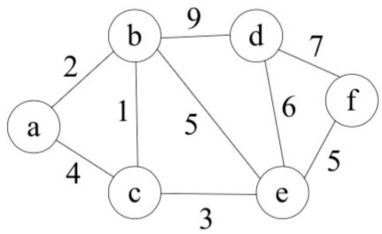
Assignment 7

VE281

Bingcheng HU

516021910219

Q1



Adjacency matrix repre- sentation of the graph.

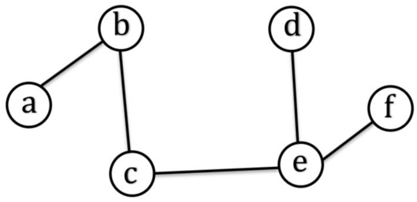
	a	*b*	*c*	*d*	*e*	*f*
a	∞	2	4	∞	∞	∞
b	2	∞	1	9	5	∞
c	4	1	∞	∞	3	∞
d	∞	9	∞	∞	6	7
e	∞	5	3	6	∞	5
f	∞	∞	∞	7	5	∞

Q2

Apply Prim's algorithm to the graph, the intermediate steps of applying the algorithm is:

1. a->b
2. b->c
3. c->e
4. e->f
5. e->d

The minimum spanning tree is



Q3

I'll use **Dijkstra's Algorithm** with some changes. Keep distance estimate $D(v)$ and predecessor $P(v)$ for each node v.

The time complexity is $O(V+E)$.

Algorithm:

1. Initially, $D(s)=0$; $D(v)$ for other nodes is $+\infty$; $P(v)$ is unknown.
2. Store all the nodes in a set R.
3. While R is not empty
    1. Choose node v in R such that $D(v)$ is the smallest. Remove v from the set R.
    2. Declare that v's shortest distance is known, which is $D(v)$.
    3. For each of v's neighbors u that is still in R, update distance estimate $D(u)$ and predecessor $P(u)$. For each of v's neighbors u that is still in R, if $D(v)+w(v,u) > D(u)$, then update $D(u)=D(v)+w(v,u)$ and the predecessor $P(u)=v$.
4. By backtracking, if we cannot find the s point, there is no path exists between the two nodes. Otherwise, we can obtain the longest path by backtracking.

Q4

Again, similar to Q3, with change $D(v)+w(v,u)>D(u)$ to $D(v)\times w(v,u)<D(u)$.

Algorithm:

1. Initially, $D(s)=0$; $D(v)$ for other nodes is $+\infty$; $P(v)$ is unknown.
2. Store all the nodes in a set R.
3. While R is not empty
    1. Choose node v in R such that $D(v)$ is the smallest. Remove v from the set R.
    2. Declare that v's shortest distance is known, which is $D(v)$.
    3. For each of v's neighbors u that is still in R, update distance estimate $D(u)$ and predecessor $P(u)$. For each of v's neighbors u that is still in R, if $D(v)\times w(v,u) < D(u)$, then update $D(u)=D(v)\times w(v,u)$ and the predecessor $P(u)=v$.
4. By backtracking, if we cannot find the s point, there is no path exists between the two nodes. Otherwise, we can obtain the longest path by backtracking.

Q5

```
DFS(v) {
  visit v;
  v.edges --;
  for(each node u adjacent to v)
    if(u.edges != 0) DFS(u);
}


path(s){
    for(each node v in the graph){
        set v.edges = # of edges adjacent to v;
    }
    DFS(s);
}
```

This is an O(|V |+|E|)-time algorithm.

For example: path(A) for the following graph, we can get `A→B→C→D→C→A→C→B→A`