Algorithm

Final - 20-21

1. (a) Mark → 3

Time complexity measure the amount of time that an algorithm takes to run as a function of the size of the input (denotes as n) It analyze the effieiney of an algorithm.

- Here is a clear difference between worst case best case and average case.

Aspects	Worst Case	Best Case	Average Case
Description	Maximum time taken for any input	Minimum time taken for any input	Expected time over all inputs
Notation	Big O (O)	Omega (Ω)	Theta (Θ)
Bound	Upper Bound	Lower Bound	Tight Bound.
Example	Element at last position or not found → O(n)	Element at the first position → O(1)	Element may in middle position → O(n) .

Mid 20-21

[Mark-3]

(1) A convex hull is the smallest convex polygon that covers all the given point in a plane.

Algorithm of BFS:

```
for each vertex u in G.V:
    u.color = white
    u.d = ∞
    u.π = nil
s.color = gray
s.d = 0
s.π = nil
create empty queue (Q = ∅)
Enqueue (Q,s)
while Q ≠ ∅:
    u = Dequeue(Q)
    for each v in G.Adj[u]
        if v.color = white
            v.color = gray
            v.d = u.d + 1
            v.π = u
            Enqueue(Q,v)
    u.color = Black.
```

(2) [Mark-2]

Sequencees are "ABCDGcH" and "AEDFHR" where LCS result of "ADH" and length 3.

```
def lcs(x,y):
    m = len(x)
    n = len(y)
    L = [[0]*(n+1) for _ in range(m+1)]
    for i in range(1, m+1):
        for j in range(1, n+1):
            if x[i-1] == y[j-1]:
                L[i][j] = L[i-1][j-1]+1
            else:
                L[i][j] = max(L[i-1][j], L[i][j-1])
    lcs_str = ""
    i,j = m,n
    while i>0 and j>0:
        if x[i-1] == y[j-1]:
            lcs_str = x[i-1] + lcs_str
            i -= 1
            j -= 1
        elif L[i-1][j] > L[i][j-1]:
            i -= 1
        else:
            j -= 1
    return L[m][n], lcs_str
```

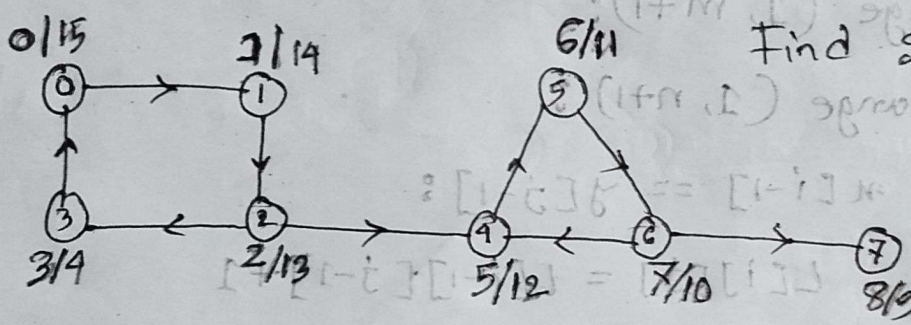
x = "ABCDGH"

y = "AEDFHR"

~~length, se~~

```
len, seq = lcs(x,y)
print ("LCS Length: ", len)
print ("sequence: ", seq)
```

(3)

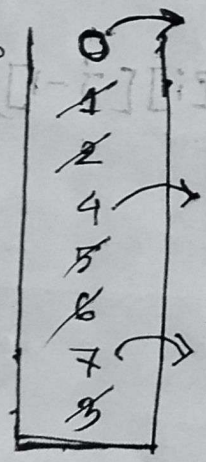


Find see from graph. [3]

step:1

step:2

DFS:



