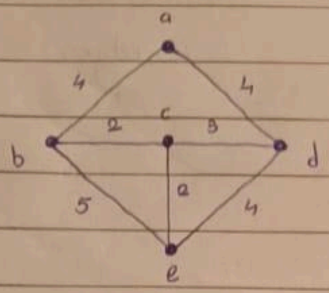
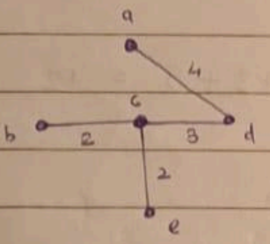
Prims Algorithm

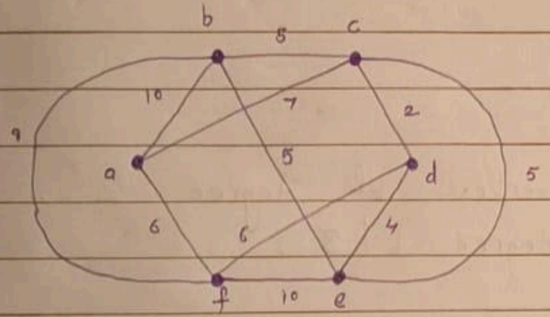
Q Find MST using Prim's





2+2+3+4 = 11

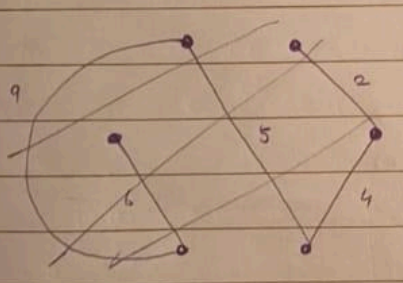
Q

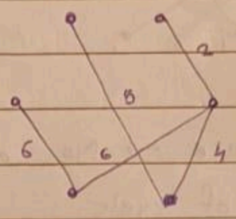


6 Vertices

11 Edges

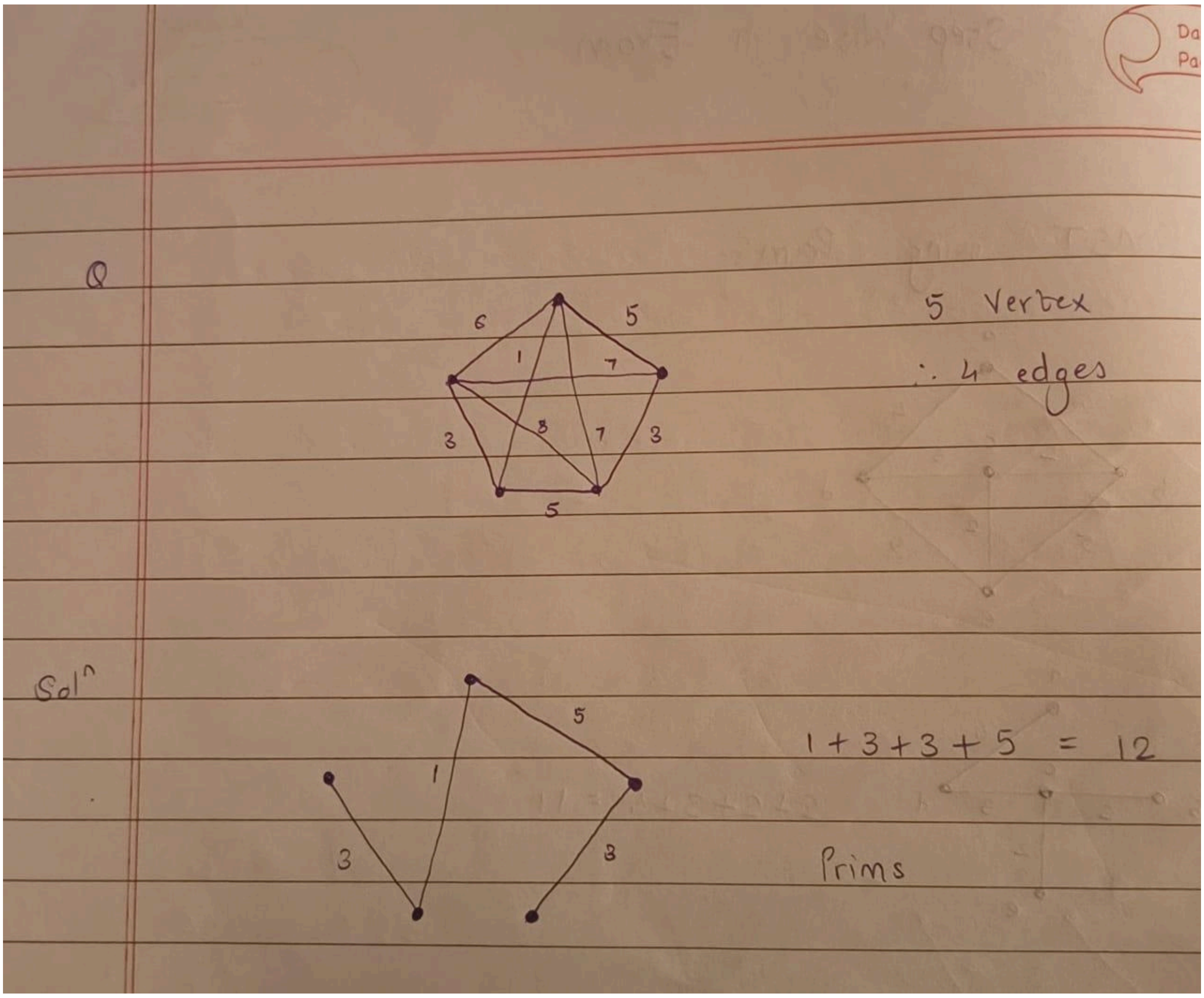
∴ 5 Edges





2+4+5+6+6 = 23

Check for smallest edge on each vertex on each step.



6
5
1
7
3
8
7
3
5
5
1
3
3

Q

5 Vertex

∴ 4 edges

Sol^n

1 + 3 + 3 + 5 = 12

Prims

Job Sequencing with Deadlines

This is a greedy algorithm that schedules jobs with deadlines and profits to maximize total earnings.

Problem Statement

- You have n jobs, each with:
    - A **deadline** (by when it must be completed).
    - A **profit** (earned only if finished before the deadline).
- You can complete only **one job per time unit**.
- The goal is to **maximize total profit** by selecting jobs optimally.

Algorithm Steps

1. **Sort** all jobs **by profit in descending order**.
2. For each job, try to schedule it at the **latest available slot before its deadline**.
3. If no slots are available, discard the job.
4. Repeat until all jobs are scheduled or time slots are full.

Example

Job	Deadline	Profit
J1	2	100
J2	1	50
J3	2	200
J4	1	20

Step-by-step execution:

1. **Sort by profit:** J3 (200), J1 (100), J2 (50), J4 (20).
2. Schedule **J3** at time **2**.
3. Schedule **J1** at time **1**.
4. **J2 & J4** cannot be scheduled (no available slots).

Final Schedule:

- **J1 at time 1**
- **J3 at time 2**
- **Total Profit = 300**

Time Complexity: O(n log n) (sorting) + O(n) (scheduling) = O(n log n)

