log^5 n \rightarrow \theta(n^2 \log^6 n)$$

⑤ $T(n) = 4T\left(\frac{n}{2}\right) + n^2 \log n$

$\Rightarrow a = 4,\ b = 2,\ f(n) = \theta(n^2 \log^1 n)$

$k = 2,\ p = 1$

and $\log_2 4 = k = 2$

$\therefore$ complexity $= \theta(n^k \log^{p+1} n)$

$= \theta(n^2 \log^{1+1} n)$

$= \theta(n^2 \log^2 n)$ (answer)

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

$\Rightarrow a=2, b=2$; $f(n) = \theta(n^1 \log^{-1} n)$

$k=1, p=-1$

$\log_b^a = k = 1$

$\therefore$ complexicity $= \theta(n^k \log\log n)$ [as $p=-1$]

~~$= \theta(n^k)$~~

~~$= \theta(n^1)$~~

~~$= \theta(n)$~~

$= \theta(n^1 \log\log n)$

$= \theta(n \log\log n)$

(—)

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

$\Rightarrow f(n) = \theta(n \log^{-2} n)$; $k=1, p=-2<-1$

and $\log_a^b = k$

$\therefore$ complexicity $= \theta(n^k)$ [as $p<-1$]

$= \theta(n^1)$

$= \theta(n)$

(—)

Case-3 example:

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

$\Rightarrow a = 1, \quad b = 2 \quad ; \quad f(n) = \theta(n^2 \log^0 n)$

$\log_b a < K \quad ; \quad \therefore$ answer $= \theta(n^K \log^P n)$ [$P \geq 0$]

$= \theta(n^2 \log^0 n)$

$= \theta(n^2)$ ←

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

$\Rightarrow a = 2, b = 2, K = 2, P = 2$

$\log_b a < K$ and $P \geq 0$.

$\therefore$ answer $= \theta(n^K \log^P n)$

$= \theta(n^2 \log^2 n)$ → which is $f(n)$

So if $\log_b a < K$ and $P \geq 0$ then the answer is simply $f(n)$.

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

←

(*) $T(n) = 4T(\frac{n}{2}) + \frac{n^3}{\log n}$

$\Rightarrow a = 4,\ b = 2,\ k = 3,\ p = -1$

as $\log_b a = 2 < k = 3$ and $p < 0$;

answer will be $= \Theta(n^k)$

$= \Theta(n^3)$ (Ans)

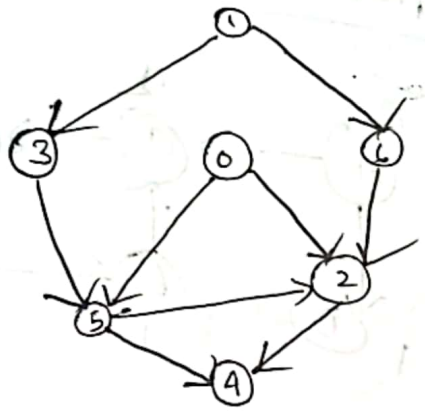
(*) Graph from slide.

or

(*) BFS, DFS Note করা।

Topological sort

* DAG → Directed Acyclic Graph.
* Topological sort can be done by both BFS and DFS but DFS is preffered.



Acceptable; T. sort

1 6 3 0 5 2 4

0 1 3 6 5 2 4

Unacceptable;

1 6 3 2 5 0 4

as 0 should come before 2.

Into steps:

* Graph directed হবে.
* ন > কোনো cycle থাকবে না
* Topological sort of a directed graph is a linear ordering of its vertices so that every directed edge UV from vertices U to V, where U comes before V.

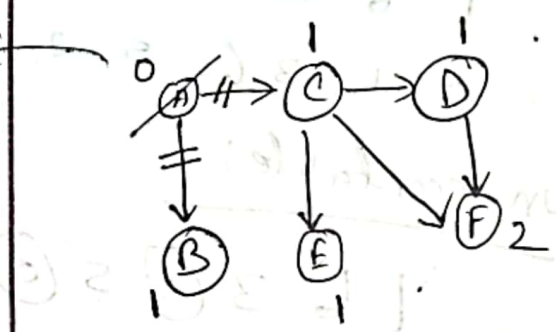
slide থেকে যেটি example + code

Method

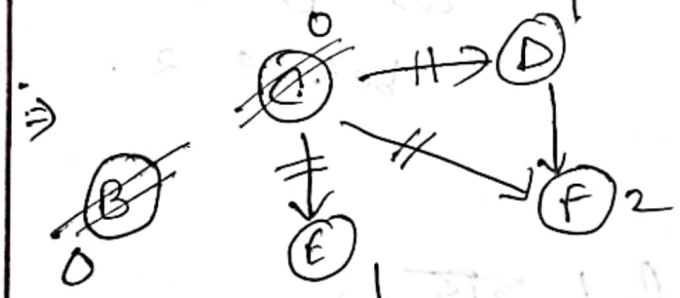
(i) প্রতিটি vertices এর indegree (no. of incoming edge) বের করতে হবে।

(ii) ৩ vertices print করতে হবে, যার indegree 0 হবে।

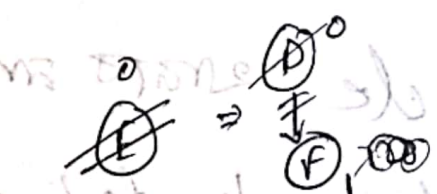
Print: A B C E D F



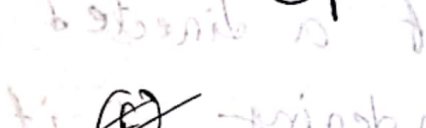
Indegree 0, এর মান Print হবে এবং vertices মুছে যাবে



⇒



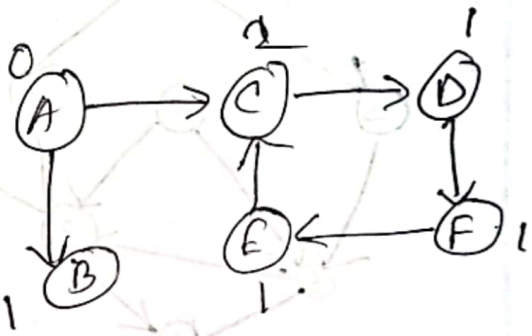
=



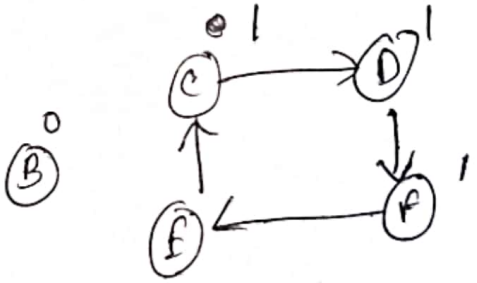
=

can take B or C any.

Print:



=

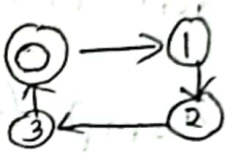


But as there is a cycle, we can't find topological sort.

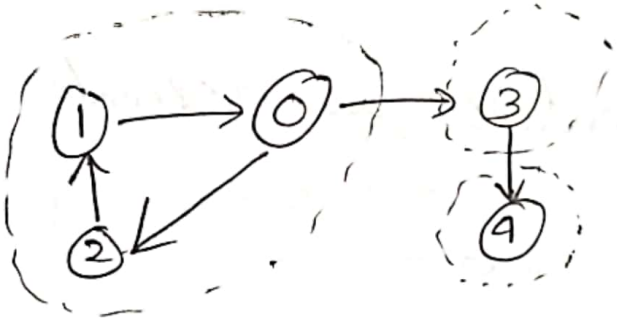
uses DFS

Strongly Connected Components

Defination: ① A directed graph is strongly connected if there is a path between all pairs of vertices.

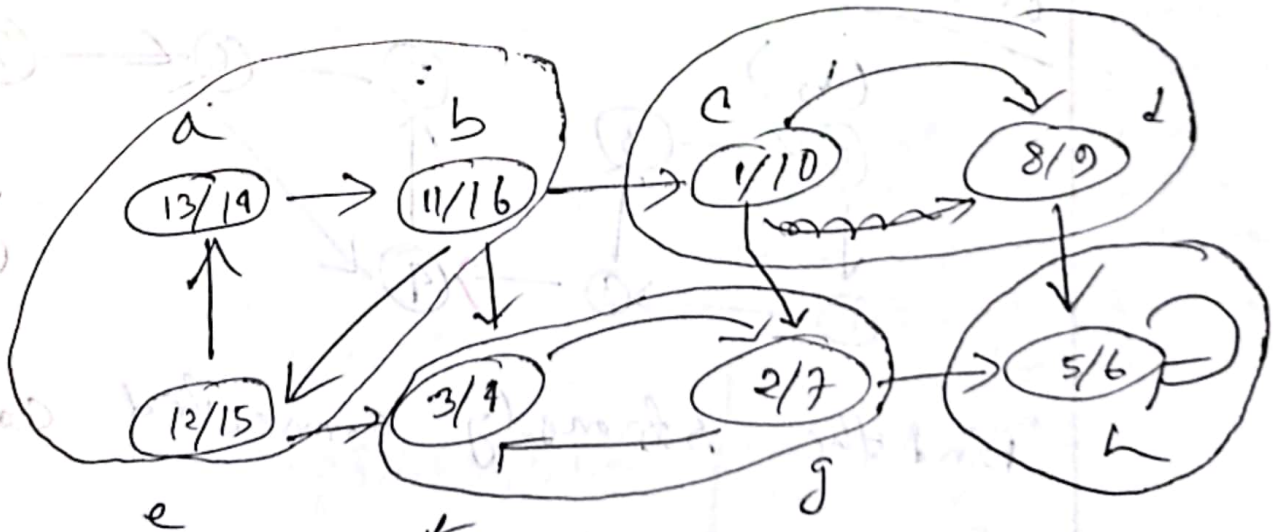


② A strongly connected component (SCC) of a directed graph is a maximal strongly connected subgraph.



strongly connected comp. -

1) 1 → 0 → 2
2) 3
3) 4



Kosaraju's idea: $\Theta(V+E)$

1) Call DFS(G) to compute finish time for each vertex.

