Editorial พีทซิลล่าตะลอนกิน (PZ_Food Tour)

ที่มา: ข้อสอบท้ายค่ายสองศูนย์ ม.บูรพา รุ่น 17

Problem Statement

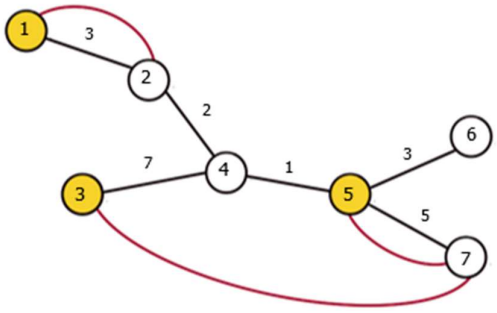
อาณาจักรแห่งหนึ่ง เป็นอาณาจักรที่โด่งดังในเรื่องของอาหารเมนูลับของอาณาจักรทั้งหมด M เมนูซึ่งได้รับการยกย่องจากเชฟทั่วโลก ในอาณาจักรแห่งนี้ มีหมู่บ้านอยู่ N หมู่บ้าน และมีถนนเชื่อมระหว่างหมู่บ้านทั้งหมด N-1 เส้นซึ่งใช้สำหรับการเดินทางไปมาระหว่างหมู่บ้าน รับประกันว่าทุกหมู่บ้านจะเชื่อมต่อกันทั้งหมด

บางหมู่บ้านในอาณาจักรอาจครอบครองตำราลับของอาณาจักรไว้ 1 เมนู ซึ่งเมนูนี้จำเป็นต้องใช้ผักเป็นวัตถุดิบพิเศษที่สามารถเติบโตได้ในถ้ำใต้ดินภายใต้หมู่บ้านนั้น ๆ เท่านั้น ซึ่งถ้ำใต้ดินนี้จะครอบคลุมพื้นที่ของหมู่บ้านอย่างน้อย 2 หมู่บ้านเสมอ โดยหมู่บ้านที่มีเส้นทางใต้ดินเชื่อมหากัน จะถือว่ามีถ้ำใต้ดินแห่งเดียวกันและสามารถผลิตเมนูลับชนิดเดียวกันได้ ส่วนหมู่บ้านที่ไม่มีถ้ำใต้ดินก็จะไม่สามารถผลิตเมนูลับใด ๆ ได้ กล่าวคือ ถ้ำใต้ดินสามารถผลิตเมนูลับได้ หากหมู่บ้านคู่ใดมีทางเชื่อมใต้ดินหากันจะอยู่ถ้ำเดียวกัน และผลิตเมนูลับชนิดเดียวกันได้เพียง 1 ชนิด

พีทซิลล่า เป็นพนักงานออฟฟิศผู้มีความสนใจด้านอาหารเป็นอย่างมาก เขาต้องการที่จะตะลุยกินอาหารทั้ง M ชนิดในอาณาจักรในช่วงวันหยุดของเขา ซึ่งเขาได้จองตั๋วเครื่องบินไว้แล้วแต่ดันเกิดโรคระบาดครั้งใหญ่ขึ้นมาทำให้ที่พักและร้านอาหารของหมู่บ้านต่าง ๆ ในอาณาจักรมีเวลาเปิดปิดอย่างไม่แน่นอน (การเปิดปิดของที่พักกับร้านอาหารในหมู่บ้านเดียวกันไม่มีความสัมพันธ์กัน) และเนื่องจากเขาได้เสียเงินไปกับตั๋วเครื่องบินจำนวนมาก เขาจึงไม่สามารถยกเลิกการเดินทางในครั้งนี้ได้

อาณาจักรแห่งนี้มีที่พักอยู่ที่ K หมู่บ้าน ในหนึ่งวัน เขาจะต้องเลือกพักในที่พักที่ยังเปิดอยู่ และเดินทางไปลิ้มลอง 1 เมนู ที่เขายังไม่เคยลิ้มลองมาก่อนในร้านอาหารที่ยังเปิดอยู่ แล้วเดินทางกลับที่พักเดิมของเขาแล้ว วันถัดไปเขาก็จะทำเช่นเดิม และเพื่อลิ้มลองเมนูลับทั้งหมด M เมนู เขาจึงจำเป็นต้องใช้เวลาทั้งสิ้น M วัน (ไม่จำเป็นต้องพักที่เดิมตลอดทั้ง M วันก็ได้)

