Uva 1267

Network

Seoul 2007, LA 3902

Time: 3 seconds

Problem Description (1/3)

- Consider a tree network with n nodes
  - internal nodes correspond to servers
  - terminal nodes correspond to clients
- The nodes are numbered from 1 to n. Among the servers, there is an original server S which provides VOD (Video On Demand) service. To ensure the quality of service for the clients, the distance from each client to the VOD server S should not exceed a certain value k.

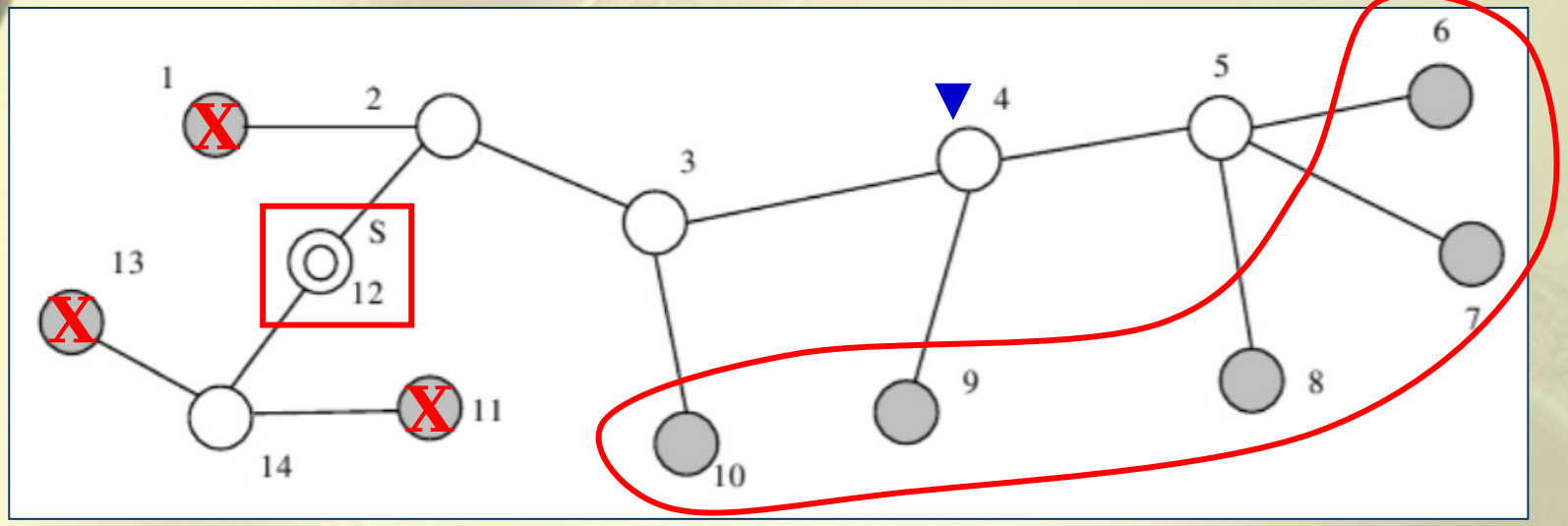
Problem Description (2/3)

- The distance from a node u to a node v in the tree is defined to be the number of edges on the path from u to v.
- If there is a nonempty subset C of clients such that the distance from each u in C to S is greater than k, then replicas of the VOD system have to be placed in some servers so that the distance from each client to the nearest VOD server (the original VOD system or its replica) is k or less.

Problem Description (3/3)

- Given a tree network, a server S which has VOD system, and a positive integer k,
- find the minimum number of replicas necessary so that each client is within distance k from the nearest server which has the original VOD system or its replica. For example, consider the following tree network.