2) Compute G^T → 

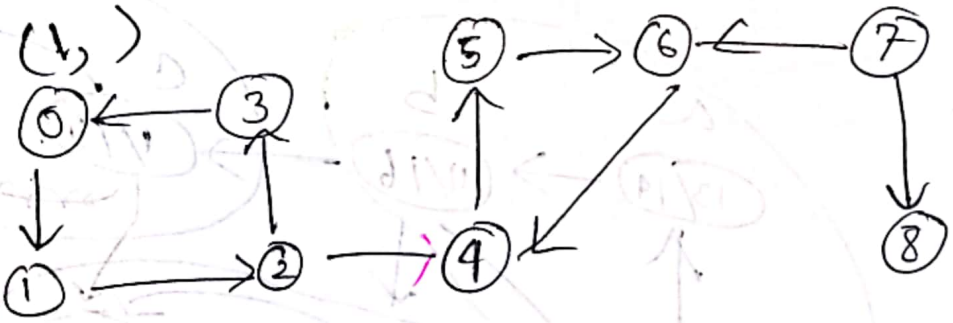
3) Call DFS(G^T) on v in decreasing order of their finish times.

G ના બધા જ v ની direction reverse

4) output : vertices as separate SCC's.

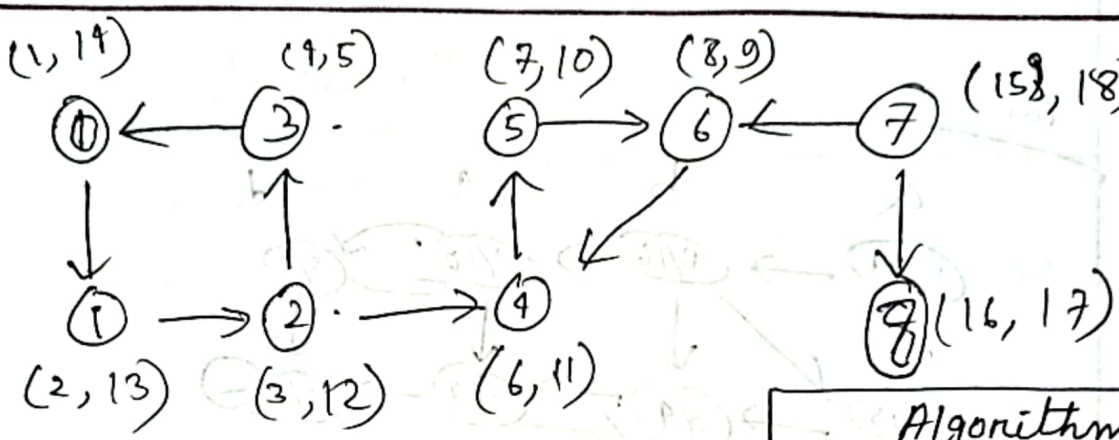
— o —

question)

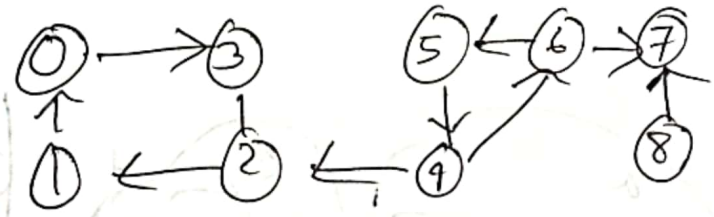


find the strongly connected component.

⇒ First we have to find start and end time.



⇒ Then perform G^T.



⇒ we will start from (7) as it has highest end time.

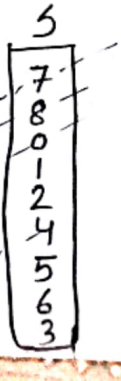
End time	Pop	Visited
		{7}
18	7	{7, 8}
17	8	{7, 8, 0, 3, 2, 1}
14	0	{7, 8, 0, 3, 2, 1, 4, 6, 5}
11	4	

∴ SCC:

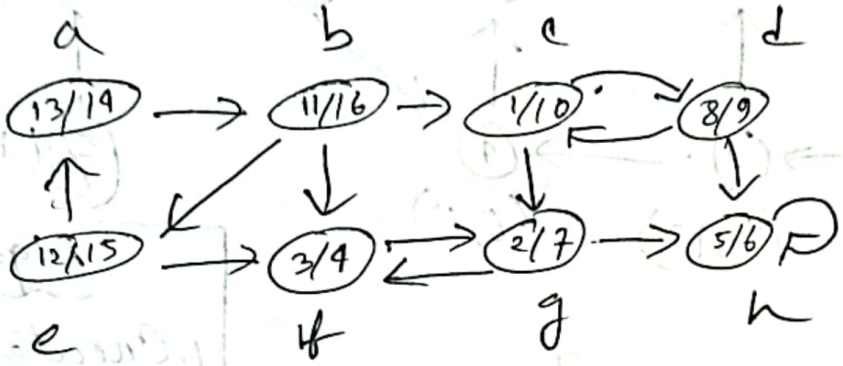
1) 7
2) 8
3) 0 — 3 — 2 — 1
4) 4 — 6 — 5

Algorithm

1. Create an empty stack 'S'.
2. Do DFS traversal of a graph. In DFS traversal, after calling recursive DFS for adjacent vertices of a vertex, Push the vertex to stack.
3. Reverse directions of all arcs to obtain the transpose graph.
4. One by one pop a vertex from S while S is not empty. Let the popped vertex be 'v'. Take v as source and do DFS call on V. The DFS starting from v prints SCC of V

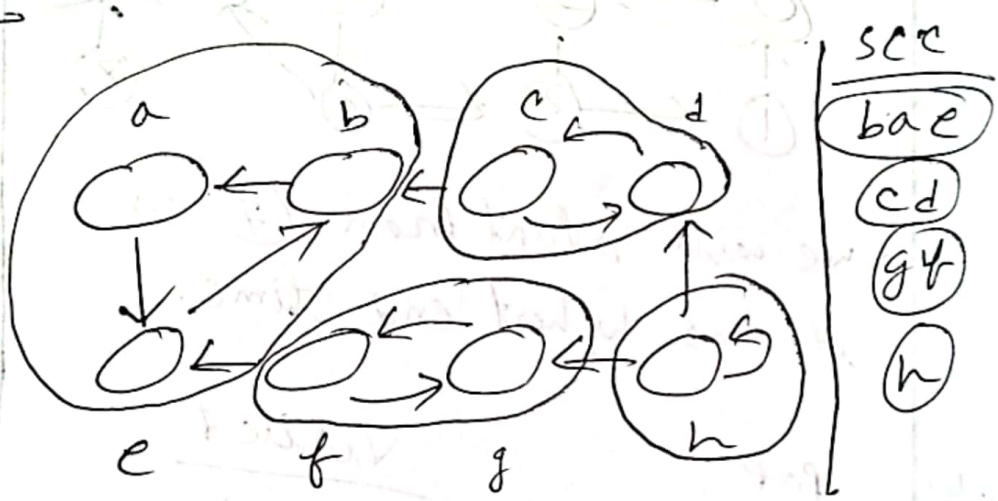


Q1)

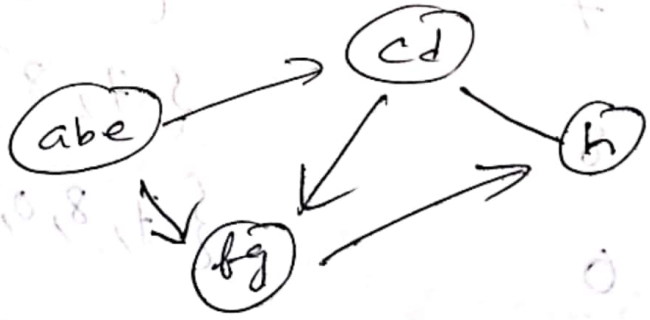


⇒ Decrease order (Finish Time)

16
15
14
10
9
7
6
4



⇒



Shortest Path Algo → (Dijkstra, bellman ford)

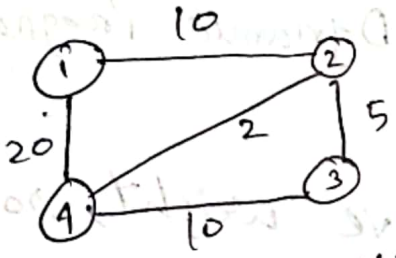
We will discuss 2 general case algo.

1) Dijkstra's Algo → (Positivite edge weights only)

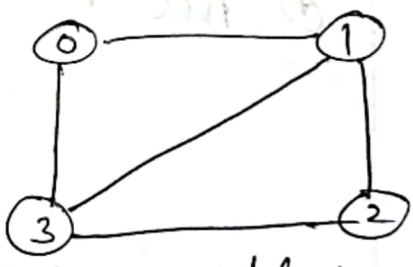
2) Bellman-Fond Algo → (Positive and (-)ve. edge weights)

If all ~~eg. edges~~ weights are same (lets say 1), then we can solve it by BFS in $\Theta(V+E)$ time.

