Chapter 12 Algorithms of Minimum Spanning Trees

Yonghui Wu

School of Computer Science, Fudan University

Wechat： 13817360465

Email: yhwu@fudan.edu.cn

- A tree is a connected undirected simple graph with no circuit. A tree with *n* vertices has *n*-1 edges.
- If an edge of a tree is deleted, the tree becomes an unconnected graph.
- If an edge is added between any two vertices in a tree, a circuit is produced.

- Finding a spanning tree so that the sum of weights of edges of the tree is minimized in a weighted undirected connected graph. Such a spanning tree is called a minimum spanning tree.

- Greedy strategy is used to find a minimum spanning tree.
- At each step, the added edge can’t form a circuit; and based on it, the weight of the added edge should be minimal. Such an added edge is called a safe edge.
- Two algorithms of minimum spanning trees:
  - Kruskal Algorithm
  - Prim Algorithm

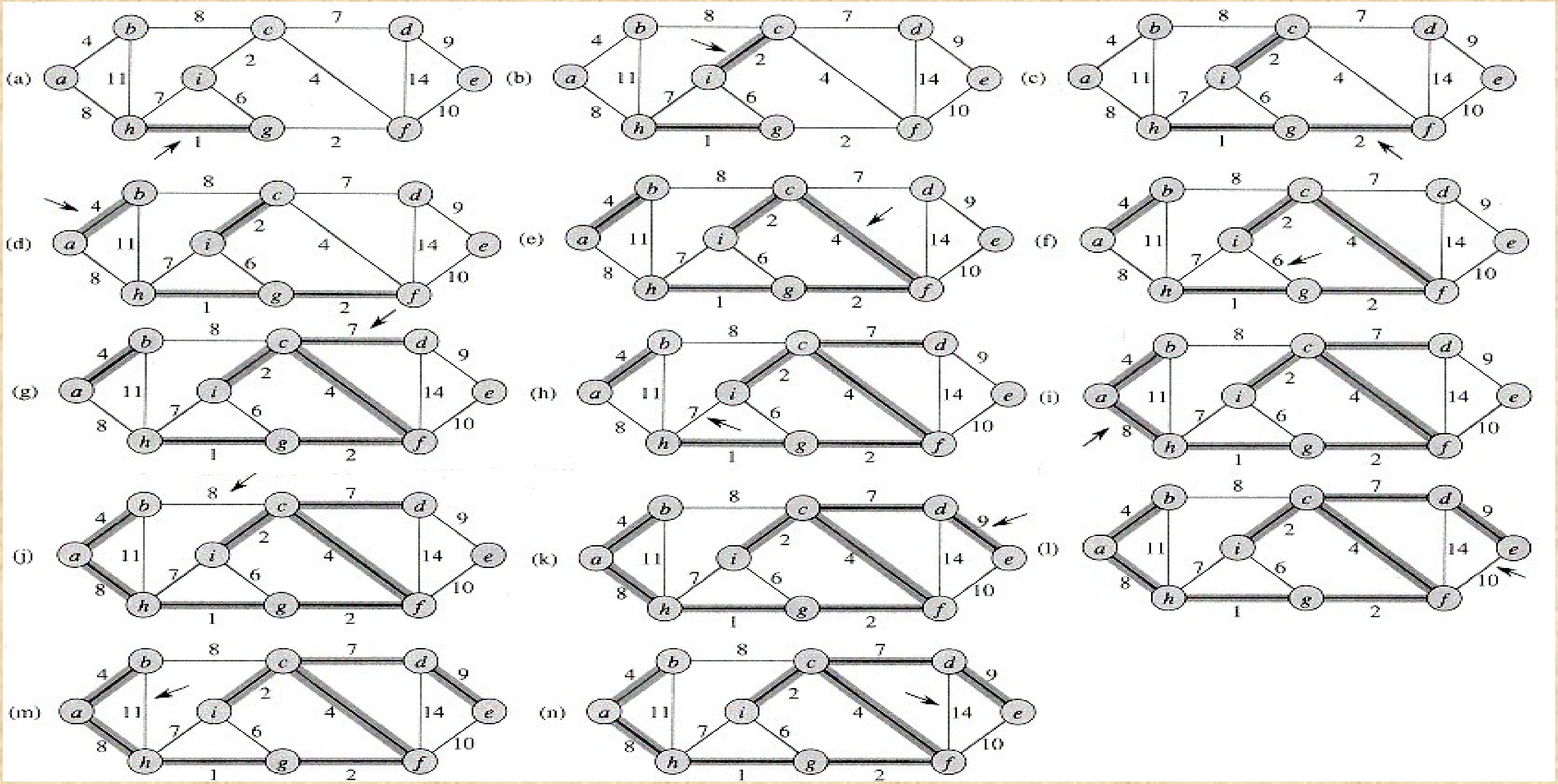
Chapter 12 Algorithms of Minimum Spanning Trees