POP and check transpose graph

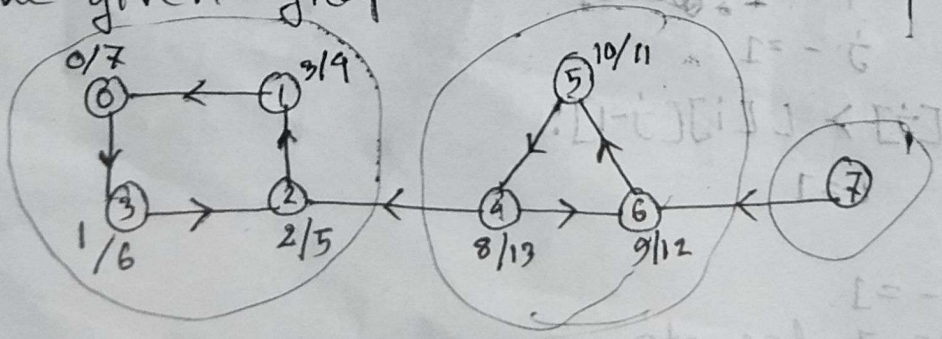
i) 3 2 3 0

ii) 5 6 4

iii) 7

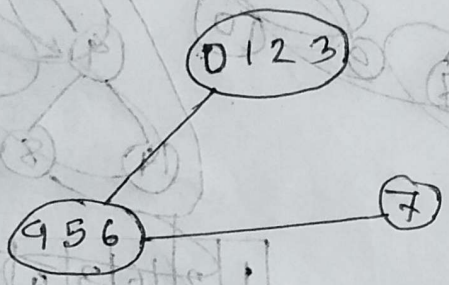
* Directed graph হতে হবে।
* cycle থাকবে এবং প্রত্যেক Node থেকে প্রত্যেকটি Node এ travars করা যাবে আবার Backtrack করা যাবে।

Now make the transpose graph of the given graph.



strongly connected components are (1238), (564), (7)

strongly connected components graph,



Mid - 19 - 20%

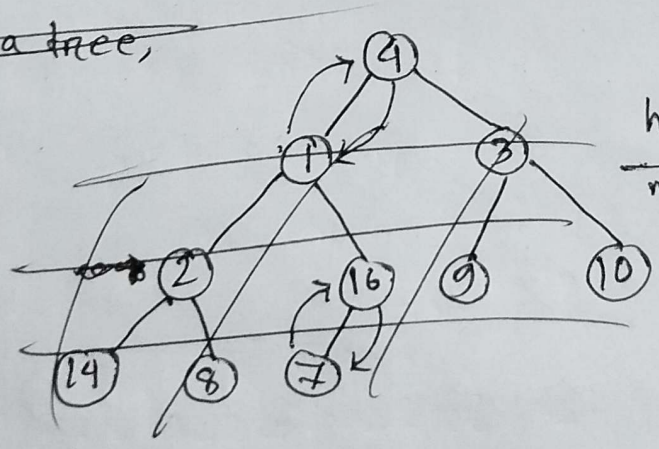
1. What do you mean by heap? ~~sto~~ sort ~~cost~~ using heap sort $A = \langle 4, 1, 3, 2, 16, 9, 10, 14, 8, 7 \rangle$ [mark-9]

A heap is a complete binary tree data structure where each node has a value greater than or equal to its child node (in max heap) or less than or equal (in min heap).

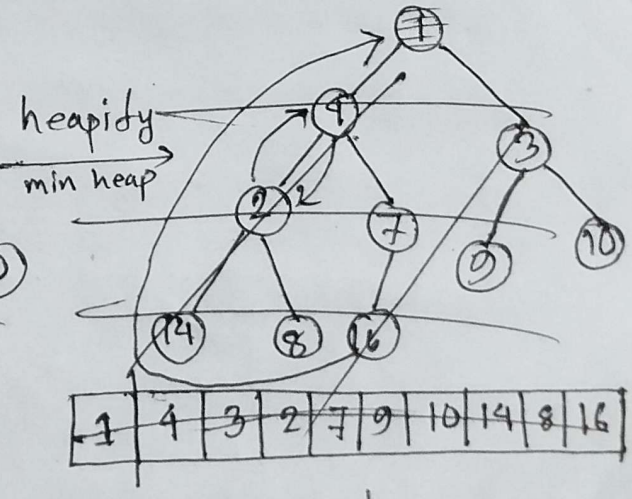
~~Given array~~ A =

4	1	3	2	16	9	10	14	8	7

~~create a tree,~~

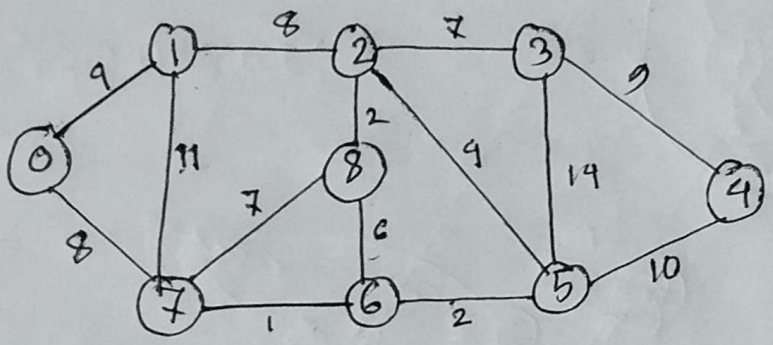


heapify
min heap



20-21

(5) What do you mean by MST? Find the MST for the following graph using prim's algo—. Is prim's a Greedy algo— [3]



A Minimum Spaning Tree (MST) connects all the points (nodes) in a graph with the least total weight/and it's ~~do~~ graph doesn't have any ~~loops~~: cycle (loops).

graph part → next page.