แต่เนื่องจากในแต่ละวัน เขาไม่รู้ช่วงเวลาเปิดปิดของที่พักและร้านอาหาร เขาจึงสงสัยว่า ระยะทางที่น้อยที่สุดระหว่างที่พักและร้านอาหารที่เปิดอยู่ที่เขาต้องเดินทางในแต่ละวัน รวมกันจะเป็นเท่าใดในกรณีที่แย่ที่สุด หรือก็คือกรณีที่ระยะห่างระหว่างร้านอาหารที่เปิดและที่พักที่เขาได้พักมีระยะห่างมากที่สุดระหว่างที่พักใด ๆ และร้านอาหารใด ๆ ที่มีเมนูลับที่เขาต้องการลิ้มลอง รับประกันว่า ในแต่ละวันจะมีร้านและที่พักเปิดอย่างน้อยอย่างละ 1 ที่



ตัวอย่างการเดินทาง

อาณาจักรแห่งนี้มีหมู่บ้าน 7 หมู่บ้าน มีที่พัก 3 หมู่บ้าน (ระบุด้วยวงกลมสีเหลือง) และมีเส้นทางระหว่างหมู่บ้าน (ระบุโดยเส้นสีดำ) และเส้นทางใต้ดินระหว่างหมู่บ้าน (ระบุด้วยสีแดง) ทำให้หมู่บ้าน 1 และ 2 สามารถผลิต

เมนูลับชนิดเดียวกันได้ (แทนด้วย เมนูลับ 1) นอกจากนี้ยังมีหมู่บ้าน 3, 5 และ 7 ที่สามารถผลิตเมนูลับชนิดเดียวกันได้ (แทนด้วย เมนูลับ 2) รวมเป็นเมนูลับ 2 เมนู

จะเห็นได้ว่าถ้าพีทซิลล่า ต้องการจะลิ้มลองเมนูลับ 1 ในกรณีที่แย่ที่สุด คือมีแค่ที่พักหมู่บ้าน 3 และร้านอาหารหมู่บ้าน 1 เท่านั้นที่เปิด เขาต้องเดินทาง (7+2+3) + (3+2+7) = 24 หน่วย ซึ่งมากที่สุดในบรรดาคู่ที่พักและร้านอาหารเมนูลับ 1 ใด ๆ แล้ว

ในทำนองเดียวกันสำหรับเมนูลับ 2 คือมีแค่ที่พักหมู่บ้าน 3 และร้านอาหารหมู่บ้าน 7 เท่านั้นที่เปิด ทำให้เขาต้องเดินทาง (7+1+5) + (5+1+7) = 26 หน่วย รวมเป็น 24+26 = 50 หน่วย รับประกันว่าภายใน M วัน สามารถกินเมนูลับได้ครบทุกเมนู

งานของคุณ

จงเขียนโปรแกรมเพื่อหาระยะทางที่น้อยที่สุดที่พีทซิลล่าต้องเดินทางในกรณีที่แย่ที่สุด

ข้อมูลนำเข้า

บรรทัดแรก รับจำนวนเต็มบวก Q แทนจำนวนคำถาม โดยที่ Q ไม่เกิน 5 ในแต่ละคำถาม

บรรทัดแรก รับจำนวนเต็มบวก N M R K ห่างกันหนึ่งช่องว่าง โดยที่ 1 <= N <= 40,000 และ 1 <= M <= 500 และ 1 <= R, K <= 40,000

อีก N-1 บรรทัดถัดมา รับจำนวนเต็มบวก u v w แทนเส้นทางระหว่างหมู่บ้าน u และ v ที่ใช้เวลาในการเดินทาง w โดยที่ 1 <= u, v <= N และ w <= 100,000

อีก R บรรทัดถัดมา รับ u v แทนทางเชื่อมใต้ดินระหว่างหมู่บ้าน u และ v โดยที่ 1 <= u, v <= N