- 12.1 Kruskal Algorithm
- 12.2 Prim Algorithm

12.1 Kruskal Algorithm

- Initially, n vertices constitute a forest.
- Then, edge (u, v) connecting two distinct trees in the forest with the least-weight is regarded as the safe edge and is added into the forest.
- Repeat the process until the minimum spanning tree is gotten.

An example for Kruskal Algorithm



Kruskal Algorithm

- Given a weighted connected graph $G(V, E)$, where $|V| = n$, and $|E| = m$.
- Sort edges in ascending order of weight values;
- Suppose initially F is a forest consisting of n trees, and each tree corresponds to a vertex in graph G;

Kruskal Algorithm

```
Initially the sum of weights of edges of the minimum spanning tree
$ans$=0;
for ( int $k$=1; $k \le m$; $k$++ )  // Enumerate $m$ edges in Graph $G$ in the order
{    if ( the current edge ($i$, $j$) connects two distinct components)
       {  add edge ($i$, $j$) into the forest and combine the two components;
          $ans$+=the weight of edge ($i$, $j$);
       }
}
Output $ans$;
```

- If a subtree is regarded as a set, and the root of the subtree is regarded as the representative of the set, then union-find set can be used to determine whether two vertices belong to one tree and combine two different trees.
- The time complexity for Kruskal Algorithm is O(E*ln$E$). That is to say, its efficiency depends on the number of edges $|E|$. Therefore Kruskal Algorithm is suitable for sparse graphs.

12.1.1 Constructing Roads

- Source: PKU Monthly, kicc
- IDs for Online Judge: POJ 2421

- There are *N* villages, which are numbered from 1 to *N*, and you should build some roads such that every two villages can connect to each other. We say two villages *A* and *B* are connected, if and only if there is a road between *A* and *B*, or there exists a village *C* such that there is a road between *A* and *C*, and *C* and *B* are connected.
- We know that there are already some roads between some villages and your job is the build some roads such that all the villages are connect and the length of all the roads built is minimum.

Input

The first line is an integer N, ($3 \le N \le 100$), which is the number of villages. Then come N lines, the i-th of which contains N integers, and the j-th of these N integers is the distance (the distance should be an integer within [1, 1000]) between village i and village j.

Then there is an integer Q ($0 \le Q \le N * \frac{N+1}{2}$). Then come Q lines, each line contains two integers a and b ($1 \le a < b \le N$), which means the road between village a and village b has been built.

Output

You should output a line contains an integer, which is the length of all the roads to be built such that all the villages are connected, and this value is minimum.

Analysis

- Villages and roads connecting villages are represented as a weighted graph, where
  - villages are represented as vertices,
  - roads are represented as edges,
  - the length of a road is the weight of an edge.

- The problem requires to add edges to spanning trees (already built roads between some villages) to construct a minimum spanning tree.

- Because the number of added edges is less than *N*-1, Kruskal Algorithm is suitable for the problem.
- The problem requires to calculate the length of all the roads to be built.

- An adjacency matrix *P* is used to represent the weighted graph, an array *Fa* is used to store every vertex's parent pointer pointing to its parent.
- By using parent pointers, the root of the tree containing the vertex can be found: from vertex *i*, repeat using array *Fa* ($Fa[Fa[\ldots Fa[i]\ldots]]$) until $x==Fa[x]$. *x* is the root of the tree containing vertex *i*, that is, $Fa[i]=x$ ($0 \le i \le n-1$).