Yes, Prim's ~~al~~ algorithm is greedy because it always chooses the smallest possible edge to grow the tree step by step.

Graph: 1. Delete loops (if any)
2. Delete parallel edge (if any)
3. Haven't any cycle in new MST.

In new MST:

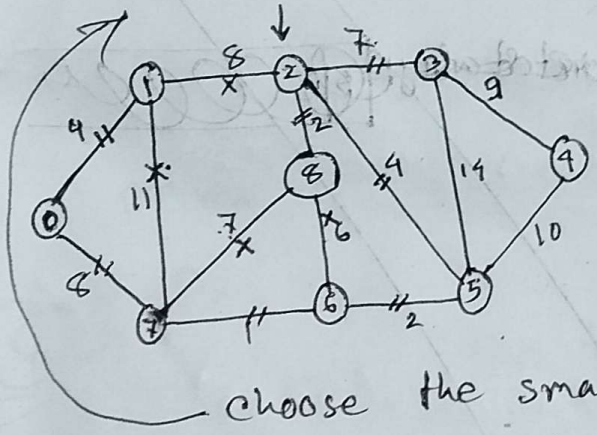
Vertex, $V' = V = 9$

Edge, $E' = V - 1 = 9 - 1 = 8$

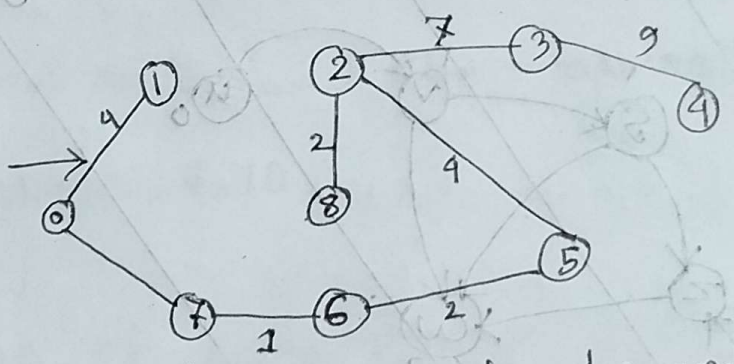
~~Delete loops and parallel edge.~~

Now, Deleting loops, cycle and parallel edge.

Given that



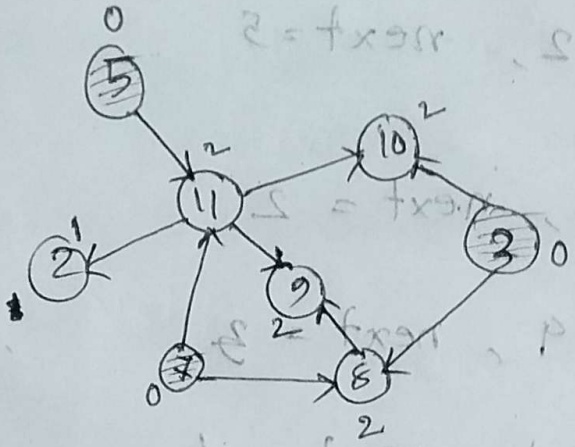
choose the smallest weight among 8 nodes and making the graph.



It is the final graph, where there is no cycle, no loops and no parallel edge.

