1. Explain the steps involved in Dijkstra's Algorithm. Illustrate with an example.

⇒ Dijkstra's Algorithm is a greedy algorithm used to find the shortest path from a single source vertex to all other vertices in a weighted graph (with non-negative edge weights). It was devoloped by Dutch computer scientist Edsger W. Dijkstra.

- Purpose and use cases:
  - Google Maps & Gps System to find the shortest path.
  - Routing protocols in computer networks
  - AI and robotics for pathfinding platform.

Algorithm steps:

1. Initialize:
   - set the distance of the source vertex to 0 and all others to ∞
   - Mark all vertices as unvisited.
   - ~~ut~~

2. Process the current vertex:

- Pick the unvisited vertex with the smallest distance.
- Update the distances of its adjacent vertex if a shorter path is found.

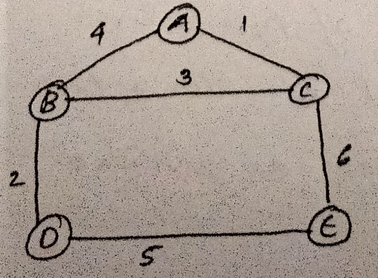
3. Update and Repeat.

- Mark the current as visited.
- Reapeted the process until all vertices are visited or the shortest path is determined.

4. Extract the shortest paths.

- Once all vertices are processed the shortest distances to all reachable nodes are known.
- optionally, backtrack to reconstruct the shortest path.

Example



steps initialization.

$A = 0, B = \infty, C = \infty$

$D = \infty, E = \infty$

2) start with A(0)

neibour : B(4), C(1)

Q. How many diffrent trees can be drawn with 4 vertices and 5 vertices?

- The no. of diffrent trees that can be drawn with n vertices is given by caylay's formula.

$$T(n) = n^{(n-2)}$$

where $T(n)$ is the number of labeled ~~user~~ trees often vertrices

for 4 vertrices $(n=4)$

$$T(4) = 4^{(4-2)} = 4^2 = 16.$$

So, there are 16 diffrent labelled trees with 4 vertices.

for 5 vertrices $(n=5)$

$$T(5) = 5^{(5-2)} = 5^3 = 125$$

so there are 125 diffrent labeled trees with 5 vertices.

If it is unlabelled tree

for 4 vertices → 3 diffred tres.

for 5 vertices → 4 diffrest trees

start with A(0):
update distance

$B = 4$, $C = 1$ $D = \infty$, $E = \infty$

3) Pick C(1):

Neibours : B(3), E(6)

$B = min(4, 1+3) = 4$

$E = min(\infty, 1+6) = 7$

4) pick B(4)

neibours : D(2)

$D = min(\infty, 4+2) = 6$

5) Pick D(6)

neibour : E(5)

$E = min(7, 6+5) = 7$

6) Pick E(7) final shortest Distance

A = 0 B = 4

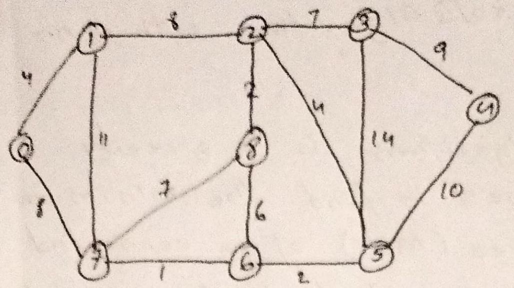
3. Explain kruskal's Algorithm with ~~exam~~ illustration.

Kruskal's Algorithm is a greedy Algorithm used to find the minimum spanning Tree (MST) of a connected, weighted graph. it finds the subset of edges that form a tree converting all vertices with minimum possible total edge weight.

Note: Always pick smallest edge. that doesn't form a cycle

Steps:

1. sort all edges in increasing order of their weights.
2. Initialize.
   - Each vertex is in its own set
   - The MST starts empty.
3. Iterate through edges
   - pick the smallest edge.
   - check if adding it forms a cycle using Union-find Algorithm.
   - If no cycle add it to the MST.
4. Reapeate until v-1 edges are added
   v. no. of vertices



step1: pick 7-6 . No weight 1

step 3: 8 – 2 picked

step 4: pick 6 – 5 – 2 weight

step 5: pick edge 0 – 1 – 4 weight

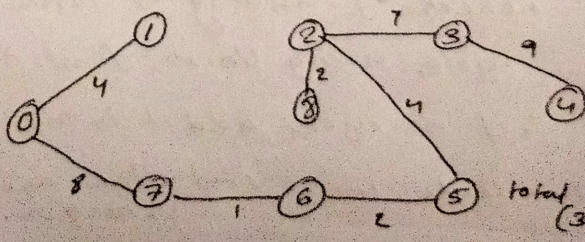
step 6: pick edge 2 – 5. 4 weight

step 7: pick edge 8 – 6 . since including this edge results in the cycle discard it.

~~step~~ : pick edge 2 – 3 – 7 weight

step 8 : pick edge 7 – 8 – Discard if cycle formed.
0 – 7 ~~no~~ w weight = 8.

step 9: discard 1 – 2 cycle formed
pick 3 – 4 . weight 9.



total cost
(37)

5 explain Prim's Algorithm with an illustrate.

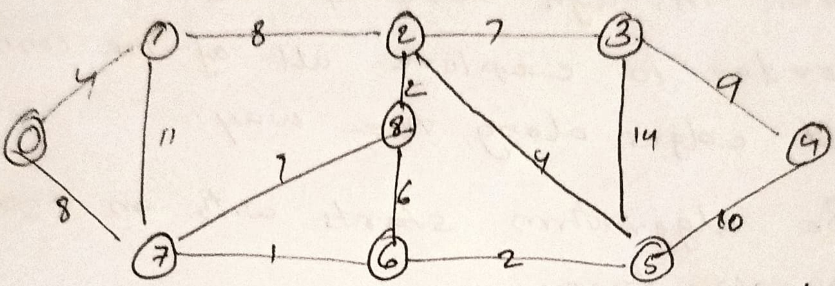
- Prim's algorithm is a Greedy algorithm like kruskals algorithm. The Algorithm always start with a single node and moves through several adjasent node in order to explore all of the connected edges along the way.

