MINIMUM SPANNING TREE

NIKITA UIKEY 181210033

Presentation Outline

MAIN TOPICS

- What is a Spanning Tree?
- What is a Minimum Spanning Tree?

1. Kruskal's Algorithm
2. Prim's Algorithm

What is a Spanning Tree?

WHAT IS A SPANNING TREE?

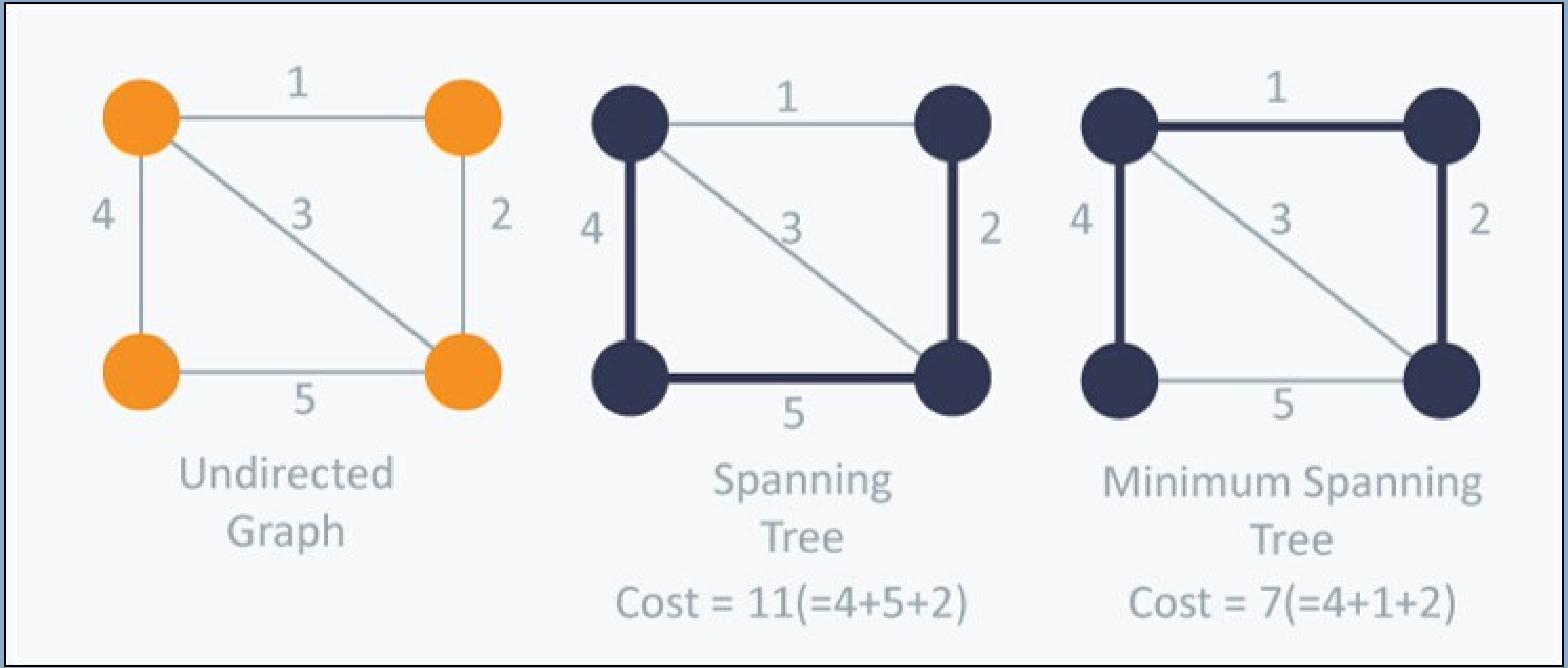
What is a Spanning Tree?

- Given an undirected and connected graph G=(V,E), a spanning tree of the graph G is a tree that spans G (that is, it includes every vertex of G) and is a subgraph of G (every edge in the tree belongs to G)
- The cost of a spanning tree is the sum of weights given to each edge of the spanning tree.

WHAT IS A MINIMUM SPANNING TREE?