Example



- **The set of clients is {1, 6, 7, 8, 9, 10, 11, 13}, the set of servers is {2, 3, 4, 5, 12, 14}, and the original VOD server is located at node 12. For k = 2, the quality of service is not guaranteed with one VOD server at node 12 because the clients in {6, 7, 8, 9, 10} are away from VOD server at distance > k. Therefore, we need one or more replicas. When one replica is placed at node 4, the distance from each client to the nearest server of {12, 4} is less than or equal to 2. The minimum number of the needed replicas is one for this example.**

Input

- Your program is to read the input from standard input. The input consists of T test cases. The number of test cases (T) is given in the first line of the input.
- The first line of each test case contains an integer n ($3 \le n \le 1{,}000$) which is the number of nodes of the tree network. The next line contains two integers s ($1 \le s \le n$) and k ($k \ge 1$) where s is the VOD server and k is the distance value for ensuring the quality of service. In the following n−1 lines, each line contains a pair of nodes which represent an edge of the tree network.

Output

- **Your program is to write to standard output. Print exactly one line for each test case. The line should contain an integer that is the minimum number of the needed replicas.**

Sample Input / Output

```
1
14
12 2
1 2
2 3
3 4
4 5
5 6
7 5
8 5
4 9
10 3
2 12
12 14
13 14
14 11
```