บรรทัดถัดมา รับจำนวนเต็มบวก K ตัว ห่างกัน 1 ช่องว่าง แทนหมายเลขหมู่บ้านที่มีที่พัก

ข้อมูลส่งออก

มีทั้งสิ้น Q บรรทัด แต่ละบรรทัดให้แสดงระยะทางที่น้อยที่สุดที่พีทซิลล่าจะต้องใช้ตลอดการเดินทาง ในกรณีที่แย่ที่สุด

ตัวอย่างที่ 1

ข้อมูลนำเข้า	ข้อมูลส่งออก
2 7 2 3 3 1 2 3 2 4 2 3 4 7 4 5 1 5 6 3 5 7 5 1 2 3 5 5 7 1 3 5 5 1 2 3 1 2 4	50 18

2 4 3 4 5 1 3 4 2 1 2 2 3 1 3 5	

คำอธิบายตัวอย่างที่ 1

มีทั้งสิ้น 2 คำถาม ดังนี้

คำถามแรก เป็นไปตามตัวอย่างที่อธิบายในโจทย์

คำถามที่สอง ในกรณีที่แย่ที่สุด เขาจะได้เข้าพักในหมู่บ้านที่ 1 และต้องเดินทางไปลิ้มลองอาหารที่หมู่บ้านที่ 3 (หรือได้เข้าพักในหมู่บ้านที่ 3 และต้องเดินทางไปลิ้มลองอาหารที่หมู่บ้านที่ 1) ซึ่งพีทซิลล่าจะใช้ระยะทางทั้งหมดเป็น 4+3+2+2+3+4 = 18 หน่วย

เกณฑ์การให้คะแนน

ปัญหาย่อย 1: (20 %)

20% ของชุดข้อมูลทดสอบ จะมี K = 1

ปัญหาย่อย 2: (20 %)

20% ของชุดข้อมูลทดสอบ จะมี K <= 3

ปัญหาย่อย 3: (60 %)

60% ของชุดข้อมูลทดสอบเป็นไปตามเงื่อนไขของโจทย์

ซึ่งการจะได้คะแนนเต็มในข้อนี้ โปรแกรมที่ส่งจะต้องทำงานได้อย่างมีประสิทธิภาพ

ข้อจำกัดของโจทย์

ข้อจำกัดของโจทย์	เงื่อนไข
ชื่อโจทย์	พีทซิลล่าตะลอนกิน (PZ_Food Tour)
ข้อจำกัดของการใช้เวลาประมวลผลไม่เกิน	**1 วินาที**
การใช้หน่วยความจำในแต่ละชุดทดสอบไม่เกิน	128 MB
คะแนนเต็ม	100