- Minimum spanning tree is the spanning tree where the cost is minimum among all the spanning trees.
- A minimum spanning tree has (V – 1) edges where V is the number of vertices in the given graph.
- There can be many spanning trees, as well as many minimum spanning trees.

EXAMPLE


1
4
3
2
5
Undirected Graph
1
4
3
2
5
Spanning Tree
Cost = 11(=4+5+2)
1
4
3
2
5
Minimum Spanning Tree
Cost = 7(=4+1+2)

THERE ARE TWO FAMOUS ALGORITHMS FOR FINDING
THE MINIMUM SPANNING TREE:

Kruskal’s Algorithm

Prim’s Algorithm

Kruskal’s Algorithm

Kruskal’s Algorithm

- Kruskal’s Algorithm builds the spanning tree by adding edges one by one into a growing spanning tree.
- Kruskal's algorithm follows greedy approach as in each iteration it finds an edge which has least weight and add it to the growing spanning tree.
- It is a minimum-spanning-tree algorithm that finds an edge of the least possible weight that connects any two trees in the forest.

ALGORITHM

- Sort the graph edges with respect to their weights.
- Start adding edges to the MST from the edge with the smallest weight until the edge of the largest weight.
- Only add edges which doesn't form a cycle , edges which connect only disconnected components.

HOW KRUSKAL’S ALGORITHM WORKS?

HOW KRUSKAL’S ALGORITHM WORKS?

- It falls under a class of algorithms called greedy algorithms which find the local optimum in the hopes of finding a global optimum.
- We start from the edges with the lowest weight and keep adding edges until we reach our goal.

HOW TO CHECK
IF 2 VERTICES ARE
CONNECTED OR NOT ?

HOW TO CHECK IF 2 VERTICES ARE CONNECTED OR NOT ?

- This could be done using DFS which starts from the first vertex, then check if the second vertex is visited or not.
- But DFS will make time complexity large as it has an order of O(V+E) where V is the number of vertices, E is the number of edges.

SO THE BEST SOLUTION IS "DISJOINT SETS"

Disjoint sets are sets whose intersection is the empty set so it means that they don't have any element in common.