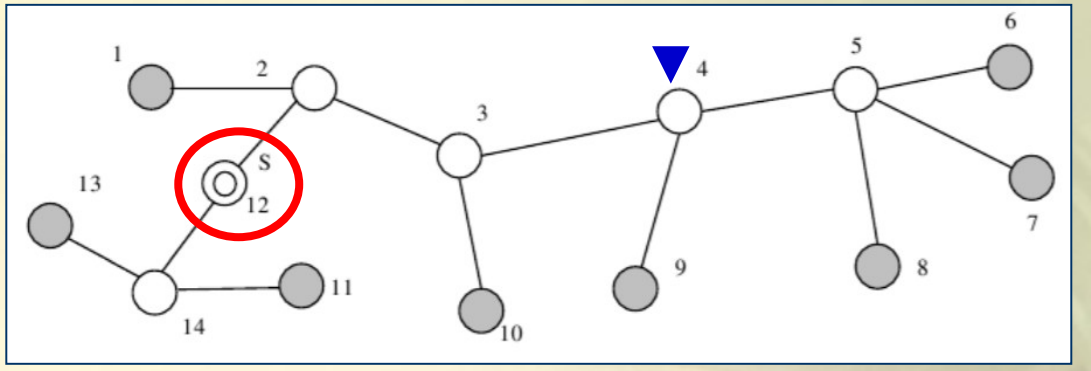
Number of test case

Number of nodes

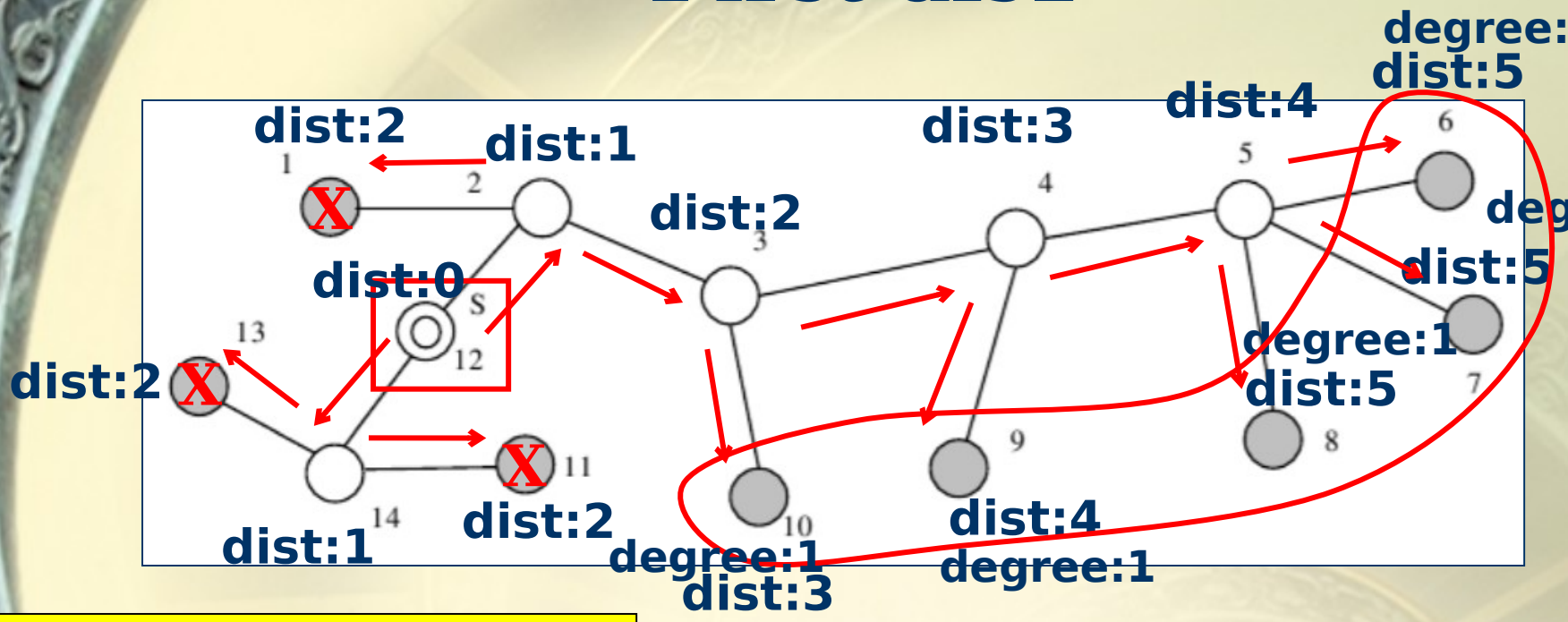
Server , k

Tree link

```
1
```



First dfs1



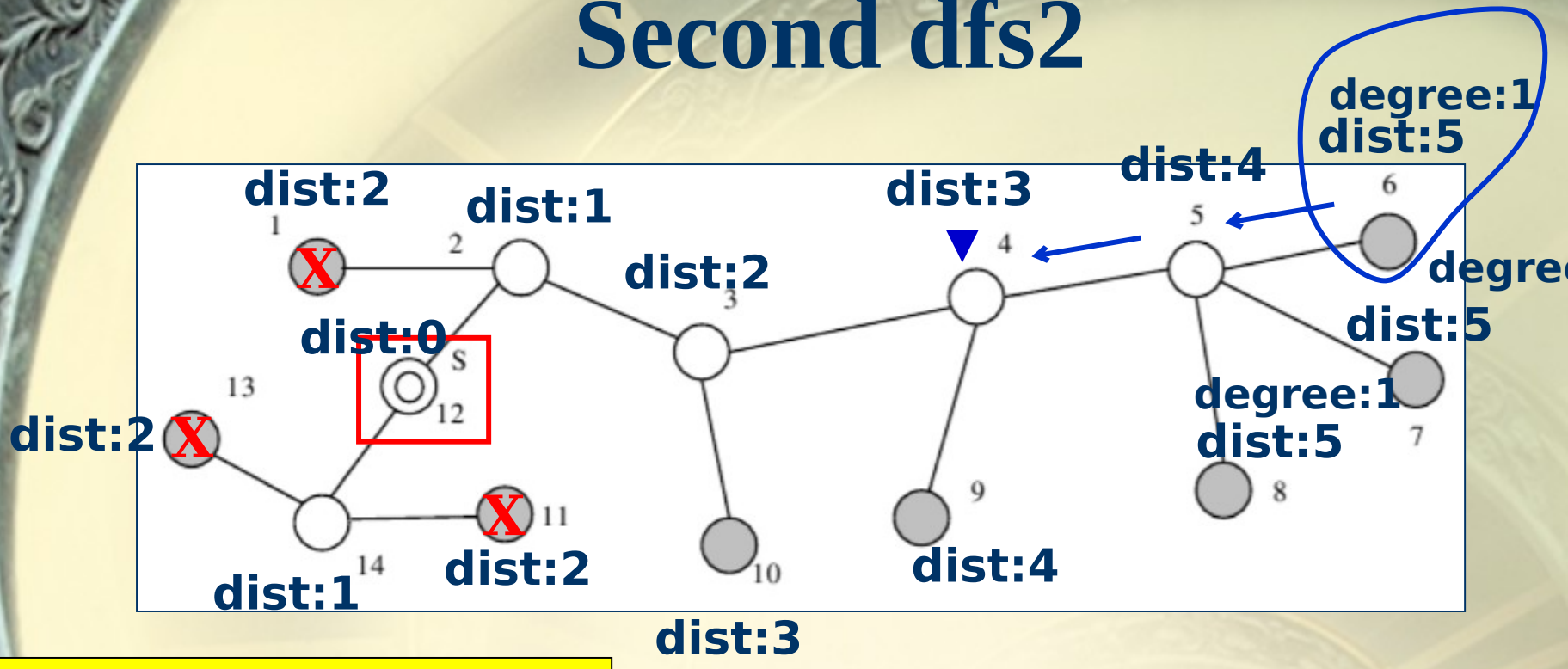
nodes

1	
2	
3	→ 10
4	→ 9
5	→ 6, 7, 8

p

1	2	3	4	5	6	7	8	9	10	11	12	13
2	12	2	3	4	5	5	5	4	3	14	0	14

Second dfs2



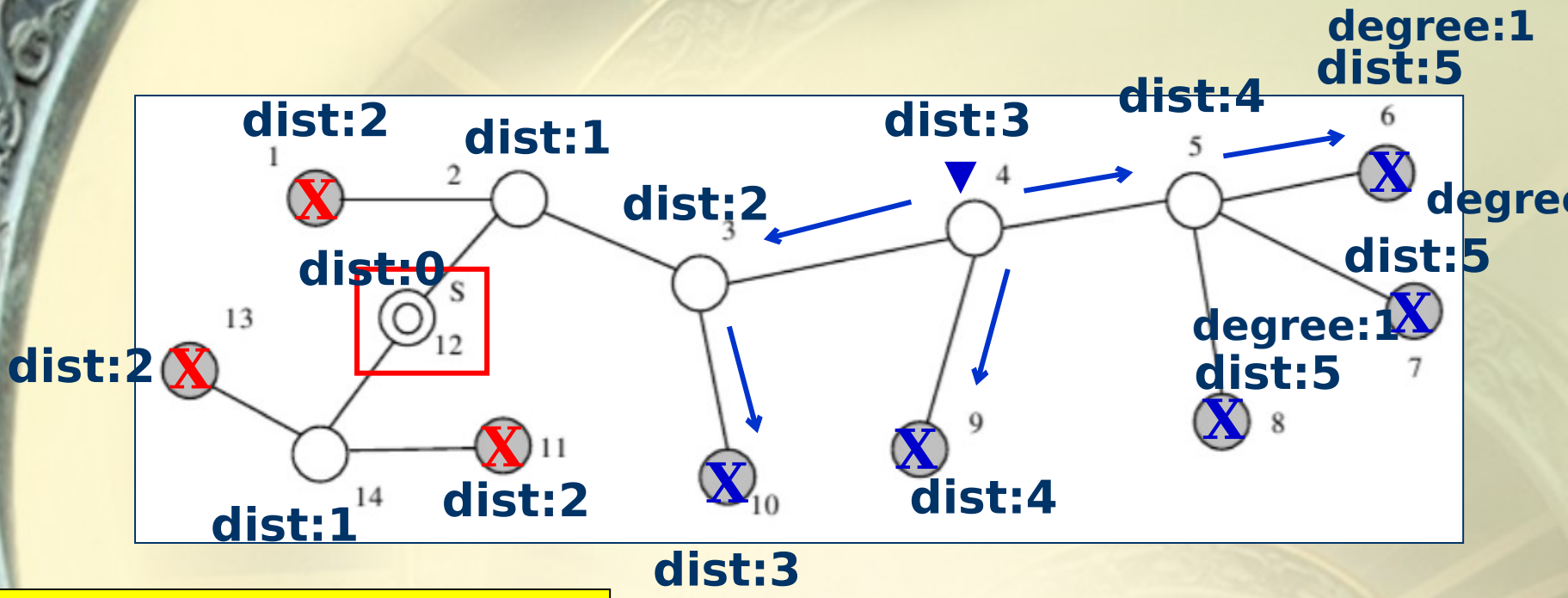
nodes

1	
2	
3 →	10
4 →	9
5 →	6 → 7 → 8

c

1	2	3	4	5	6	7	8	9	10	11	12	13	
0	0	0	0	0	0	0	0	0	0	0	0	0	

Second dfs2



nodes

1	
2	
3 →	10
4 →	9
5 →	6 → 7 → 8

c

1	2	3	4	5	6	7	8	9	10	11	12	13	
0	0	0	0	0	0	0	0	0	0	0	0	0	

Depth-First Search: Algorithm

```
DFS(G(V,E))
{
   for each vertex u   V
   {
      u->color = WHITE;
   }
   time = 0;
   for each vertex u   V
   {
      if (u->color ==
   WHITE)
         DFS_Visit(u);
   }
}
```

First u is root

```
DFS_Visit(u)
{
   u->color = GRAY;
   time = time+1;
   u->d = time;
   for each v   u->Adj[]
   {
      if (v->color == WHITE)