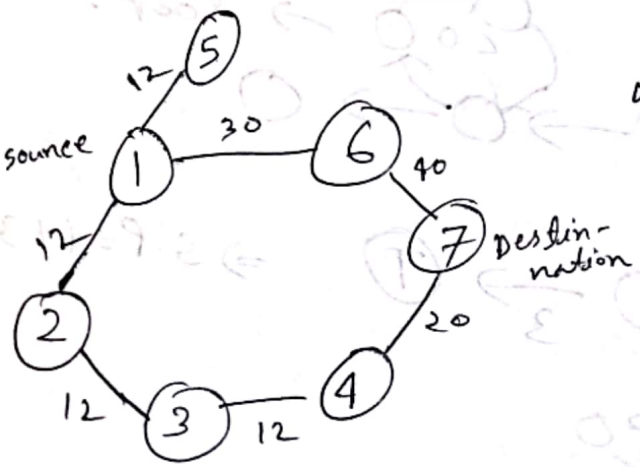
—o—



weighted positive



Unweighted graph



weighted shortest path:

1, 2, 3, 4, 7

Unweighted shortest path:

1, 6, 7

* shortest path = a path with minimum weight.

Problems can be like -

* single source → S.P from a given source to each of the vertices.

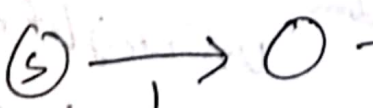
* single destination → S.P from to a given destination vertex from each node.

* single pair → Given 2 V, find the shortest path between them.

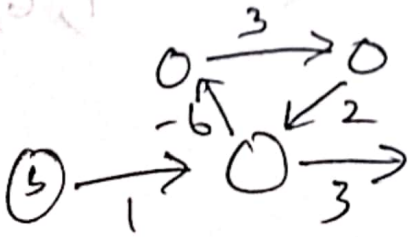
* All pairs → Find S.P from every pair of vertices. (Dynamic Programming).

In graph with (-)ve weight, some shortest path will not exist. explain.

→ 



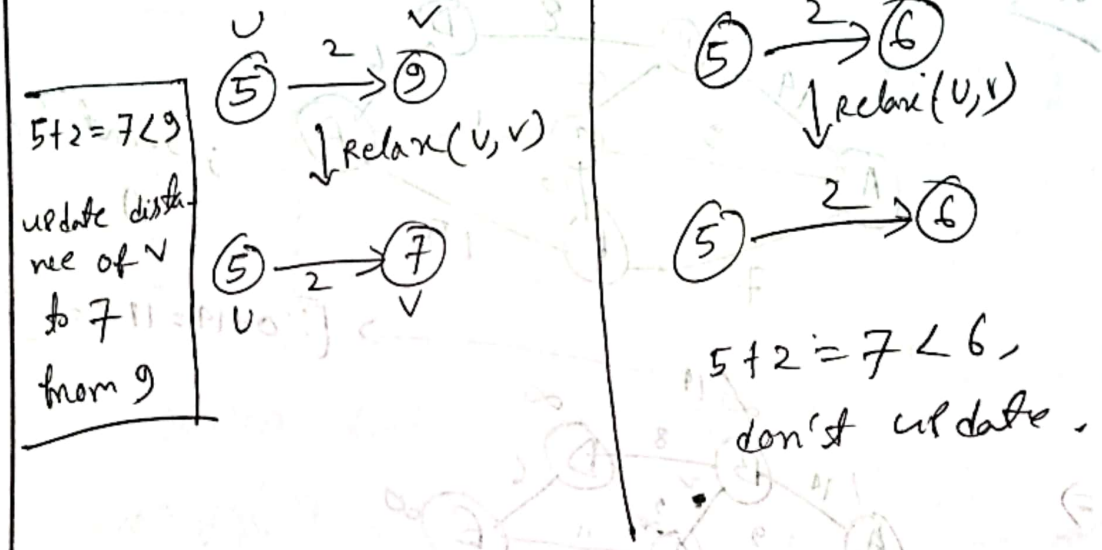
$\Rightarrow$ S.P = 1 + 3 = 4



$\Rightarrow 1-6+3+2-6+3+2+3 = 2$

✱ Relaxation :

```
Relax(U,V,W){
    if (d[V] > d[U] + W[U,V])
        then d[V] = d[U] + W(U,V);
}
```



Dijkstra's Algorithm (BFS)

✱ very similar to BFS, instead of queue we use min heap.

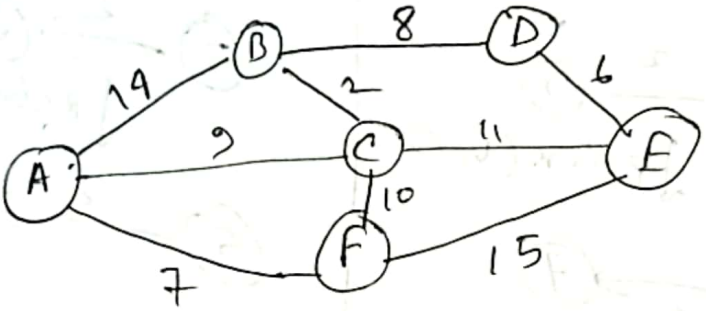
✱ ⓐ যে point থেকে শুরু করবো সেটি নাও (suppose A) তার সকল ∞ মান দিবো and check করবো A এর সকল ~~vertices~~ vertices গুলোর $d[V] > d[U] + W(U,V)$ কিনা, যদি হয় তবে $d[V] = d[U] + W(U,V)$.

Dijkstra's for Undirected Graph

If ($d[v] > d[u] + w[u,v]$);

$$d[v] = d[u] + w(u,v)$$

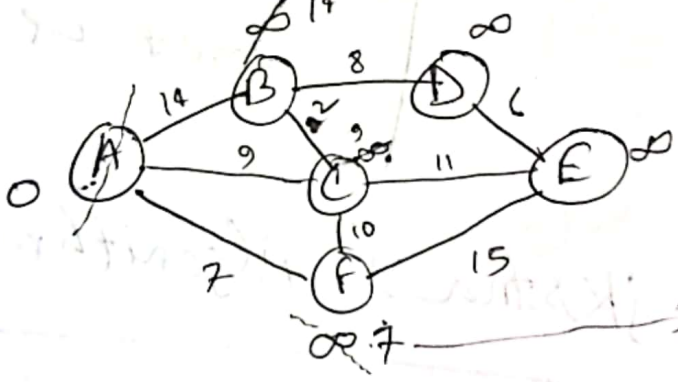
Q.



; start from A.

$\rightarrow$ [$\because 0+14 = 14 < \infty$]

$\Rightarrow$



$\rightarrow$ [$0+7 = 7 < \infty$]

A থেকে on visited করবে, then যে vertex এর মান সবচেয়ে কম তার থেকে Relaxation start করবো, But পরবর্তী simulation থেকে ঠিক system betten.

যে vertex হয়ে যাবে ওর column এ এর পর আর ওর value নিতাম না,

22 কারণ F এর value ছিল 7, so, 7+15=22<∞

Visited vertex	A	B	C	D	E	F
A	0	∞	∞	∞	∞	∞
F		14	9	∞	∞	[7]
C		14	[9]	∞	22	
B		[11] (9+2)		∞	20 (9+11)	
D				[19] (11+8)	20	
E					[20]	

as A relax এর পর সব থেকে ছোট value 7, F এর.

Now suppose Ques দিলো A থেকে E তে যাওয়ার shortest path কোনটি ?

⇒ যেটি destination থাকবে ওর selected ঘরে যাবো, মানে যে ঘরে যে □ হয়েছে, এ ক্ষেত্রে [E] = [20] ঘর,

Then উপরের দিকে উঠবো, value same থাকলে continue করবো, Change হলে ওর উপরের Row এর ওপর থেকে ছোট element এ Pointer দিবো,

	A	B	C	D	E	F
A	0	∞	∞	∞	∞	∞
F		14	9	∞	∞	[7]
C		14	[9]	∞	22	
B		[11]		∞	20	
D				[19]	20	
E					[20]	

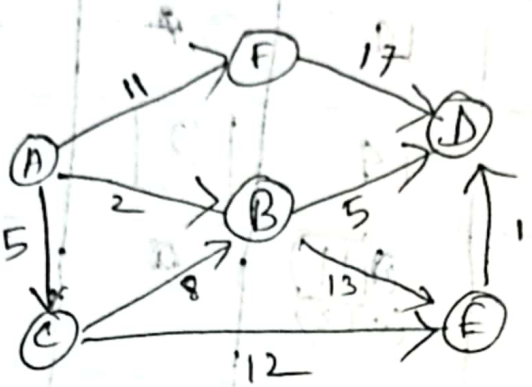
উল্টো way E → C → A

∴ Final answer, A → C → E

Some 7

Dijkstra's for Directed graph

Q



; start at A.