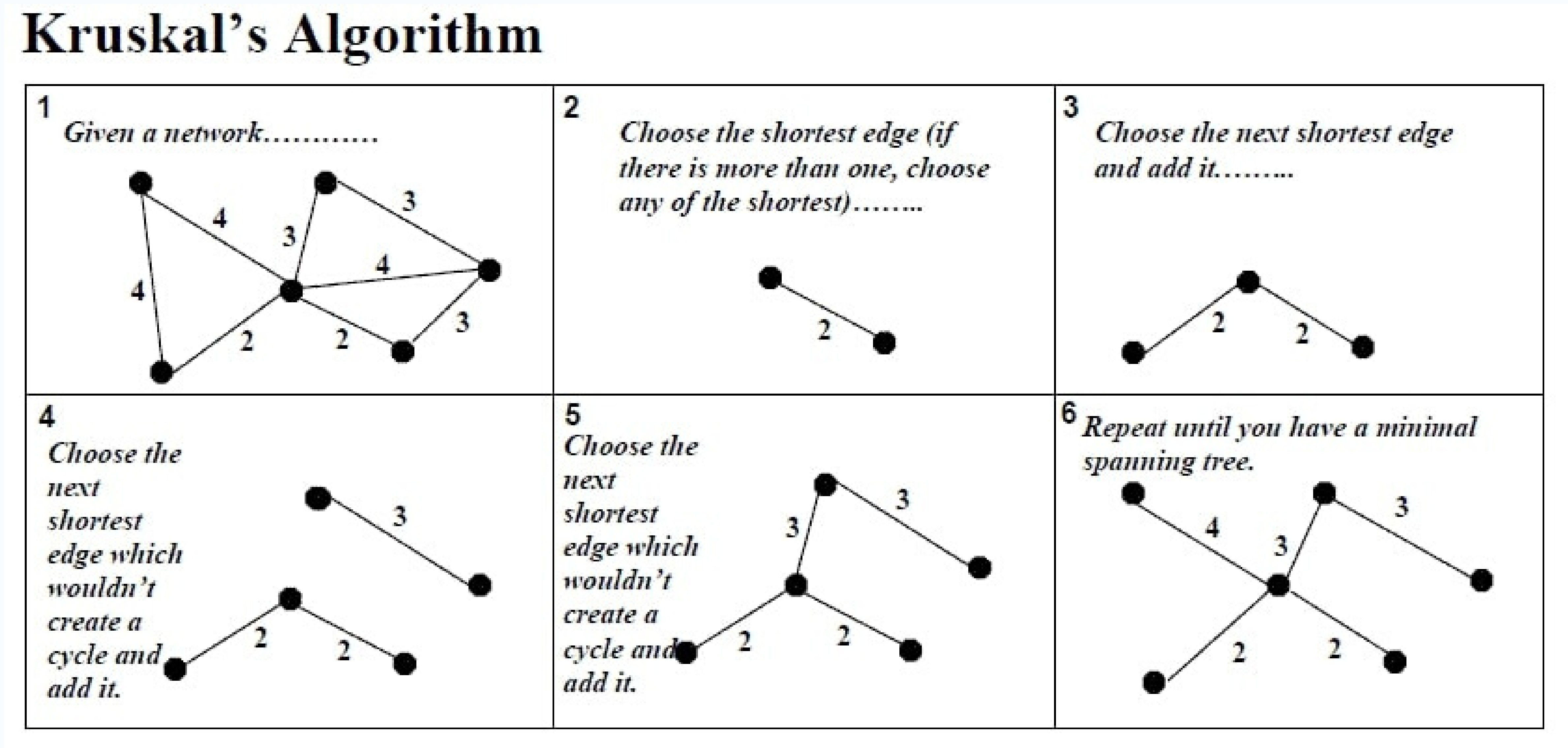
PSEUDOCODE

PSEUDOCODE

PSEUDOCODE

```
Kruskal(G) :
    A := $\emptyset$
    for each v $\in$ G.V do
       MAKE-SET(v)
    for each (u,v) in G.E ordered by weight(u, v), increasing do
        if FIND-SET(u) $\neq$ FIND-SET(v) then
           A := A $\cup$ {(u, v)}
                UNION(FIND-SET(u), FIND-SET(v))
    return A
```

EXAMPLE



Kruskal's Algorithm

1 *Given a network…………*	2 *Choose the shortest edge (if there is more than one, choose any of the shortest)……..*	3 *Choose the next shortest edge and add it……..*
4 *Choose the next shortest edge which wouldn't create a cycle and add it.*	5 *Choose the next shortest edge which wouldn't create a cycle and add it.*	6 *Repeat until you have a minimal spanning tree.*

Time Complexity

In Kruskal’s algorithm, most time consuming operation is sorting because the total complexity of the Disjoint-Set operations will be O(ElogV), which is the overall Time Complexity of the algorithm.

PRIM'S ALGORITHM
PRIM'S ALGORITHM

PRIM’S ALGORITHM

- Prim’s Algorithm also use Greedy approach to find the minimum spanning tree. In Prim’s Algorithm we grow the spanning tree from a starting position.
- Unlike an edge in Kruskal's, we add vertex to the growing spanning tree in Prim's.

ALGORITHM

- Maintain two disjoint sets of vertices. One containing vertices that are in the growing spanning tree and other that are not in the growing spanning tree.
- Select the cheapest vertex that is connected to the growing spanning tree and is not in the growing spanning tree and add it into the growing spanning tree. This can be done using Priority Queues. Insert the vertices, that are connected to growing spanning tree, into the Priority Queue.
- Check for cycles. To do that, mark the nodes which have been already selected and insert only those nodes in the Priority Queue that are not marked.