19-20

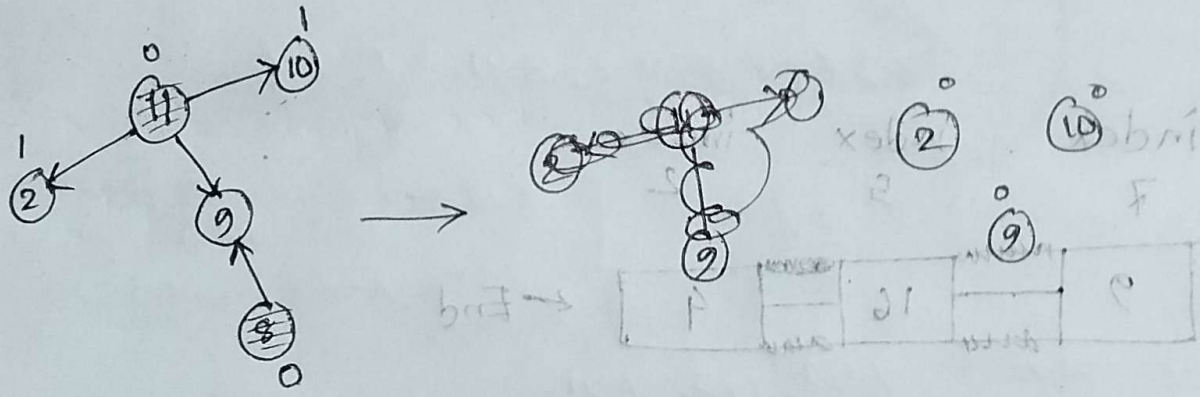
(5)



step1: Find the indegree of all nodes.

step2: Remove the node having indegree = 0.

step3: Repeat 1 and then 2



∴ This is the topological sort and the topological order is 3, 5, 7, 8, 11, 2, 9, 10

(6) 19-20

We use this type of representation instead of pointer-based linked list, those in environment where pointers are not allowed. This types of representation simulates a doubly linked list using arrays.

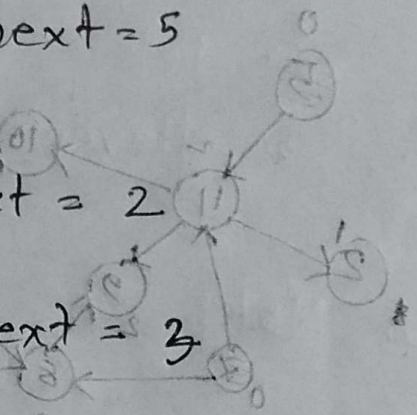
The given fig. shows three arrays next, key and prev, where each index represents a node in a doubly linked list.

L = 7 (head) → : key = 2, next = 5

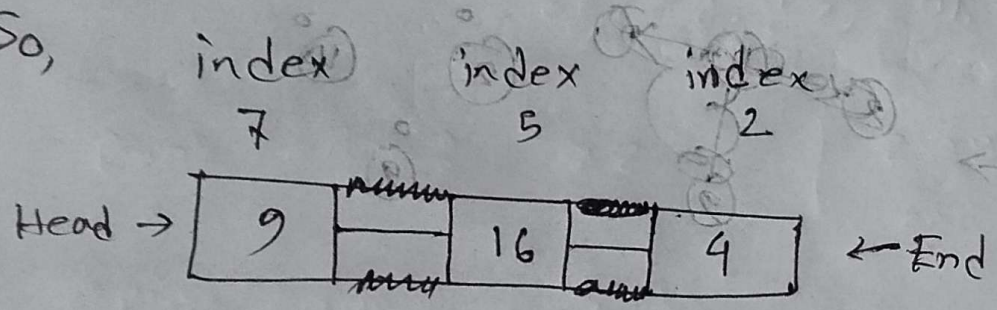
(i) Go to 5 → key = 16, next = 2

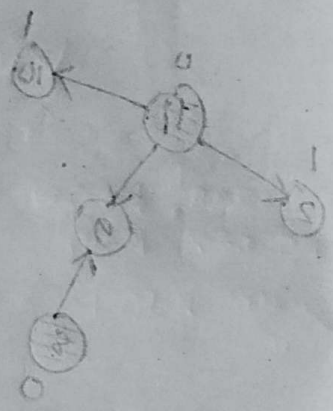
(ii) Go to 2 → key = 4, next = 3

(iii) Go to 3 → key = 1, next = 1.



So,



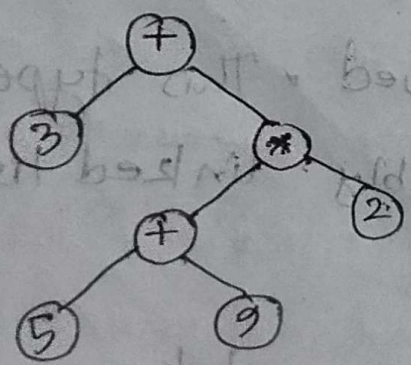


~~Rules: The given~~

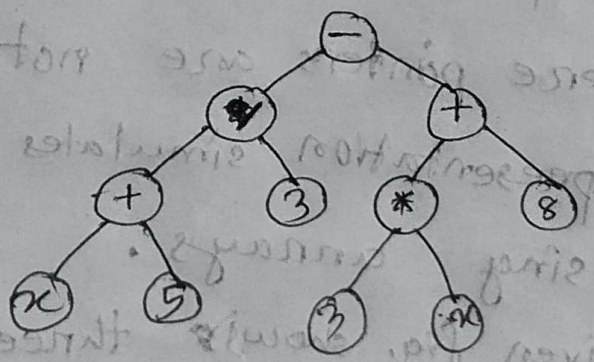
(9) 19-20

Now draw the expression tree from the follow ——

(i) 3 + ((5+9)*2)