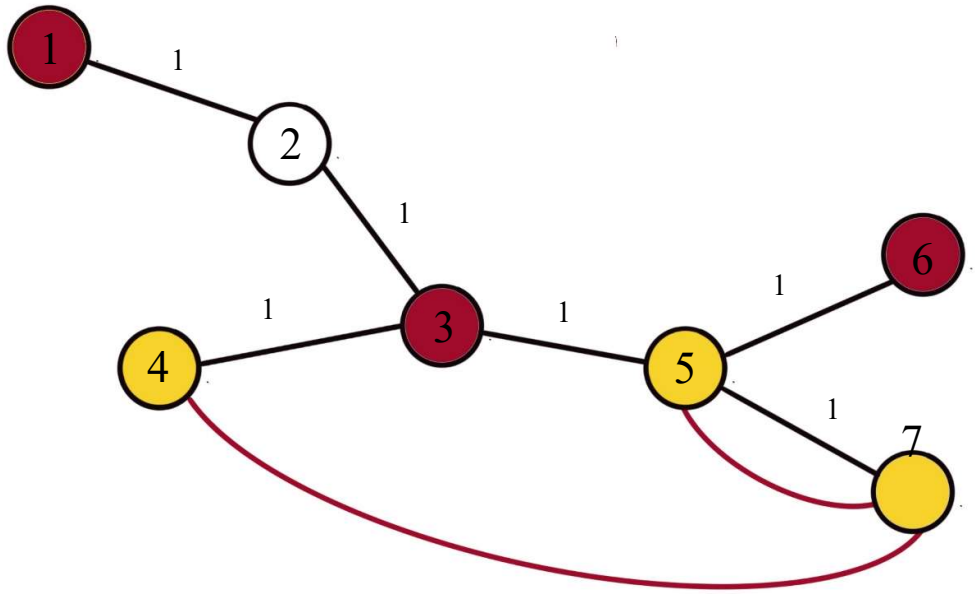
ข้อกำหนดส่วนหัวของโปรแกรม

สำหรับผู้เข้าแข่งขันที่เขียนภาษา C	สำหรับผู้เข้าแข่งขันที่เขียนภาษา C++
/* TASK: PZ_Food Tour LANG: C AUTHOR: YourName YourLastname SCHOOL: Your school */	/* TASK: PZ_Food Tour LANG: CPP AUTHOR: YourName YourLastname SCHOOL: Your school */

<u>แนวคิด</u>

จากโจทย์สิ่งที่เราต้องการหาคือ ผลรวมของระยะห่างระหว่างคู่ของที่พักใด ๆ ไปยังร้านอาหารร้านใดร้านหนึ่งในกลุ่มของร้านอาหารที่ขายอาหารชนิดเดียวกันของแต่ละกลุ่ม ที่ให้ระยะทางมากที่สุด (ผลรวมที่ได้จะนำไปคูณสองเพื่อตอบแทนระยะทางไปและกลับ)

<u>พิจารณากรณีที่มีอาหารเพียง 1 ชนิด (ทุกหมู่บ้านที่มีทางใต้ดินเชื่อมกันอยู่ทั้งหมด) และเส้นทางแต่ละเส้นทางมีระยะทาง 1 หน่วยทั้งหมด</u>



ให้เมืองสีแดงแทน เมืองที่มีที่พักให้บริการ ส่วนเมืองสีเหลืองแทนเมืองที่มีร้านขายอาหาร เนื่องจากเมืองที่มีร้านขายอาหารมีทางใต้ดินที่เชื่อมเข้าด้วยกันทั้งหมด ร้านอาหารในเมืองสีเหลืองทั้งหมดจึงขายอาหารชนิดเดียวกัน สิ่งที่เราต้องการจะทราบก็คือระยะห่างระหว่างเมืองที่มีสีแดงและเมืองที่มีสีเหลืองซึ่งให้ระยะทางมากที่สุด จากตัวอย่างข้างต้น คู่ของเมืองที่ให้ระยะห่างมากที่สุดคือคู่ของเมือง 1 และ เมือง 7 ซึ่งให้ระยะทางเท่ากับ 4 หน่วย คำตอบของตัวอย่างนี้จึงจะเท่ากับ 4 x 2 = 8 หน่วย

ในทำนองเดียวกัน ในโจทย์จริงนั้น กลุ่มของร้านอาหารจะมีทั้งหมด M กลุ่มซึ่งแต่ละกลุ่มจะขายเมนูที่แตกต่างกัน M เมนู แทนกลุ่มของเมืองที่ระบายสี C_1, C_2, C_3, ..., C_M สิ่งที่เราต้องการหาก็คือ ระยะห่างของคู่เมือง M คู่ที่เชื่อมระหว่าง เมืองที่มีที่พักกับเมืองที่ระบายสีแต่ละสีทั้ง M สี แล้วจึงนำมาหาผลรวมคูณสอง

Solutions

ในขั้นตอนแรกเราจะทำการจัดกลุ่มของเมืองต่าง ๆ ตามตำราเมนูลับที่เมืองนั้น ๆ ได้ครอบครองเอาไว้ หรือไม่ได้ครอบครองตำราลับเมนูไหนเลย โดยใช้อัลกอริทึมที่มีชื่อว่า **DSU – Disjoint Set Union** ในการจำแนกเมืองตามกลุ่มจากเส้นทางเชื่อมใต้ดินระหว่างหมู่บ้าน