⇒

visited vertices	A	B	C	D	E	F
A	0 ←	∞	∞	∞	∞	∞
B		2 ←	5	∞	∞	11
C			5	5+2 = 7	13+2 = 15	11
D				7	15	11
F					15	11
E					15	

সুতরাং দেখা যাচ্ছে A থেকে D এ যাওয়ার shortest path arrow দিয়ে দেখানো হলো।

7 → 7 → 2 → 0 : A → B → D

D → D → B → A

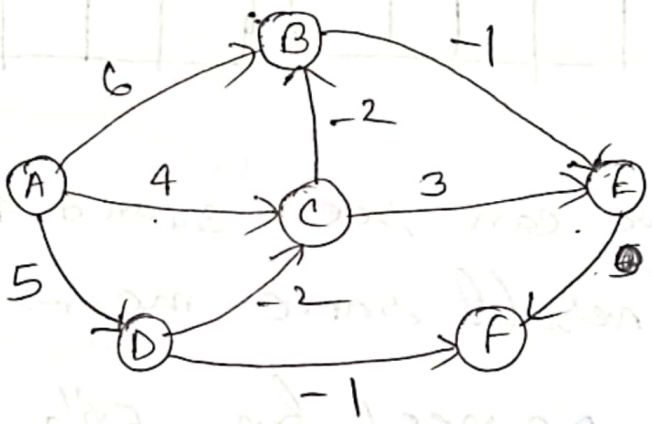
Bellman - Ford Algorithm

(*). Can work with negative weight.

⇒ Go on relaxing with all edges.

Iteration: max iteration needed, $n(n-1)$ [n = no. of vertices]

Q:



; start from A.

⇒ $n(n-1) = 6-1 = 5$ বার iteration করে max লাগবে।

Serially করে

1st iteration

	A	B	C	D	E	F
A	[0]	∞	∞	∞	∞	∞
B	0	[6]	[4]	5	∞	∞
C	0	6	[4]	5	5	∞
D	0	2	4	[5]	5	∞
E	0	2	3	5	[5]	4
F	0	2	3	5	5	4

Final row after 1st iter.

2nd iteration

	A	B	C	D	E	F
A	[0]	2	3	5	5	4
B	0	[2]	3	5	5	4
C	0	2	[3]	5	1	4
D	0	1	3	[5]	1	4
E	0	1	3	5	[1]	4
F	0	1	3	5	1	[4]

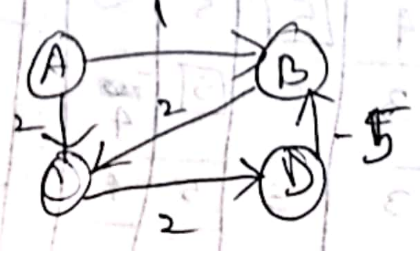
3rd

	A	B	C	D	E	F
A	[0]	1	3	5	1	4
B	0	[1]	3	5	1	4
C	0	1	[3]	5	0	4
D	0	1	3	[5]	0	4
E	0	1	3	5	[0]	4
F	0	1	3	5	0	[3]

	A	B	C	D	E	F
A	[0]	1	3	5	0	3
B	0	[1]	3	5	0	3
C	0	1	[3]	5	0	3
D	0	1	3	[5]	0	3
E	0	1	3	5	[0]	3
F	0	1	3	5	0	[3]

but as we can see 3rd and 4th iteration so final result same. means amra answer পেয়ে গেছি. No need for 5th iteration.

<u>Backdraw</u>: If there is a cycle between some vertices, if sum of weights of them is negative, we can't find shortest path by using bellman-ford.



B - C - D ⇒

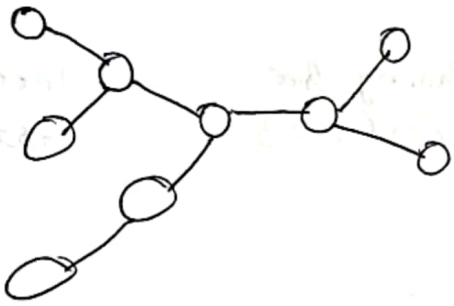
$2+2+(-5) = -1$

we can't find.

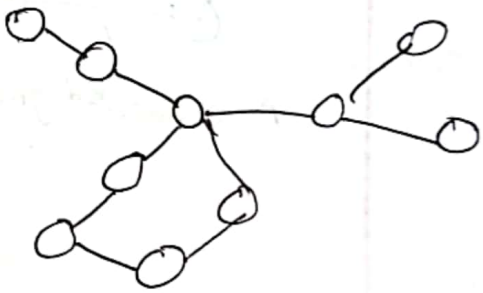
MST (Minimum Spanning Tree)

Trees: Undirected graph G is tree if -

1. G is connected
2. G does not contain a cycle
3. G has n-1 edges.



Tree

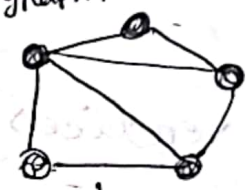


NOT Tree

Difference between Tree and spanning Tree

- A tree is a type of graph. A spanning is a subgraph of the graph that is a tree and hits every vertex.

This graph:



has a spanning tree

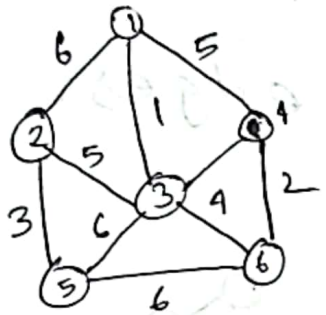
whearas the subgraph

is not a spanning tree (it's tree, not spanning). The subgraph -

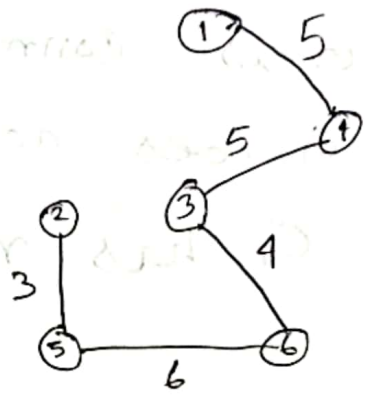
is not a spanning tree (its spanning but not tree).

MST

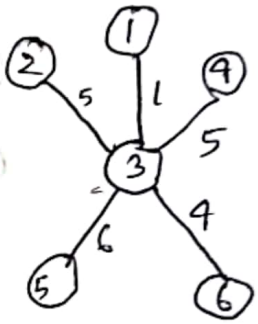
- minimum spanning tree is the spanning tree where the cost is minimum among all the spanning tree.



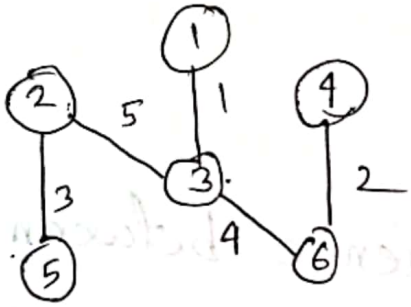
A connected graph



A spanning tree with cost = 23



S. Tree with cost = 21

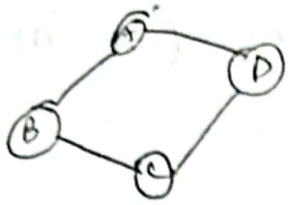


M.S.T cost = 15 (3+5+1+4+2)

Rules:

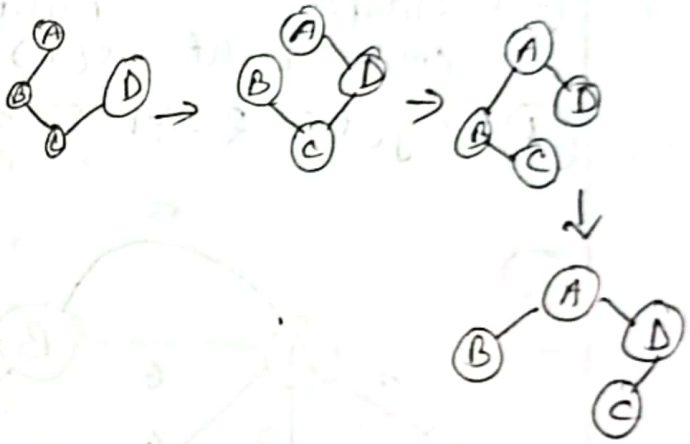
1. Subset of a graph
2. NO cycle
3. NO disconnected vertices
4. New edge can't be add/deleted.

(ঘ)



Total sub.graph which can be made is, $\boxed{{}^{E}C_{E'}} = {}^{4}C_{3}$

$= 4$



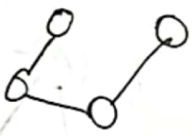
ধরি, graph, $G(V, E)$

$\therefore$ subgraph, $S(V', E')$

where,

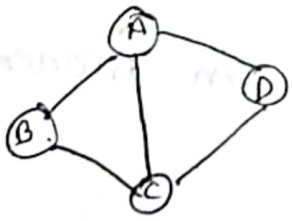
$\boxed{\begin{aligned} V' &= V \\ E' &= |V| - 1 \end{aligned}}$

like,



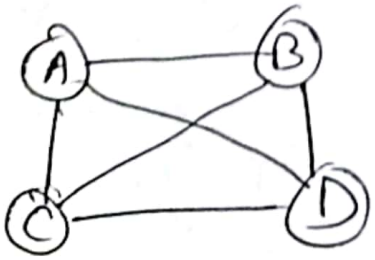
$E' = |4| - 1 = 3$

(ঘ) যদি cycle থাকে তবে, $\boxed{{}^{E}C_{E'} - \text{no. of cycle}}$



$= {}^{5}C_{3} - 2 = 8.$

(ঙ) Complete graph:



$E = 3$

Remove edge

$\boxed{E - V - 1}$

$\Rightarrow 6 - 4 + 1$

$\Rightarrow 2 + 1$

$\Rightarrow 3$

(সর্বনিম্ন 3টা edge রাখলেই হবে)