(ii) ((x+5)/3) - (3x+8)



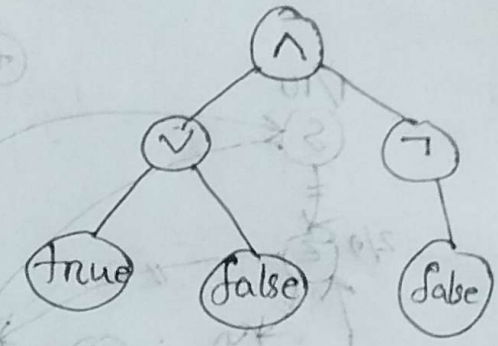
(iii) (true ∨ false) ∧ ¬false

here,

∨ = OR

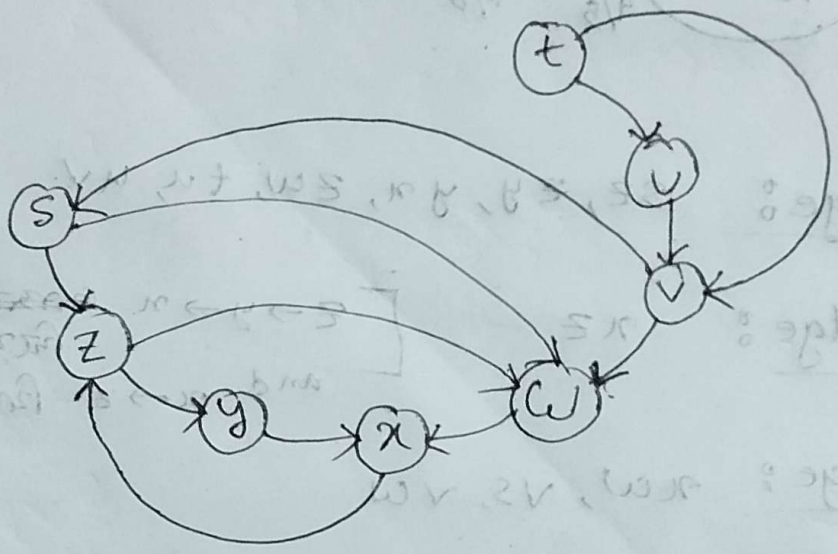
∧ = AND

¬ = NOT



(2) 19-20

(1)



If we want to perform topological sort we remember that:

(i) It will be a directed graph

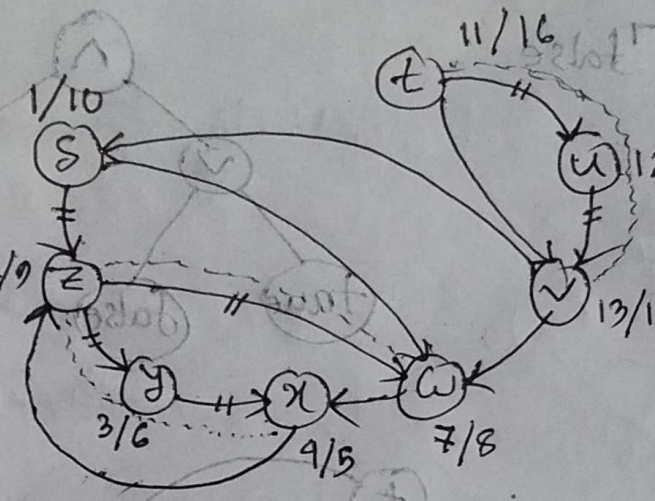
(ii) There will be no cycle.

But in this graph there are two cycle present.

(i) $z \to y \to x \to z$ and (ii) $z \to w \to x \to z$

Having those cycle we can't perform topological sort of those node. So there is no topological sort of this graph.

(ii)



Here:

Tree edge: sz, zy, yx, zw, tu, uv

Back edge: xz

[$z \to y \to x$ ~~cross~~ tree edge
and $x \to z$ back edge]

Cross edge: xw, vs, vw

~~Forward~~ Forward edge: sw, tv