- ①Initialization;
- Input a test case and construct an adjacency matrix P to represent the weighted graph;
- N vertices are represented as N distinct spanning trees ($Fa[i]=i$, $0\leq i\leq n-1$);
- Input Q built roads (a, b), Suppose $Fa[b]=a$ ($1\leq a<b\leq N$);
- Initialize the minimal length of all the roads to be built ans 0;

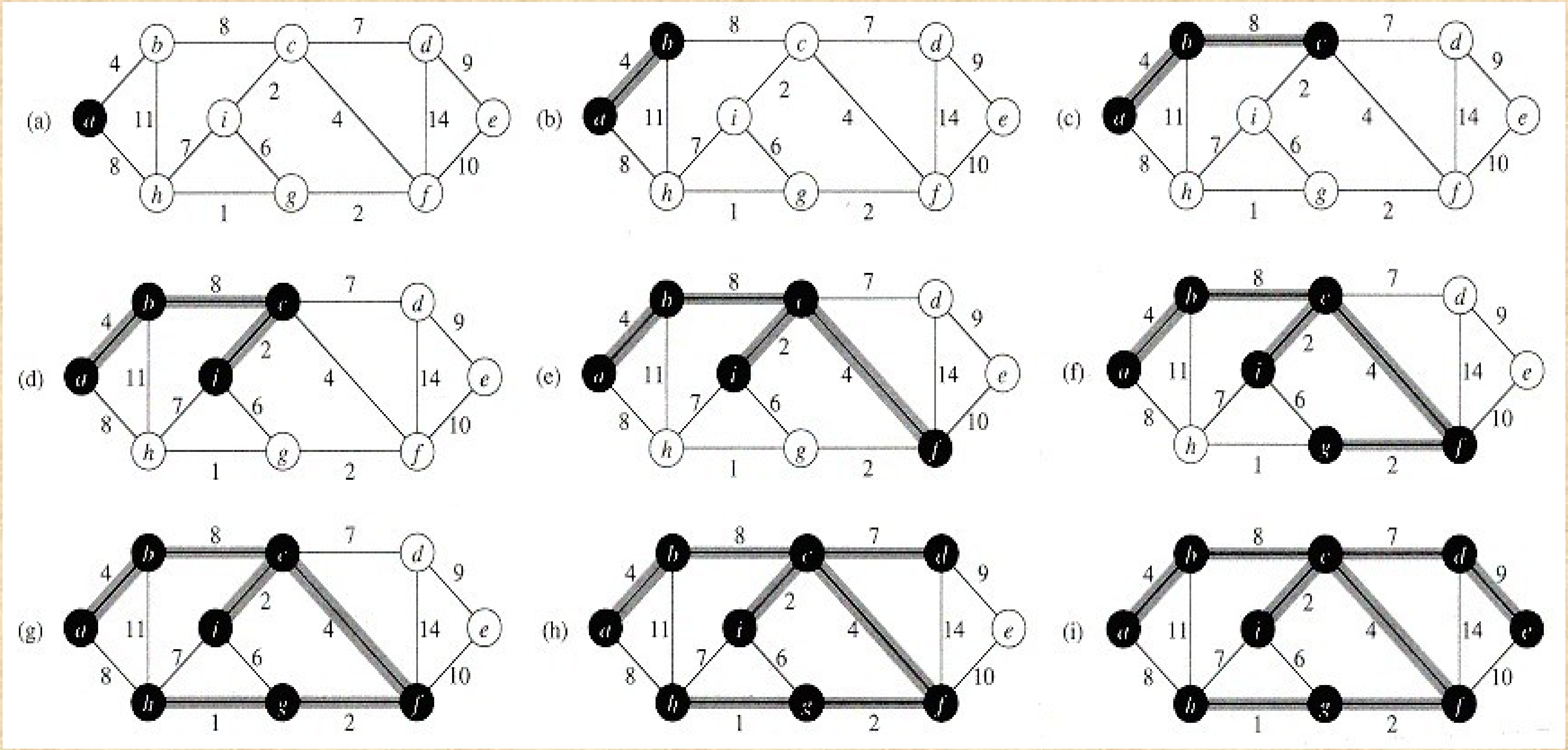
- ②Compute the minimal length of all the roads to be built *ans*;
- Sort weights of edges k in ascending order ($1 \le k \le 1000$) and enumerate these edges:
- If the current enumerated edge (i, j) whose weight is k connects two distinct subtrees ($P[i][j]==k$ && the root of the subtree containing vertex i != the root of the subtree containing vertex j, $0 \le i < j \le n-1$), then combine the tree containing vertex i into the tree containing vertex j ($Fa[Fa[i]]=Fa[j]$); and k is added into the length *ans* (*ans* += k);