No of MST

$\boxed{V^{V-2}}$

$\Rightarrow 4^{4-2}$

$\Rightarrow 4^{2}$

$= 16$

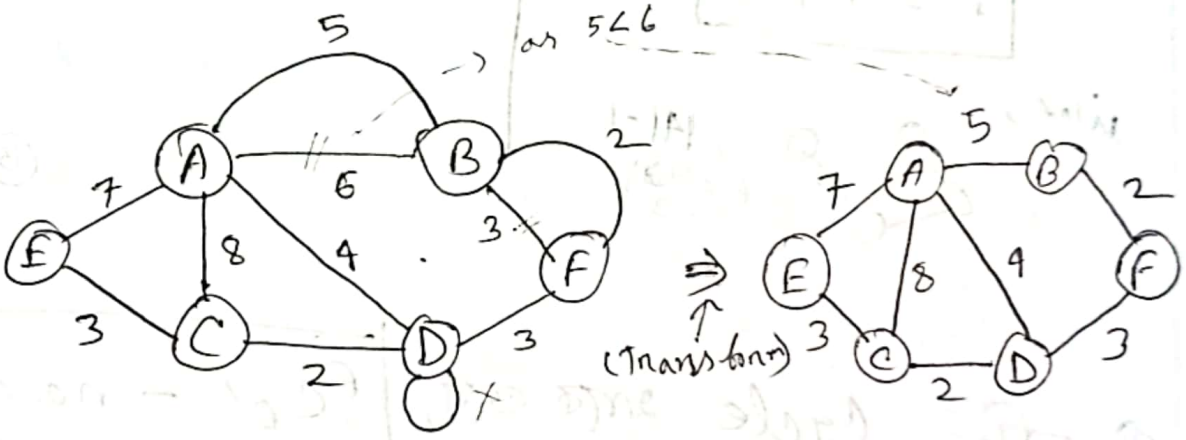
Kruskal's Algorithm (for MST)

(1) Graph থেকে loop delete করতে হবে, (self loop)

(2) Graph থেকে parallel edge delete করতে হবে, (min weight রাখতে হবে).

(3) Cycle হওয়া যাবে না, নতুন MST এর জন্য.

Q

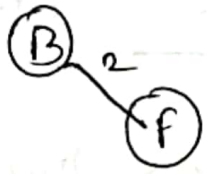


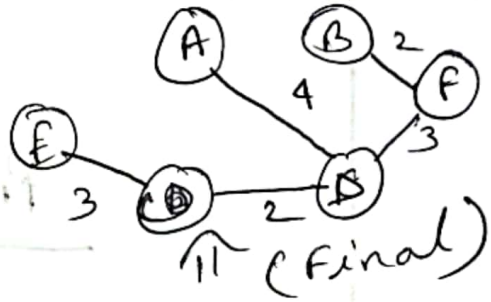
Now we will work on this ↗.

Now we will take edges in increasing order.

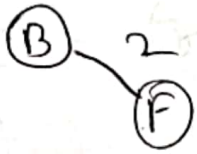
BF → 2
CD → 2
DF → 3
CE → 3
AD → 4
AB → 5
AE → 7
AC → 8

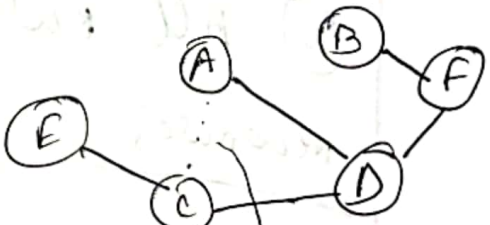
Now we will start plotting in increasing order one by one. Try to draw it ~~while~~ in a way so that it matches the real Graph (নিচে সুবিধার জন্য). (cycle যাতে না হয়).



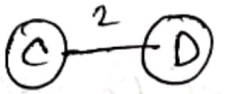


⇒ (Final)

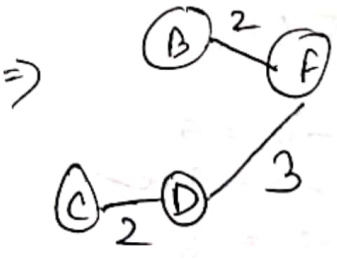




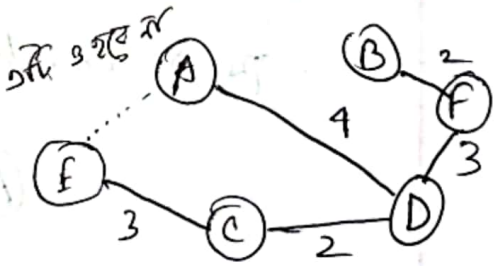
এটি ও হবে না।

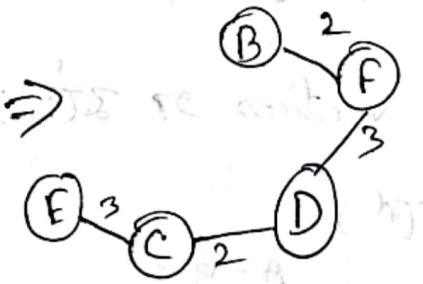


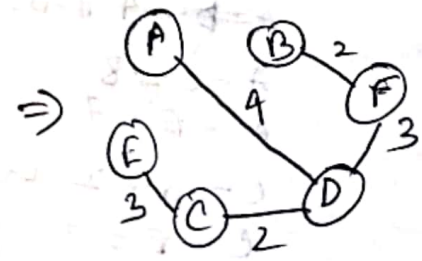
BF → 2
CD → 2
DF → 2
CE → 3
AD → 4
AB → 5 X
AE → 7 X
AC → 8 X



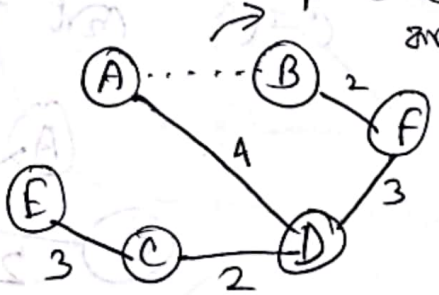
এটি ও হবে না







⇒



A-B হবে না, কারণ cycle হয়ে যাবে। তাই এটি আমরা নিব না

Let's see if it is really a MST.

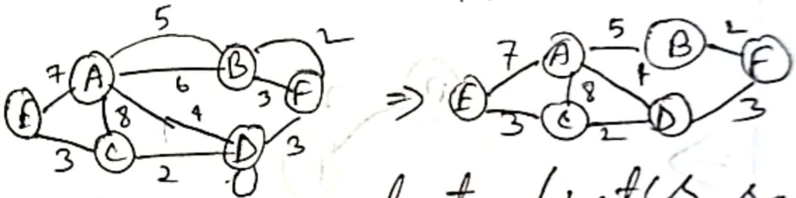
$G(V, E)$ ⇒ $V' = V = 6$

$S(V', E)$ ⇒ $E' = E - 1 = 5$ ⇒ matched

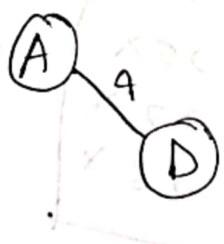
Prim's Algorithm (for MST)

(*) All Rules are same as Kruskal's.

New rule;

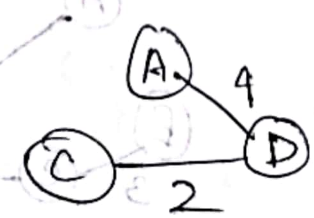


① যেকোনো একটি Node থেকে start (Let's say A) করে A থেকে যত বের হয়েছে তাদের মধ্যে যে গুলোর weight সবথেকে কম তাকে নিবো।



A – B = 5
A – D = 4 ✓
A – C = 8
A – E = 7

② A এবং D দুইটিরই out vertices এর মধ্যে সব থেকে ছোট টি select করবো।



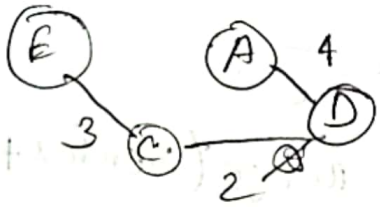
already done

A – B = 5
~~A – D = 4~~ A – D = 4 ✗
A – C = 8
A – E = 7
C – D = 2 ✓
C – E = 7

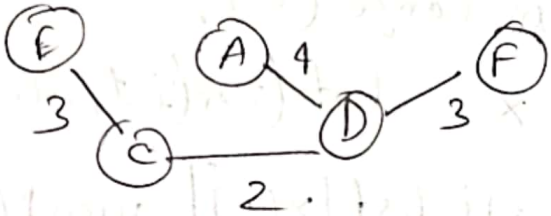
③ এখন A, D, C এর Edge গুলো Minimum বের করি।

..... min হবে CE = 3
DF = 3

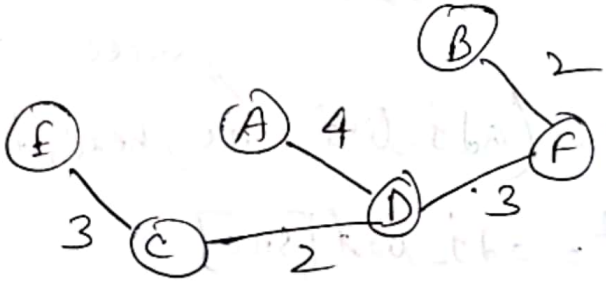
we can choose any one. lets take C-E



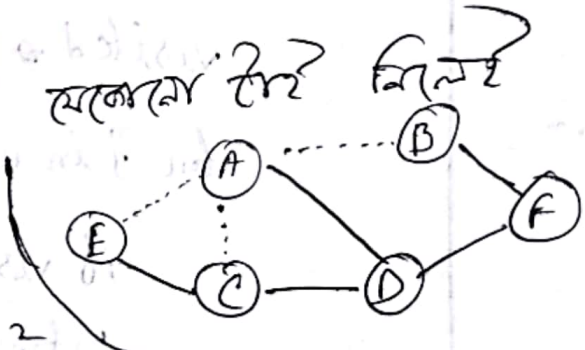
④ A, D, C, E এর মধ্যে সবথেকে ছোট D-f = 3.