         DFS_Visit(v);
   }
   u->color = BLACK;
   time = time+1;
   u->f = time;
}
```

```
60  int main()
61  {
62      int T;
63      scanf("%d", &T);
64      while (T--)
65      {
66          scanf("%d%d%d", &n, &s, &k);
67          for (int i = 1; i <= n; i++)
68          {
69              g[i].clear();
70              nodes[i].clear();
71          }
72          for (int i = 0; i < n - 1; i++)
73          {
74              int a, b;
75              scanf("%d%d", &a, &b);
76              g[a].push_back(b);
77              g[b].push_back(a);
78          }
79          dfs(s, -1, 0); //dfs(server, father, dist)
80          printf("%d\n", solve());
81      }
82      return 0;
83  }
```

```cpp
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1005;

vector<int> g[maxn], nodes[maxn];
int n, s, k, p[maxn];
bool c[maxn];

void dfs1(int u, int f, int dist)
{
    p[u] = f; //parent of node "u" is set to node "f"
    int nd = g[u].size(); //number of degree of node "u"

    if (nd == 1 && dist > k)
        nodes[dist].push_back(u);

    for (int i = 0; i < nd; i++)
    {
        int v = g[u][i];
        if (v != f) dfs1(v, u, dist + 1);
    }
}
```

```
36  int solve()
37  {
38      int ans = 0;
39      memset(c,false, sizeof(c));
40
41      for (int dist = (n-1); dist > k; dist--)
42      {
43          for (int i = 0; i < nodes[dist].size(); i++)
44          {
45              int u = nodes[dist][i];
46              if (c[u])
47                  continue;
48
49              int v = u;
50              for (int j = 0; j < k; j++)
51                  v = p[v];
52
53              dfs2(v, -1, 0);
54              ans++;
55          }
56      }
57      return ans;
58  }
```

```
24  void dfs2(int u, int f, int d)
25  {
26      c[u] = true;
27      int nd = g[u].size();
28
29      for (int i = 0; i < nd; i++)
30      {
31          int v = g[u][i];
32          if (v != f && d < k) dfs2(v, u, d + 1);
33      }
34  }
```