- The algorithm starts with an empty spanning tree.
- The idea is to maintain two sets of vertices. The first set contains the vertices already included in th MST and the other set contains the vertices not yet included.
- At every step, it considers all the edges that connet the two sets and picks the minimum weight edges. After picking the edge, it moves the other endpoint of the edges to the set containing MST.

A group of edges that connects two sets of vertices in a graph is called Articulation points (or cuts Vertices) in graph theory. so, at every step of Prim's Algorithm find a cut, pick the minimum weight edge from the cut, and include this vertex in MST set.

example

start with edges {0, 1} and {0, 7}; pick the minimum {0, 1} and add vertex 1



step 2 : choose between {0, 7} and {1, 2} pick {0, 7} or {1, 2} adding vertex 7

step 3 : select {7, 6} as it has the least weight (1), adding vertex 6.

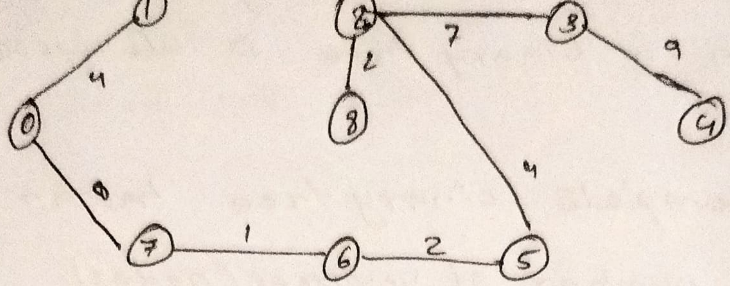
step 4 : Among {7, 8}, {6, 8} and {6, 5} pick {6, 5} (weight 2) and add vertex 5.

step 5 : Select {5, 2} (weight 4), adding vertex 2.

step 6 : choose {2, 8} weight 2), adding vertex 8.

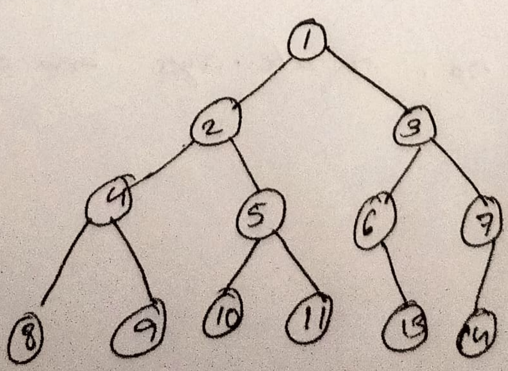
step 7 : pick {2, 3} as the smallest edge adding vertex 3

step 8 : finally select {3, 4}, adding vertex 4.



5. Define Binary Tree.

A Binary tree is a hierarchical data structure in which each node has at most 2 childre, refered to as the left child and the right child. It is commonly used in computer science for efficient storage and retrival of data, with various operations such as insertion, deletion and traversal.



6. Prove that the number of vertices n in a binary tree is always odd

A complete binary tree has an odd number of vertices (nodes)

At level 0, (at the root), we have 1 node.

At level 1, we has 2 nodes.

At level 2, we have $2 * 2 = 4$ nodes

At depth or height $n-1$, we have 2^{n-1} nodes.

Total $= 1 + 2 + 2^2 + 2^3 + \dots + 2^{n-1} = 2^n - 1$

Hence it is an odd number

At all other depths other than at level 0, we have even no. of nodes.

At level 0, we ~~t~~ have 1 node.

So total no. is always an odd number

7. Let p be the number of pendent vertices in a binary tree T. Prove that $p = \frac{n+1}{2}$

Fundamental property $p = I + 1$

The total no. of vertices in binary tree

$$n = I + p$$

Substituting $p = I + 1$

$$n = I + (I + 1) = 2I + 1$$

Solving for I:

$$2I = n - 1$$

$$I = \frac{n-1}{2}$$

using $p = I + 1$

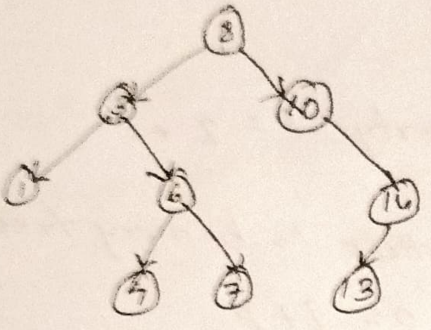
$$p = \frac{n-1}{2} + 1$$

$$p = \frac{n-1+2}{2}$$

$$p = \frac{n+1}{2}.$$

Tmz

5. Determine the preorder, in order and post order traversals of the orderd rooted tree below.



1. preorder Traversal (Root, left, Right)

root 8.

Preorder of left subtree 3, 1, 6, 4, 7

Preorder of Right Subtree : 10, 14, 13

final preorder 8, 3, 1, 6, 4, 7, 10, 14, 13

2. Inorder Travesal (Left, Root, Right)

inorder of left subtree 1, 3, 4, 6, 7

root 8.

final inorder : 1, 3, 4, 6, 7, 8, 10, 13, 14

3. postorder Traversal (left, Right, Root)

Postorder of left subtree : 14, 7, 6, 3

postorder of right subtree : 13, 14, 10

root 8.

final : 1, 4, 7, 6, 3, 13, 14, 10, 8.