Residual Network, Augmenting Path & Ford-Fulkerson Method for Max Flow

1. Residual Network

A **Residual Network** is used in flow problems where we track:

- **Remaining capacity** on each edge after sending flow.
- **Reverse edges** to allow "pushing back" flow.

Example: If an edge (A → B) has a capacity of **10** and we push a flow of **7**, the residual capacity is **3**.

2. Augmenting Path

An **augmenting path** is a path from the **source** to the **sink** in the residual network **where additional flow can be pushed**.

- If there exists an augmenting path, the network **is not yet at max flow**.
- We keep finding augmenting paths and **increasing flow** until no more augmenting paths exist.

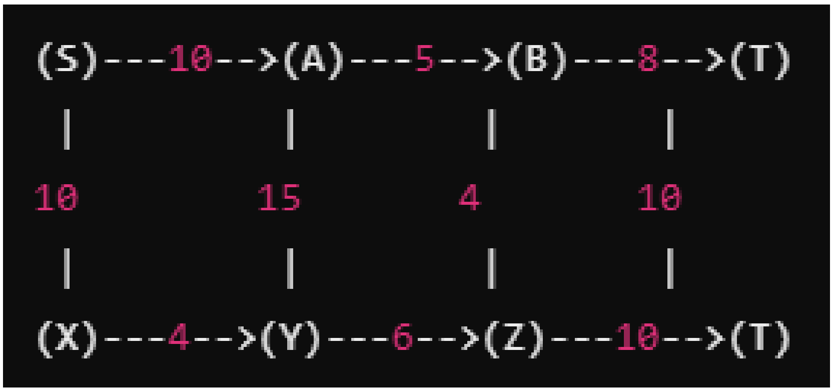
3. Ford-Fulkerson Method (For Maximum Flow)

This method calculates the **maximum flow** in a network.

Algorithm Steps:

1. Start with **zero flow** in all edges.
2. Find an **augmenting path** in the residual network using **DFS/BFS**.
3. Find the **minimum residual capacity** (bottleneck) along this path.
4. **Update flow values** along the path.
5. Repeat until no more augmenting paths exist.

Example:



```
(S)---10-->(A)---5-->(B)---8-->(T)
 |          |         |       |
10         15         4      10
 |          |         |       |
(X)---4-->(Y)---6-->(Z)---10-->(T)
```

1. Find an augmenting path, e.g., S → A → B → T.
2. Find the minimum capacity (bottleneck) on this path.
3. Update residual capacities.
4. Repeat until no more augmenting paths exist.

Time Complexity: $O(VE^2)$ (in worst case with DFS), **$O(V^3)$** (with BFS & Edmonds-Karp).