DSU Code example

```
13. int p[N];
14. long long food[510],dis[N];
15. bool res[N];
16. vector<pair<int,int>> g[N];
17. int findr(int i){
18.     if(p[i]==i) return i;
19.     return p[i]=findr(p[i]);
20. }
21. int main()
22. {
23.     int q,n,m,r,k,u,v,w,idx=0;
24.     long long ans=0;
25.     scanf("%d",&q);
26.     while(q--){
27.         int idx=ans=0;
28.         scanf("%d %d %d %d",&n,&m,&r,&k);
29.         for(int i=1;i<=n;i++)p[i]=i;
30.         for(int i=1;i<n;i++){
31.             scanf("%d %d %d",&u,&v,&w);
32.             g[u].push_back({v,w});
33.             g[v].push_back({u,w});
34.         }
35.         for(int i=1;i<=r;i++){
36.             scanf("%d %d",&u,&v);
37.             p[u]=findr(p[v]);
38.             res[u]=res[v]=true;
39.         }
```

หลังจากนั้น เราจะคำนวณหาผลรวมของระยะห่างระหว่างคู่ของที่พักใด ๆ ไปยังร้านอาหารร้านใดร้านหนึ่งในกลุ่มของร้านอาหารที่ขายอาหารชนิดเดียวกันของแต่ละกลุ่ม ที่ให้ระยะทางมากที่สุด ซึ่งวิธีการจะแตกต่างไปตามปัญหาย่อยแต่ละปัญหาย่อยดังนี้

ปัญหาย่อยที่ 1 จะมี K เท่ากับ 1

ในปัญหาย่อยนี้จะมีเมืองที่มีที่พักเพียง 1 ที่เท่านั้น เราสามารถใช้ **Dijkstra Algorithm** ซึ่งเป็นอัลกอริทึมสำหรับการหา **single source shortest path** ในการแก้ปัญหานี้ได้ โดยการหาระยะทางจากเมืองที่มีที่พักไปยังเมืองต่าง ๆ แล้วจึงนำระยะทางที่ใช้ในการเดินทางไปยังเมืองที่มีเมนูอาหารชนิดเดียวกันมาเทียบกัน เพื่อที่จะนำระยะทางที่มากที่สุดมาคำนวณหาผลรวมสำหรับหาคำตอบ

Algorithm: Dijkstra

Time complexity: $O(|E|\log|V|)$

Full source code

```cpp
#include<bits/stdc++.h>
#define N 60100
using namespace std;
struct A{
    int u;
    long long v;
    bool operator <(const A & o) const{
        return v>o.v;
    }
};
priority_queue< A > h;
unordered_map<int,int> mp;
int p[N];
long long food[510],dis[N];
bool res[N];
vector<pair<int,int>> g[N];
int findr(int i){
    if(p[i]==i) return i;
    return p[i]=findr(p[i]);
}
int main()
{
    int q,n,m,r,k,u,v,w,idx=0;
    long long ans=0;
    scanf("%d",&q);
    while(q--){
        int idx=ans=0;
        scanf("%d %d %d %d",&n,&m,&r,&k);
        for(int i=1;i<=n;i++)p[i]=i;
        for(int i=1;i<n;i++){
            scanf("%d %d %d",&u,&v,&w);
            g[u].push_back({v,w});
            g[v].push_back({u,w});
        }
        for(int i=1;i<=r;i++){
            scanf("%d %d",&u,&v);
            p[u]=findr(p[v]);
            res[u]=res[v]=true;
        }
        scanf("%d",&k);
        for(int i=1;i<=n;i++){
            dis[i]=3e9;
            if(res[i] && !mp[findr(i)]) mp[findr(i)]=++idx;
        }
            res[i]=0;
        }
        mp.clear();
    }
    return 0;
}
```

ปัญหาย่อยที่ 2 จะมี K น้อยกว่าเท่ากับ 3

ในปัญหาย่อยนี้เราสามารถใช้วิธีการเช่นเดียวกันกับในปัญหาย่อยที่ 1 ได้ โดยเราจะทำการ Dijkstra ทั้งหมด K รอบ แล้วจึงนำระยะทางที่ได้จากทุกรอบมาใช้ในการเปรียบเทียบ