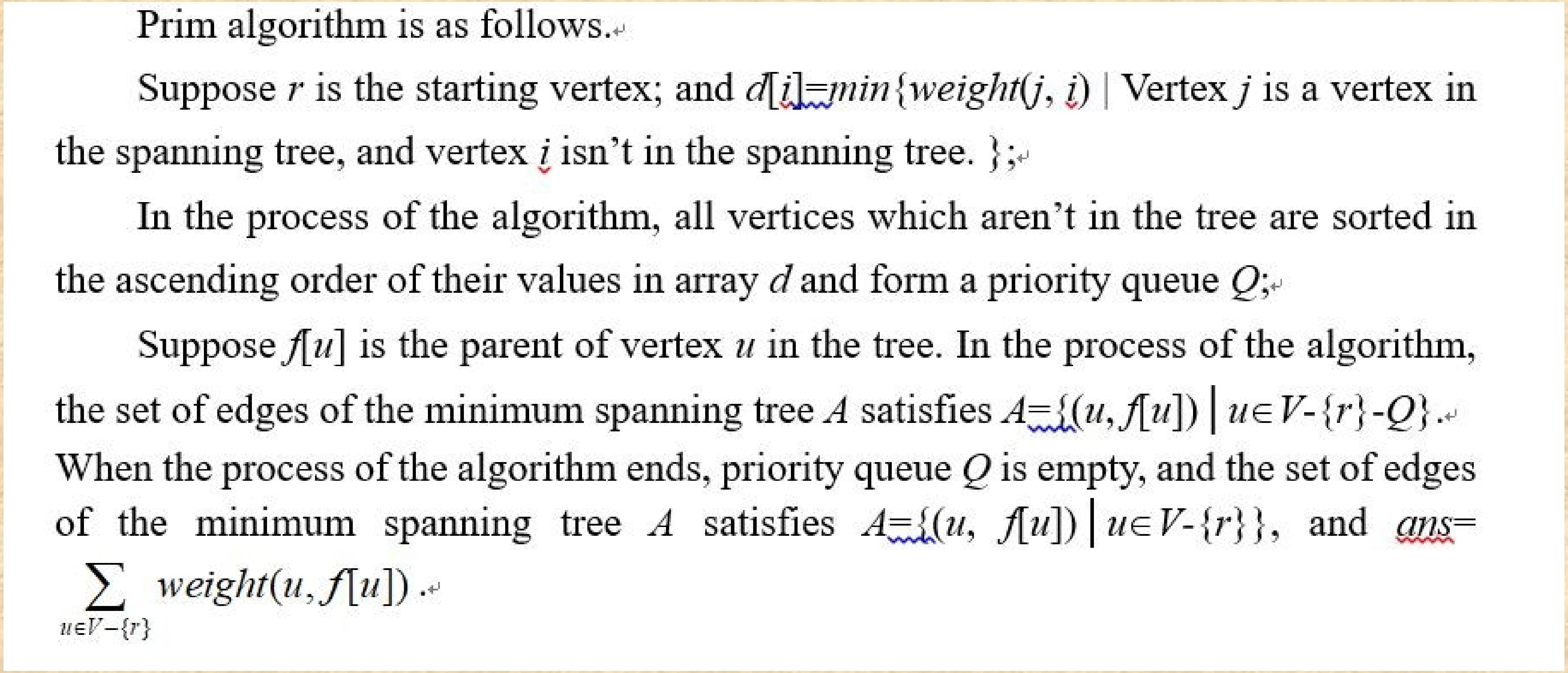
- ③Output the length of all the roads to be built *ans*;

12.2 Prim Algorithm

- In Prim Algorithm, edges in set A make up a single minimum spanning tree.
- Initially A is empty.
- Repeatedly add edges to A so that at each step an added edge has only one vertex in the tree and contributes the minimum amount possible to the tree’s weight.

An example for Prim Algorithm





Prim algorithm is as follows.

