Table 1: Data showing the cost of workers doing different jobs

	Jobs			
Workers	Cleaning	Sweeping	Receptionist	Cooking
Agness	ZAR9000	ZAR7500	ZAR7500	ZAR800
Rose	ZAR3500	ZAR8500	ZAR5500	ZAR6500
Joyce	ZAR12500	ZAR9500	ZAR9000	ZAR10500
Zainab	ZAR4500	ZAR11000	ZAR9500	ZAR11000

$$\text{max} \qquad \sum_{e} c_e x_e \tag{18}$$

$$\text{subject to} \qquad \sum_{\text{e incident to } i} x_e = 1 \qquad (\text{for each } i \in V) \tag{19}$$

$$x_e \in \{0, 1\} \tag{20}$$

where c_e is the weight or utility of the edge e. For the case when the matching is not perfect the constraint (19) is replaced with

$$\sum_{\text{e incident to i}} x_e \leq 1, \, \forall i \in V,$$

then this problem is known as bipartite matching problem.

Consider that a matching gives an assignment of people to tasks. You want to get as many tasks done as possible, i.e. you want a maximum matching, one that contains as many edges as possible. Create an instance of bipartite matching. Then create an instance of a network flow where the solution to the network flow problem can easily be used to find the solution to the bipartite matching. The edges used in the maximum network flow will correspond to the largest possible matching!

1.5 Minimum Spanning Tree Problem

A spanning tree is a tree (i.e., a connected acyclic graph) that spans (touches) all the nodes of an undirected network[4]. The cost of a spanning tree is the sum of the costs (or lengths) of its edges. In the minimum spanning tree problem, we wish to identify a spanning tree of minimum cost (or length). A spanning tree of a graph has the following features:

- Every node in the network is connected to every other node by a sequence of arcs from the subnetwork, where the direction of the arcs is ignored.
- The subnetwork contains no loops, where a loop is a sequence of arcs connecting a node to itself, where again the direction of the arcs is ignored.

A subnetwork that satisfies the above two properties is called a spanning tree.

[4]Minimum spanning tree for directed graph can also be constructed but it has limited applications.

The minimum spanning tree (MST) problem is one of the simplest and most fundamental problems in network optimization where given an undirected $G = (V, E)$ and the edge cost $c : E \rightarrow R$ the goal is to find the spanning tree of minimum total cost. If all edges have non-negative costs (i.e $c : E \rightarrow R^+$) , then the MST problem is equivalent to the connected subgraph problem which asks for the computation of a minimum cost subgraph H of G that connects all nodes of G. The MST of the graph in Fig. 15 is shown in bold.

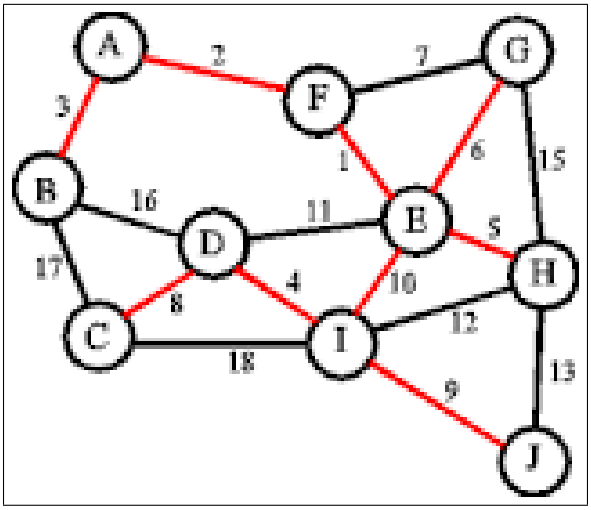

Figure 15: Minimum spanning tree of network in red

As an application, consider the communication company Telkom needs to build a communication network that connects n different users. The cost of making a link joining i and j is c_{ij}. What is the minimum cost of connecting all of the users?

Find the MST of the graph in Fig. 16, where $c_{12} = 1, c_{15} = 1.5, c_{26} = 4, c_{34} = 1.5, c_{36} = 0.5, c_{45} = 1, c_{47} = 2, c_{89} = 2, c_{2,10} = 6, c_{7,10} = 1, c_{67} = 5, c_{68} = 1$.

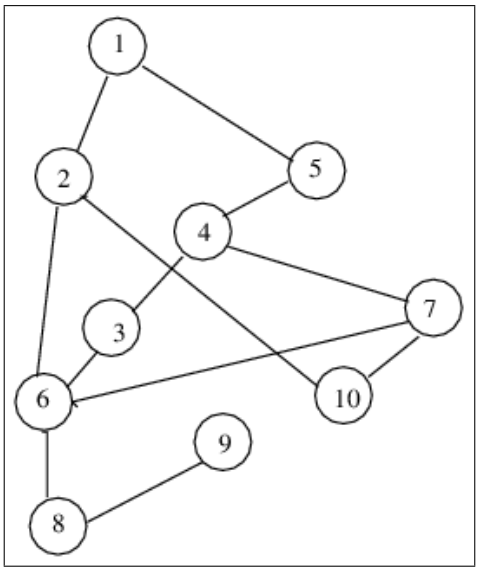

Figure 16: Find the MST of the given network

The mathematical formulation of the MST is based on the fact that the tree must be connected. How can you model this requirement? Let $x_e \in E$ be such that

$$x_e = \begin{cases} 1 & \text{if item } e \text{ included in the spanning tree,} \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

Let S be any set of vertices. Then S and $V \setminus S$ should be connected. In addition the MST must have $(n-1)$ edges in total. We will now address both of these two requirements in the mathematical formulation. Let

$$\hat{\delta}(S) = \left\{ e = (i,j) \middle| i \in S, j \in V \setminus S \right\}.$$

$$\text{min} \quad \sum_{e \in E} c_e x_e \tag{22}$$

$$\text{subject to} \quad \sum_{e \in \hat{\delta}(S)} x_e \geq 1 \qquad \forall S \subset V, S \neq \emptyset \tag{23}$$

$$\sum_{e \in E} x_e = n - 1 \tag{24}$$

$$x_e \in \{0, 1\}. \tag{25}$$

Clearly, the constraint (23) ensures that the spanning tree is connected, while the number of edges requirement is fulfilled by the constraint (24).

1.6 Solving Max-Flow Problem: The Ford Fulkerson Algorithm

The Ford-Fulkerson algorithm (FFA) [3] is a greedy algorithm that computes the maximum flow in a flow network. The algorithm does not need to maintain the amount of flow on each edge but work with capacity values directly. The algorithm is based on three bright ideas:

- residual flow networks: the graph that shows where extra capacity might be found
- augmenting paths: paths along which extra capacity is possible
- cuts: used to characterize the flow upper bounds in a network

The basic method is iterative, starting from a network with zero flow. Starting with the zero flow, repeatedly augment the flow along any path from s to t in the residual graph, until there is no such path.

The idea behind the algorithm is as follows: as long as there is a path P from the source (start node s) to the sink (end node t), with available capacity on all edges in the path, we send a flow, say x, along the path[5]. We then find another path, add the feasible flow used on each path, and so on. The amount of flow x (from s to t) which satisfies

[5] A path with available capacity is called an augmenting path.