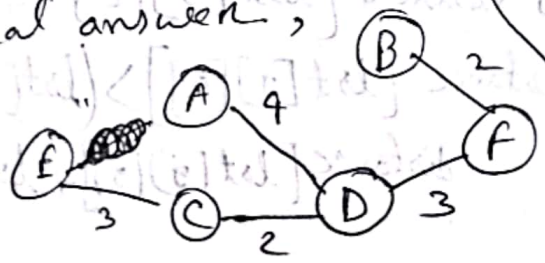
⑤ A, D, C, E, f এর মধ্যে, B-f = 2



⑥ এখন যদি edge 3 টার যেকোনো টাই নিলেই cycle create হবে,



so final answer,



(c)

Greedy

* BFS (adj. list

Huffman Coding

* Compression technique
* Used to reduce the size of data or message.

[Extra details slides এ আছে।]

* Huffman Coding is a form of statistical coding.
* In a text file not all the characters occur with the same frequency.

In ASCII all chr will be 8bit. means fixed size.

But in huffman, ~~we will~~ code word lengths vary and will be shorter for the ~~same~~ more frequently used characters.

B B B C A A A D E E E A B B B A C C D A

In ASCII (8bit):

size = 20 × 8 = 160 bit

In fixed length code;

As we have only 5 chr (A, B, C, D, E)

code word bit we need → 2^n

$2^2 = 4 < 5$

$2^{(3)} = 8 > 5$ ✓

we need atleast 3bit code word to represent all 5 chr separately.

Chr.	Count / Frequency	code (Random 5 code word)
A	6	0 0 0
B	6	0 0 1
C	3	0 1 0
D	2	0 1 1
E	3	1 0 0

Size

5 × 8 (ASCII) = 40 bits + (5 × 3) bits = 55 bit

so, to send the size of the table to P.C so that it know how to decode it we need to send a instruction of 55 bits.

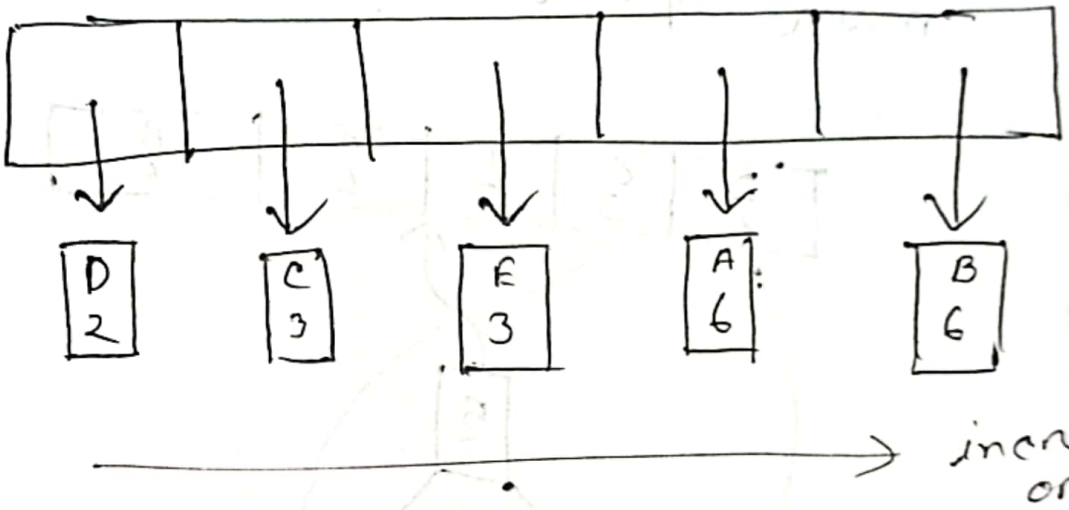
And bits for the string will be = 20 × 3
= 60 bits

∴ Total bits = (55 + 60)
= 115 bits < ASCII 160 bits
(Reduced)

** In Variable length code:

Chr	Freq
A	6
B	6
C	3
D	2
E	3
	20

Now we have to make a Priority queue in a the increasing order.
(The lower the occurrence, the higher the Priority in the queue.)



Now we have to build a binary Tree.

Building a Tree

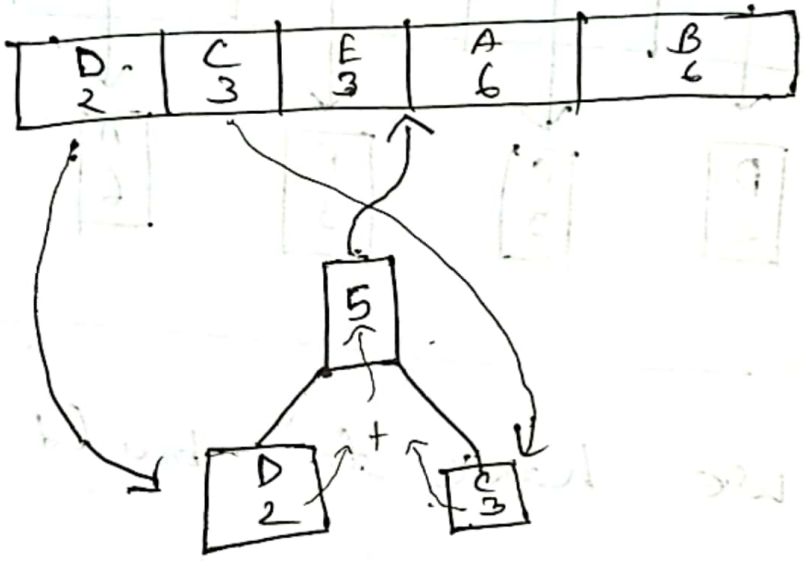
While Priority queue contains 2 or more nodes —

① create new node

② Dequeue ~~node~~ 2 nodes from the front of the queue, in serial make the first node into left subtree " " 2nd " " right "

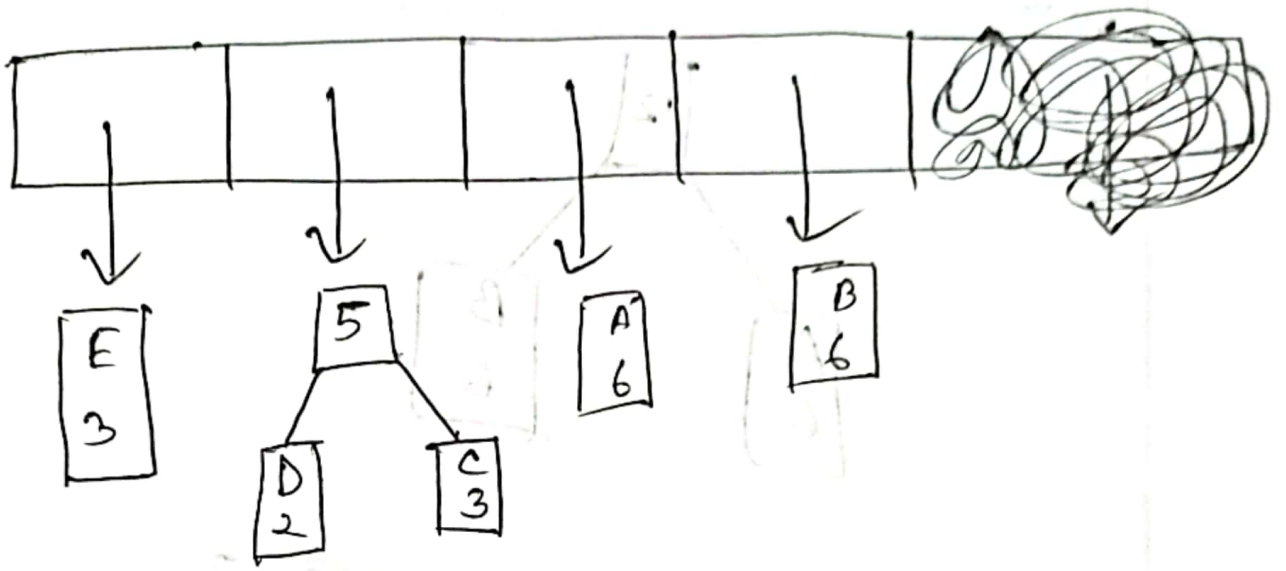
(3) frequency of the node equals sum of frequency of left and right child.

(4) Enqueue new node back into the main queue.