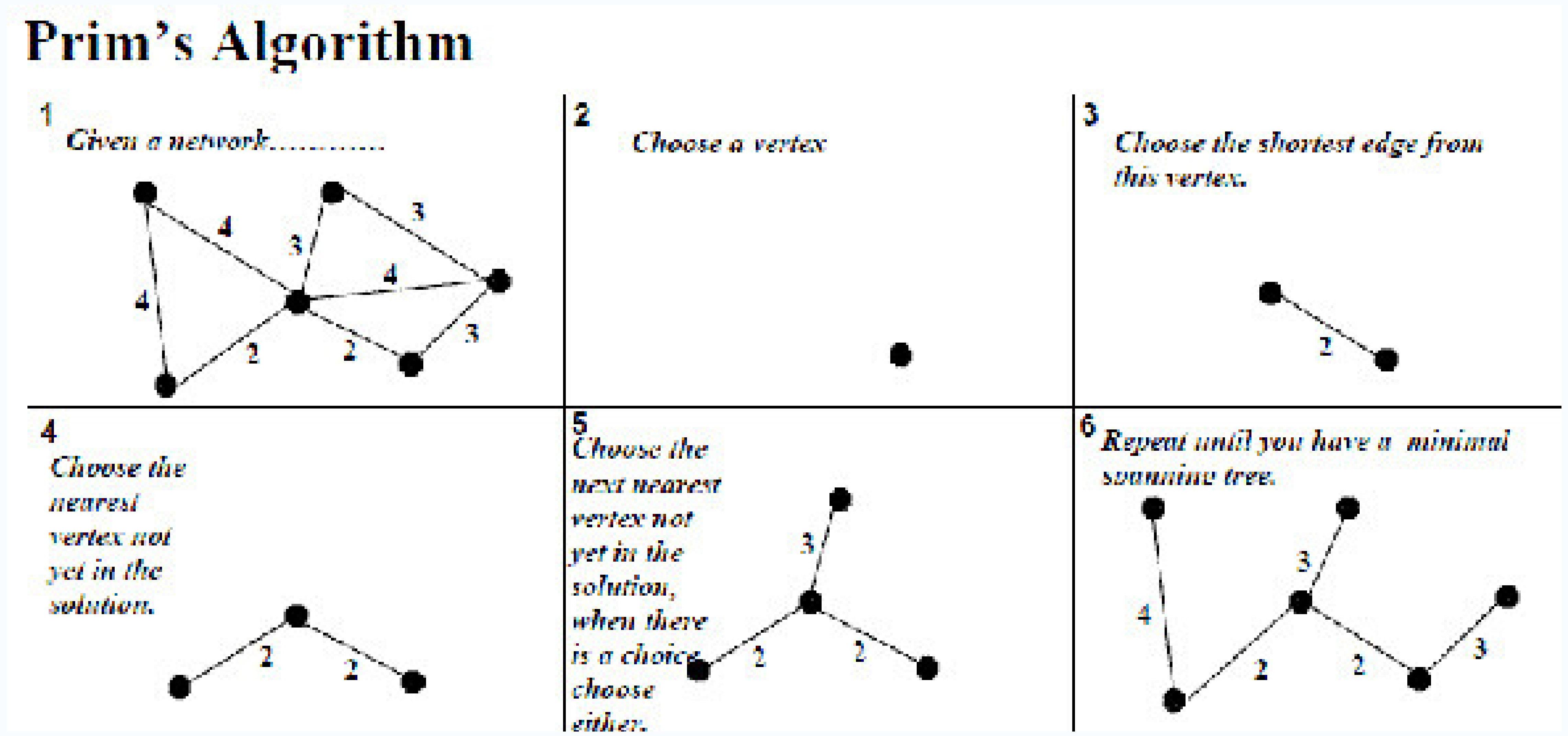
PSEUDOCODE

PSEUDOCODE

PSEUDOCODE

```
Prim's (G) :
T = $\varnothing$;
U = { 1 };
while ($U \neq V$)
    let (u, v) be the lowest cost edge such that $u \in U$ and $v \in V - U$;
    $T = T \cup \{(u, v)\}$
    $U = U \cup \{v\}$
```

EXAMPLE



Prim's Algorithm

1 Given a network............

2 Choose a vertex

3 Choose the shortest edge from this vertex.

4 Choose the nearest vertex not yet in the solution.

5 Choose the next nearest vertex not yet in the solution, when there is a choice choose either.

6 Repeat until you have a minimal spanning tree.

Time Complexity

Itis O((V+E) logV) because
each vertex is inserted in the priority queue only once and insertion in priority queue take logarithmic time