Suppose r is the starting vertex; and $d[i]=min\{weight(j, i)$ | Vertex j is a vertex in the spanning tree, and vertex i isn't in the spanning tree. $\}$;

In the process of the algorithm, all vertices which aren't in the tree are sorted in the ascending order of their values in array d and form a priority queue Q;

Suppose $f[u]$ is the parent of vertex u in the tree. In the process of the algorithm, the set of edges of the minimum spanning tree A satisfies $A=\{(u, f[u]) \mid u\in V\text{-}\{r\}\text{-}Q\}$. When the process of the algorithm ends, priority queue Q is empty, and the set of edges of the minimum spanning tree A satisfies $A=\{(u, f[u]) \mid u\in V\text{-}\{r\}\}$, and $ans=\sum_{u\in V-\{r\}} weight(u, f[u])$.

Prim algorithm

```
• for (each $v \in G(V)$)
• { $d[v]=\infty$; $f[u]$=nil };
•   $d[r]=0$; $Q=G(V)$;
•    while ($Q!=\varnothing$)
•   {   Get vertex $u$ whose $d[u]$ is the least in $Q$;  // Add vertex $u$ into minimum spanning tree
•       $Q=Q-\{u\}$;
•        if ($u!=r$) $ans=ans+w[u,f[u]]$;   //If vertex $u$ isn't the root, then add the weight into $ans$
•        for (each $v \in$ the set of vertices adjacent to vertex $u$)        // renew the value of $d$ and parent pointer of vertex $v$ which is adjacent to vertex $u$
•        if ($v \in Q$)&&($w[u,v]<d[v]$)
•              { $f[v]=u$; $d[v]=w[u,v]$; }
•   };
• Output the weight of the minimum spanning tree $ans$;
```

12.2.1 Agri-Net

- **Source: USACO**
- **IDs for Online Judge: POJ 1258**

- Farmer John has been elected mayor of his town! One of his campaign promises was to bring internet connectivity to all farms in the area. He needs your help, of course.
- Farmer John ordered a high speed connection for his farm and is going to share his connectivity with the other farmers. To minimize cost, he wants to lay the minimum amount of optical fiber to connect his farm to all the other farms.
- Given a list of how much fiber it takes to connect each pair of farms, you must find the minimum amount of fiber needed to connect them all together. Each farm must connect to some other farm such that a packet can flow from any one farm to any other farm.
- The distance between any two farms will not exceed 100,000.

- **Input**
- The input includes several cases. For each case, the first line contains the number of farms, *N* ($3 \leq N \leq 100$). The following lines contain the $N \times N$ conectivity matrix, where each element shows the distance from one farm to another. Logically, they are *N* lines of *N* space-separated integers. Physically, they are limited in length to 80 characters, so some lines continue onto others. Of course, the diagonal will be 0, since the distance from farm i to itself is not interesting for this problem.
- **Output**
- For each case, output a single integer length that is the sum of the minimum length of fiber required to connect the entire set of farms.

Analysis

- Farms and fibers connecting farms are represented as a weighted graph, where
    - farms are represented as vertices, John’s farm is represented as vertex 0;
    - straight lines connecting farms are represented as edges;
    - distances between two farms are weights of corresponding edges.

- Finding the minimum amount of fiber needed to connect them all together is to calculate the minimum spanning tree of the graph.
- Because the upper limit of the number of vertices is 100, Prim Algorithm is suitable for the problem.

- Suppose
- *v* is the adjacency matrix for the graph.
- Array *dist* is used to store a priority queue *Q*, where *dist*[*i*] is the distance between vertex *i* and the spanning tree.
  - Initially $dist[0]=\infty$, $dist[i]=v[0][i]$($1\le i\le n$-1).
- Array *use* is the flag that a vertex is in the spanning tree or not.
  - Initially, only John's farm is in the spanning tree (*use*[0]=true), and other vertices aren't in the spanning tree (*use*[*i*]=false, $1\le i\le n$-1).

```
$n$-1 edges are added into the spanning tree as follows.
{ find such a vertex $tmp$ connecting the spanning tree that $dist[tmp] = \min_{1 \le i \le n-1, usd[i]=false} \{dist[i]\}$ );
    The weight of the edge connecting vertex $tmp$ and the spanning tree is accumulated ($tot$ +=$dist[tmp]$);
    Vertex $tmp$ is added into the spanning tree ( $use[tmp]$=true );
    Adjust array $dist$ ($dist[k]$=min{$dist[k]$, $v[k][tmp]$ | $use[k]=false$ },$1 \le k \le n$-1);
};
Output the weight of the minimum spanning tree $tot$;
```