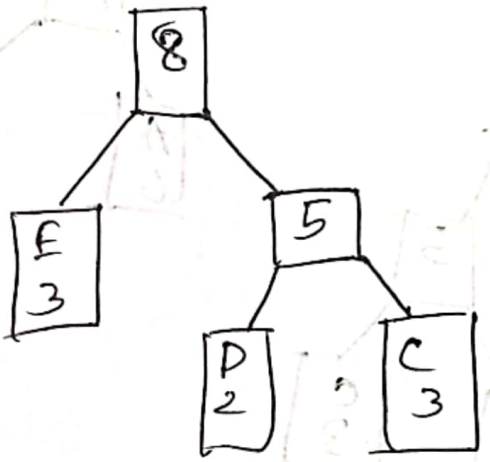


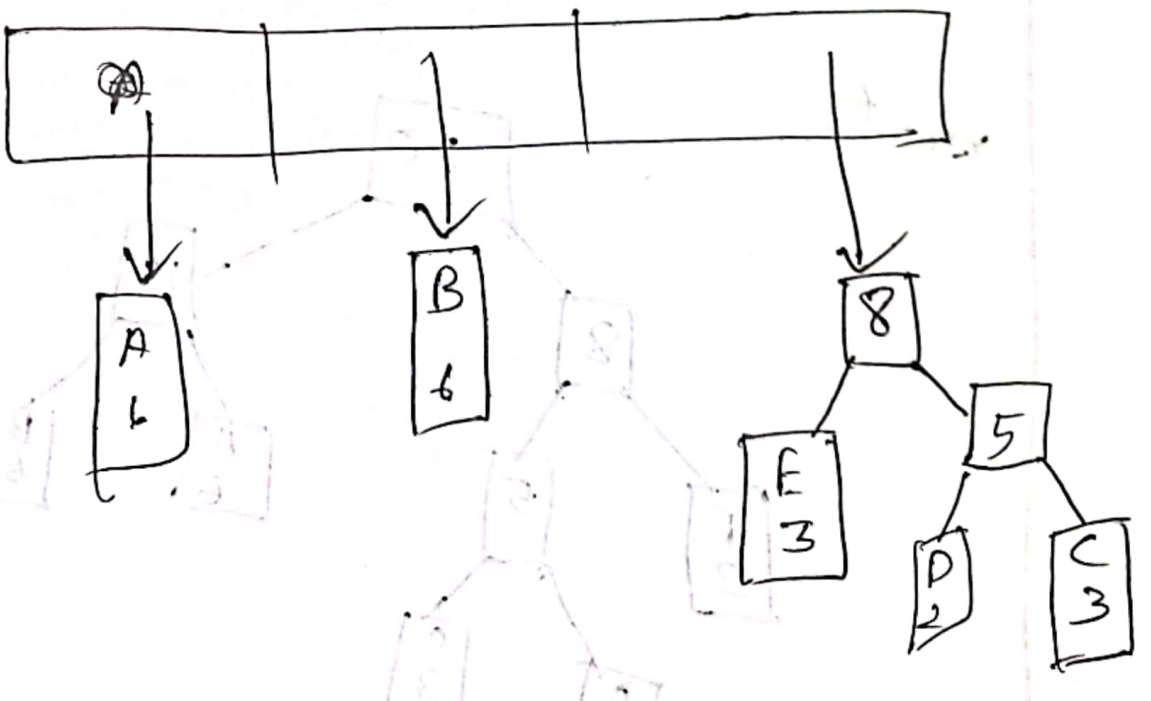
after de-que D and C, D is left and C is right child of new node. and freq of new node is 2+3 = 5. as

3 < 5 < 6, we will put it back between E and A.

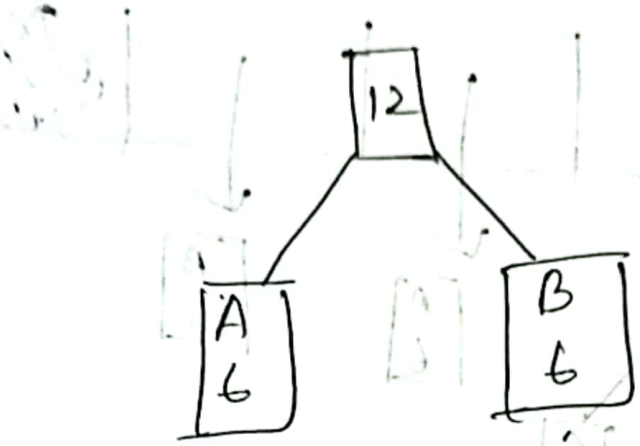


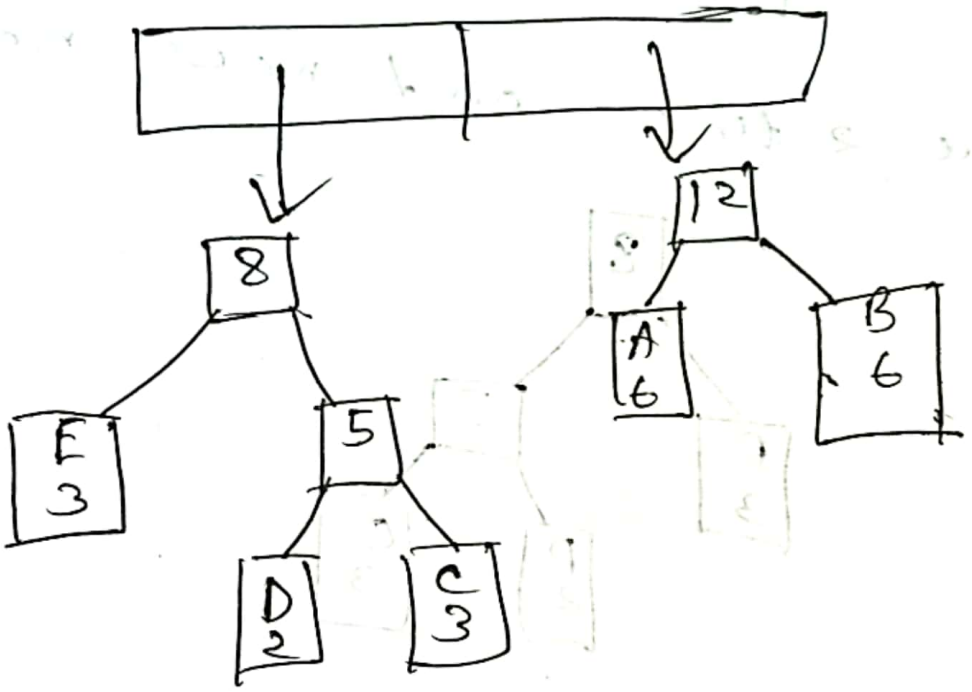
Dequeue 2 time and new node —



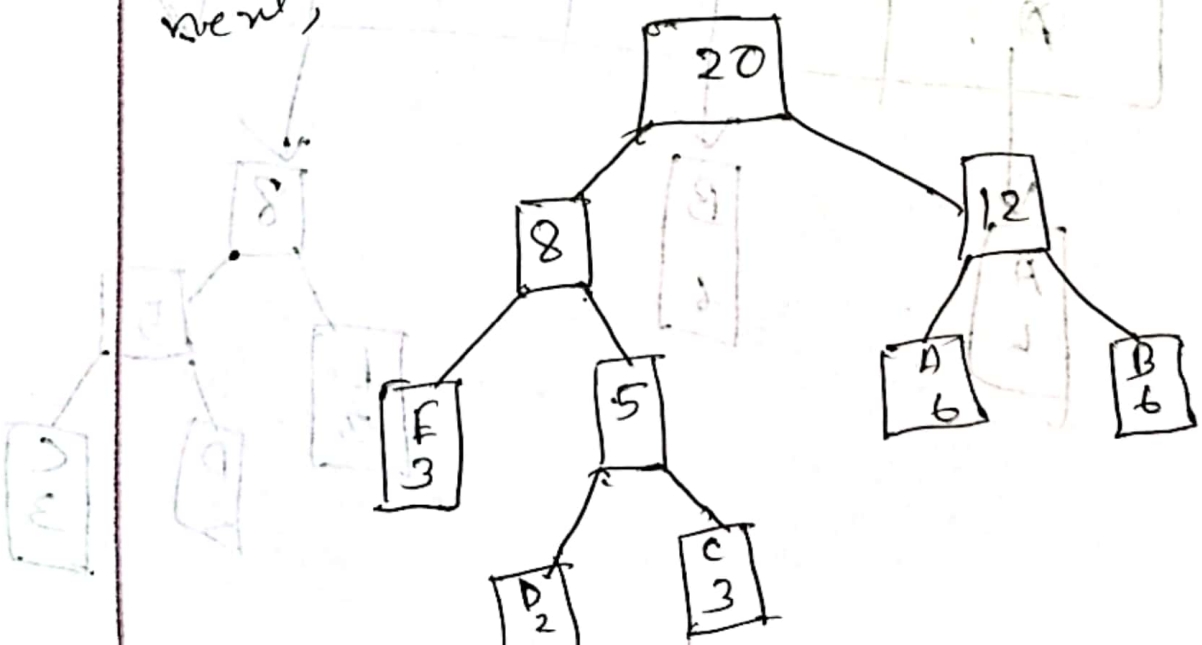


next;

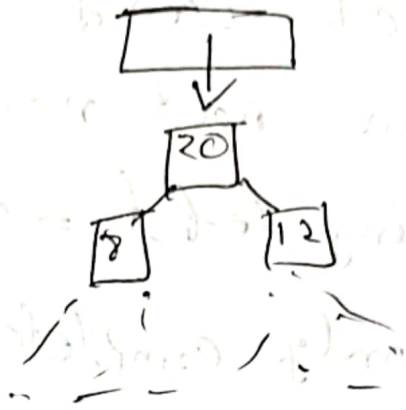




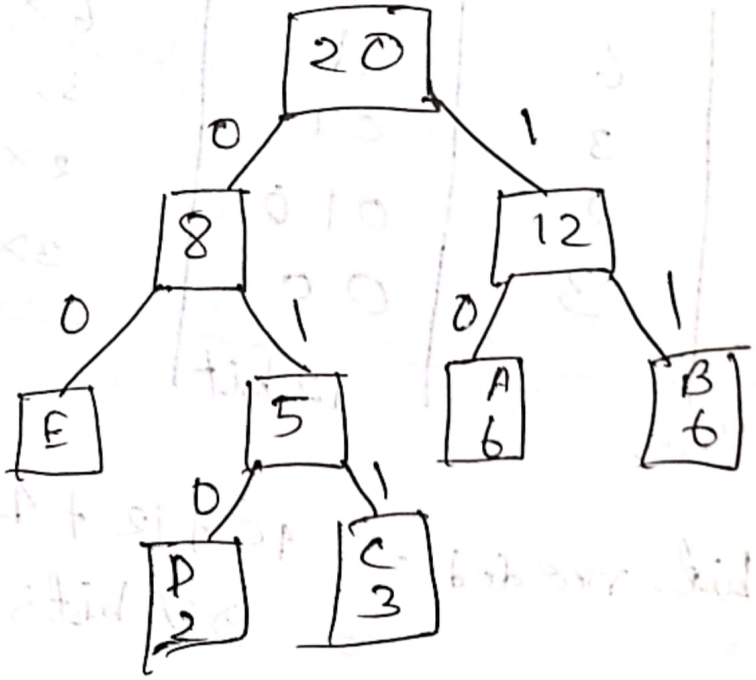
next;



→



as only one element in the queue, we have found the final tree. Now every left ~~subtree~~ hand we will mark 0 and for every right hand we will mark 1.