Algorithm: Dijkstra

Time complexity: `O(K(|E|log|V|))`

Full source code

```cpp
#include<bits/stdc++.h>
#define N 60100
using namespace std;
struct A{
    int u;
    long long v;
    bool operator <(const A & o) const{
        return v>o.v;
    }
};
priority_queue< A > h;
unordered_map<int,int> mp;
int p[N];
long long food[510],dis[N];
bool res[N];
vector<pair<int,int>> g[N];
int findr(int i){
    if(p[i]==i) return i;
    return p[i]=findr(p[i]);
}
int main()
{
    int q,n,m,r,k,u,v,w,idx=0,now;
    long long ans=0;
    scanf("%d",&q);
    while(q--){
        idx=ans=0;
        scanf("%d %d %d %d",&n,&m,&r,&k);
        for(int i=1;i<=n;i++)p[i]=i;
        for(int i=1;i<n;i++){
            scanf("%d %d %d",&u,&v,&w);
            g[u].push_back({v,w});
            g[v].push_back({u,w});
        }
        for(int i=1;i<=r;i++){
            scanf("%d %d",&u,&v);
            p[u]=findr(p[v]);
            res[u]=res[v]=true;
        }
        for(int i=1;i<=n;i++){
            if(res[i] && !mp[findr(i)]) mp[findr(i)]=++idx;
        }
```

```
        while(k--){
            scanf("%d",&now);
            for(int i=1;i<=n;i++){
                dis[i]=3e9;
            }
            dis[now]=0;
            h.push({now,0});
            while(!h.empty()){
                u=h.top().u;
                v=h.top().v;
                h.pop();
                if(res[u] && food[mp[findr(u)]]<dis[u]) food[mp[findr(u)]]=dis[u];
                for(auto x: g[u]){
                    if(dis[x.first]>dis[u]+x.second){
                        dis[x.first]=dis[u]+x.second;
                        h.push({x.first,dis[x.first]});
                    }
                }
            }
        }
        for(int i=1;i<=m;i++) ans+=food[i],food[i]=0;
        printf("%lld\n",ans*2);
        for(int i=1;i<=n;i++){
            g[i].clear();
            res[i]=0;
        }
        mp.clear();
    }
    return 0;
}
```

<u>ปัญหาย่อยที่ 3</u> จะมี K น้อยกว่าเท่ากับ 40,000

สำหรับปัญหาย่อยนี้ เราไม่สามารถใช้ Dijkstra Algorithm เช่นเดียวกับในปัญหาย่อยก่อนๆได้ เนื่องจาก time complexity ของอัลกอริทึม ซึ่งจะทำให้เกิด time limit exceeded ทำให้เราจึงต้องใช้วิธีการอื่นในการแก้ปัญหาย่อยนี้ สำหรับปัญหาย่อยนี้นั้น จะเป็นการทำ dynamic programming on tree หรือ DP on tree โดยเราจะทำการ DFS Depth First Search บนต้นไม้ที่สร้างจากถนนที่ใช้ในการเดินทางระหว่างเมืองต่าง ๆ ที่มีทั้งหมด N-1 เส้น และรับประกันว่าทุกเมืองสามารถไปหากันได้ ทำให้เราทราบได้ว่า เมืองแต่ละเมืองจะถูกเชื่อมกันในรูปแบบของ tree ระหว่างการ backtracking ของ DFS เราจะทำการเก็บค่าระยะทางของเมืองที่เป็นที่พักและเป็น node ใน subtree ของ node ปัจจุบันของเรา ที่มีระยะทางมากที่สุดจากเมืองปัจจุบันของเราเอาไว้ ในทำนองเดียวกันเราจะเก็บระยะทางที่มากที่สุดระหว่างเมืองปัจจุบันกับเมืองที่มี node อยู่ใน subtree ของ node ปัจจุบันและเป็นเมืองที่มีเมนูลับต่าง ๆ ทั้ง M เมนูเอาไว้เช่นเดียวกัน เราจะทำการเช็คโดยการวนเปรียบเทียบระหว่างระยะทางไปยัง node ลูก และค่าที่เก็บเอาไว้ใน node ลูกของ node ปัจจุบันแต่ละ node เพื่อที่จะหาค่าระยะทางมากที่สุดจากเมืองปัจจุบันไปยังเมืองที่มีที่พักและเมืองที่มีเมนูอาหารทั้ง M เมนูเอาไว้ ระหว่างการเก็บค่ามากที่สุดเราก็จะทำการเก็บค่าคำตอบไปด้วยโดยการเช็คหาค่า max ของผลรวมระหว่างระยะทางไปยังเมืองที่มีที่พักที่อยู่ไกลที่สุดจากเมืองปัจจุบัน และระยะทางไปยังเมืองที่มีเมนูอาหารแต่ละชนิดที่อยู่ไกลที่สุดจากเมืองปัจจุบัน โดยที่ subtree ของ node ลูกที่นำมาคิดระยะทางของเมืองที่พักและเมืองที่มีเมนูอาหารนั้นจะต้องมี root ซึ่งเป็น node ลูกของ node ปัจจุบันคนละ node กันเพื่อป้องกันกรณีเส้นทางซ้อนทับ ซึ่งจะทำให้ค่าที่นำมาคิดนั้นไม่ใช้ shortest path ระหว่างสอง nodes ได้ หรือถ้าหากเป็นกรณีที่ node ปัจจุบันเป็นเมืองที่มีที่พัก หรือเป็นเมืองที่มีเมนูลับ เราก็จำเป็นจะต้องเปรียบเทียบคำตอบของเรากับระยะทางจากที่ node ปัจจุบันใช้เพื่อไปยังที่พักหรือเมืองที่มีเมนูแต่ละเมนูที่มีระยะทางมากที่สุดด้วย

Algorithm: Dynamic programming on tree

Time complexity: O(NM)

Full source code

```cpp
#include<bits/stdc++.h>
using namespace std;
#define N 60100
struct A{
    long long ht,fd[510];
};
A town[N];
int p[N],idx=0,mp[N];
long long food[510];
bool hotel[N],res[N];
// unordered_map< int,int > mp;
vector< pair<int,int> > g[N];
int findr(int i){
    if(p[i]==i) return i;
    return p[i]=findr(p[i]);
}
void dfs(int now,int la){
    town[now].ht=-3e9;
    for(int i=1;i<=idx;i++)town[now].fd[i]=-3e9;
    if(res[now]) town[now].fd[mp[findr(now)]]=0;
    if(hotel[now]) town[now].ht=0;
    for(auto x: g[now]){
        if(x.first==la) continue;
        dfs(x.first,now);
        for(int i=1;i<=idx;i++){
            food[i]=max(food[i],town[now].ht+town[x.first].fd[i]+x.second);
            food[i]=max(food[i],town[now].fd[i]+town[x.first].ht+x.second);
            town[now].fd[i]=max(town[now].fd[i],town[x.first].fd[i]+x.second);
        }
        town[now].ht=max(town[now].ht,town[x.first].ht+x.second);
    }
    return ;
}
int main()
{
    int q,n,m,r,k,u,v,w;
    scanf("%d",&q);
    while(q--){
        idx=0;
        scanf("%d %d %d %d",&n,&m,&r,&k);
        for(int i=1;i<n;i++){
            scanf("%d %d %d",&u,&v,&w);
            g[u].push_back({v,w});
            g[v].push_back({u,w});
        }
        for(int i=1;i<=n;i++) p[i]=i;
        for(int i=1;i<=r;i++){
            scanf("%d %d",&u,&v);
            p[findr(u)]=findr(v);
            res[u]=res[v]=true;
        }
        for(int i=1;i<=n;i++){
            if(res[i] && !mp[findr(i)]) mp[findr(i)]=++idx;
        }
        for(int i=1;i<=k;i++){
            scanf("%d",&u);
            hotel[u]=1;
        }
        dfs(1,-1);
        long long ans=0;
```

```
        for(int i=1;i<=idx;i++){
            ans+=food[i];
            food[i]=0;
        }
        printf("%lld\n",ans*2);
        for(int i=1;i<=n;i++){
            g[i].clear();
            mp[i]=0;
            res[i]=hotel[i]=0;
        }
    }
    return 0;
}
```