Now for code word we will start from root node [20], then for going left = 0, right = 1. Code word is only completed when a leaf node is reached,

like for B,

$$20 \xrightarrow{1} 12 \xrightarrow{1} B$$

∴ Codeword of B will be = 11

Chr	Freq	code	Bit needed for every single chr in text.
A	6	10	6 × 2 = 12
B	6	11	6 × 2 = 12
C	3	011	3 × 3 = 9
D	2	010	2 × 3 = 6
E	3	00	3 × 2 = 6
5 × 8 = 40 bits		12 bit	45 bits

∴ Total bits needed = 10 + 12 + 45

= 97 bits < 115 < 160

B B B C A A A D E E E A B B B A C C D A

code word ⇒ 11 11 11 011 10 10 10 010 00 - - - ——— ———

— o —

Greedy

(*) At each step of solution, pick the best choice given the info currently available.

(*) Often leads to very efficient solutions to optimization problems..

However not all problems have greedy solutions.

(*) If a problem can be solved by greedy method, it should have 2 properties —

(1) Greedy choice property (If local optimal solution leads to global optimal soln)

(2) sub problem optimality (If optimal soln. of global problem contain optimal soln. of sub problems)

fractional knapsack ($T(n) = O(n \lg n)$)

Given a set S of n items, such that each item i has a positive benefit b_i and a positive weight w_i, the goal is to find the maximum-benefit subset that does not exceed a given weight W, allowing for fractional items.

since we are maximizing the benefit, select the next item with the highest benefit per weight $\Rightarrow \boxed{\frac{b_i}{w_i}}$

* In fractional knapsack in greedy method we can take fraction of a object. like in product A is 2Kg and its value = 30$, we can take 1Kg of A at 15$.

Quer:
Below 7 objects and their weight, Profits is given. You have to take object such that you earn the most Profit in 15 kg. means, you have to take total of 15kg objects and Profit should be maxed.

objects :	1	2	3	4	5	6	7
Profit (P) :	12	5	16	7	9	11	6
weight (W) :	3	1	4	2	9	4	3

⇒ value index $\left(\frac{P}{W}\right)$:

4	5	4	3.5	1	2.75	2

we have to pick that object 1st which have the highest value index.

Like here object 2 has values index 5 which is highest. So we will take obj-2. It will fill up 1 kg.

weight left in bag = (15 − 1) = 14 kg.

If we do it until bag is full —

object	Profit	weight	Remaining weight
2	5	1	15 - 1 = 14
1	12	3	14 - 3 = 11
3	16	4	11 - 4 = 7
4	7	2	7 - 2 = 5
6	11	4	5 - 4 = 1
7	$6 \times \frac{1}{3} = 2$	1	1 - 1 = 0
	∴ Profit = 56	W = 15 kg	

Decreasing value index order

As 6 Profit for 3 kg, and we are taking only 1 kg.

∴ We will get highest Profit 56 if we take items in the above serial.

Reuse-able code for longer scenario.

Dynamic Programming ($T(n) = \theta(n)$)

* Details slide ১ দেখো, mainly use recursion.
* Memoization ⟶ slide.
  ⟶ (bottom up)
* DP is top-down.

0/1 Knapsack

Given a set S of n items, such that each item i has a positive benefit v_i and a positive weight w_i, the goal is to find the maximum-benefit subset that does not exceed a given weight W.

Ques:

items/objects:	1	2	3	4
weight:	3	2	5	4
value/Profits:	4	3	6	5

$n = 4$

max weight, $W = 4$ 5

B, W

i ↓ \ W →	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	4	4	4
2	0	0	3	4	4	7
3	0	0	3	4	4	7
4	0	0	3	4	5	7

not same W

same value নিয়ে নাও

∴ max Profit will be 7. and items will be

1	2	3	4
1	1	0	0

∴ 1 and 2 is the objects.

Formula to check —

$$B[i, W] = \max\left(B[i-1, W], B[i-1, W-w[i]] + v[i]\right)$$

$$\therefore B[4,5] = \max\left(B[3,5], B[3,5-4] + 5\right)$$

$$= \max\left(B[3,5], B[3,1] + 5\right)$$

$$= \max(7, 0+5)$$

$$= \max(7, 5)$$

$$= 7$$

Process:

(1) 0 row ও 0 column ও 0 বসাবো।

(2) 1 no. item ও weight যাবে তত no. column ও আগের ঘর থুলে উপরের row ও value copy/paste করবো।

i ↓ / W →	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0			

Profit

③

(1, 3) এর value হবে 1 item এর 4 + (1-1, 0) = 4 + B[0, 0] = 4 + 0 = 4

[1, 4] এর value = 1 এর Profit 4 + B[1-1, 1] এর val

= 4 + 0

= 4

[1, 5] = 4 + B [1-1, 2]

4 + 0 = 4.

④ Same করে বাকি এর মুল্যের জন্য করবো,

i ↓ \ w →	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	4	4	4
2	0	0	3+[1,0] = 3+0 = 3	3+[1,1] = 3+0 = 3 ④	3+[1,2] = 3+0 = 3 ④	3+[1,3] = 3+4 = 7

but এর উপরের মানের value 4 > 3, we have to keep 4.

Ques:

$W = 9$

$w_i = \{2, 3, 4, 5\}$

$v_i = \{3, 4, 5, 7\}$

B ↓ \ W →	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	3	3	3	3	3	3	3	3
2	0	0	3	4	4	7	7	7	7	7
3	0	0	3	4	5	7	8	9	9	12
4	0	0	3	4	5	7	8	10	11	12

∴ max Profit = 12.

1	2	3	4
1	1	1	0

0 0 1 1

on,

Longest Common Subsequence (LCS)

we can't make 'X' only '1'.

X = E A B C D
Y = B A C D E ⇒ BCD

⇓

X = E A B C D
Y = B A C D E ⇒ ACD

B, B is there but if we connect B there will be a X.

⇓

X = E A B C D
Y = B A C D E ⇒ CD

common sub-sequence

and longest common sub.sequence = BCD, ACD
length (3)

—o—

X = PRESIDENT
Y = PROVIDENCE ⇒ PRIDEN = 6

⇓

X = PRESIDENT
Y = PROVIDENCE ⇒ RIDEN = 5

∴ LCS = PRIDEN.

Rule:

(1)

	0	P
0	0	0
P	0	↖ 1

→ If P, P match we have to put value = diagonal + 1 = 0 + 1 = 1 and put a arrow towards diagonal.

(2)

	0	R
0	0	0
P	0	← 0 ↑

If value doesn't match as $P \neq R$, we have to put — max(উপরের val, বামের val) যেদিক থেকে বড় সংখ্যা নিবে সেদিকে arrow.

Q:

$X = A\ B\ C\ G$

$Y = A\ B\ D\ C\ A\ G$

$m = |X| = 4$

$n = |Y| = 5$

∴ array = $c[5, 4]$

i		0	1	2	3	4	5
	Y_i		B	D	C	A	G
0	x_i	0	0	0	0	0	0
1	A	0	0 ↑	0 ↑	0 ↑	↖ 1	← 1
2	B	0	↖ 1	← 1	← 1	1 ↑	1 ↑
3	C	0	1 ↑	1 ↑	↖ 2	← 2	← 2
4	G	0	1 ↑	1 ↑	2 ↑	2 ↑	↖ 3

∴ len = 3 , LCS = B C G

only diagonal ~~go~~ গুলো নিবো,

Q:

X = A B C B D A B

Y = B D C A B A

X ↓ \ Y →	0	B	D	C	A	B	A
0	0	0	0	0	0	0	0
A	0	0↑	0↑	0↑	↖1	←1	↖1
B	0	↖1	←1	←1	1↑	↖2	←2
C	0	1↑	1↑	↖2	←2	2↑	2↑
B	0	↖1	1↑	2↑	2↑	↖3	←3
D	0	1↑	↖2	2↑	2↑	3↑	3↑
A	0	1↑	2↑	2↑	↖3	3↑	↖4
B	0	↖1	2↑	2↑	3↑	↖4	4↑

∴ LCS = B C B A = len (4)

এখান থেকে start করলাম

অথবা এখান থেকে start করলে

LCS = B C A B