

[洛谷](#) / [题目列表](#) / [题目详情](#) / [查看题解](#)

应用 >>



题库



题单



比赛



记录



讨论

P8817 [CSP-S 2022] 假期计划 题解

[返回题目](#)

本题目已不接受新题解提交

当题目的题解数量、做题思路已足够丰富，题目过于简单，或处于月赛保护期时，题解提交入口会被关闭。

不要把题解发布在题目讨论区。

题解仅供学习参考使用

抄袭、复制题解，以达到刷 AC 率/AC 数量或其他目的的行为，在洛谷是严格禁止的。

洛谷非常重视**学术诚信**。此类行为将会导致您成为**作弊者**。具体细则请查看[洛谷社区规则](#)。

提交题解前请务必阅读[《洛谷主题库题解规范》](#)。



洛谷出品教材

深入浅出 程序设计竞赛

基础篇

送官方特典

明信片/kkk签名信封

限定赠品

官方网店

洛谷推荐

关闭

10 篇题解

默认排序 按时间排序



dbxxx



创建时间: 2022-10-31 01:34:35

[在 Ta 的博客查看](#)

欢迎您到我的博客中查看本文，谢谢！

下文中，

- u 和 v 之间**可达**意为 u 和 v 之间可以经过不多于 k 次转车到达，即 u 到 v 的距离不多于 $k + 1$ 。
- 一个点 u **在家附近**，意为 u 和 1 之间可达。

注意，为方便，特殊地，我们认为**自己和自己不可达**。

观察数据范围， n^2 可过，已经足够处理出每个点对之间的距离了。因此首先应该采用 bfs， n^2 计算任意点对之间是否可达。

注意：对于满足边权全为 1 的图，单源最短路可以做到 $\mathcal{O}(n)$ 的复杂度（采用 bfs）；

对于满足边权只有 0, 1 两种的图，单源最短路也可以做到 $\mathcal{O}(n)$ 的复杂度（采用 0-1 bfs）。

我们考虑一条 $1 - a - b - c - d - 1$ 的路径， n^4 的枚举是不可行的。

很直接的想法是，依次考虑 a, b, c, d ，发现贪心是不对的，从 1 选择了更大的权值景点 a ，但是可能接下来能到达的 b 就小的可怜；而选一个权值稍小一点的 a ，可能可达的 b 会很大。

但有一种贪心是对的，那就是确定了 a, b, c ，选择 d 的时候。如果我们确定了 c ，那么直接选择 c 可达的，在家附近的，不为 a 也不为 b 的权值最大的景点作为 d 即可。反正 d 之后没有景点了，所以可以直接贪心，没有后顾之忧。

由于环的对称性，可以发现确定了 b, c, d 之后，也可以贪心地选择 a 。

有了大体思路。

我们定义 $f(u)$ 表示 u 可达，且在家附近的权值最大的景点。

第一步：我们预处理出 $f(u)$ 。

第二步：直接 n^2 枚举，循环确定景点 b 和 c ($b \neq c$)，然后记 $a = f(b)$ ， $d = f(c)$ ，然后试图用 $w(a) + w(b) + w(c) + w(d)$ 更新答案，其中 $w(u)$ 表示景点 u 的权值。

发现重复性的细节是存在问题的，比如：会出现 $f(b) = c$ 的情况，此时让 $a = f(b)$ 会导致 $a = c$ ，这是不允许的。

不过，贪心思想还是不变的： a 应该尝试更换为 u 可达，且在家附近的**权值第二大**的景点。

更换定义， $f(u, k)$ 表示 u 可达，且在家附近的权值第 k 大的景点。

当发现 $f(b, 1) = c$ 时，将 a 设置为 $f(b, 2)$ 即可。

当发现 $f(c, 1) = b$ 时，将 d 设置为 $f(c, 2)$ 即可。

发现并没有完全解决：如果这样处理后的 a 和 d 仍然重复怎么办？

还是贪心思想，考虑把 a 或者 d 换成更小的那个比较一下就行，具体来说，比如原先 $a = f(b, 2)$ ，就把 a 下调为 $f(b, 3)$ ；或者原先 $d = f(c, 2)$ ，就把 d 下调为 $f(c, 3)$ ，然后比较两种方案哪种更好就行了。

重复性解决了，但是还有个细节：如果原先 $a = f(b, 1)$ ，发现 $a = d$ ，我们想下调 a 。是直接把 a 设置成 $f(b, 2)$ 吗？错误，我们还需要检查一下 $f(b, 2)$ 是否等于 c ，如果等于 c 还需要继续下调到 $f(b, 3)$ 。下调 d



应用 >>



题库



题单



比赛



记录



讨论

同理。

至于原先 $a = f(b, 2)$ 为啥下调 a 不需检查？因为如果原先 $a = f(b, 2)$ 了说明 $f(b, 1)$ 已经等于 c 了。所以 $f(b, 3)$ 肯定什么问题都没有。

如果直接这么写是没有问题的，但是麻烦了。下面是一种更好写的处理方法：

直接分别枚举 a 分别作为 $f(b, 1)$, $f(b, 2)$, $f(b, 3)$ 和 d 分别作为 $f(c, 1)$, $f(c, 2)$, $f(c, 3)$ 组成的共九种情况，检查互异性然后更新答案即可。

需要提前预处理出 $f(u, k \leq 3)$ ，其实只要在最开始 bfs 预处理的同时维护一下就行了。

注意某些点可达，还在家附近的点数可能没有 3 个，因此注意类似访问 $f(b, 3)$ 时产生的越界问题。

如果枚举到某个 b , c ，发现 b 或者 c 有对应点数不超过 3 的情况时，这组 b , c 不一定可以生成一组合法的解，属于正常现象。

题目保证全局上是有解的。

```

/*
 * @Author: crab-in-the-northeast
 * @Date: 2022-10-30 17:40:03
 * @Last Modified by: crab-in-the-northeast
 * @Last Modified time: 2022-10-30 19:07:52
 */

#include <bits/stdc++.h>
#define int long long
inline int read() {
    int x = 0;
    bool flag = true;
    char ch = getchar();
    while (!isdigit(ch)) {
        if (ch == '-')
            flag = false;
        ch = getchar();
    }
    while (isdigit(ch)) {
        x = (x << 1) + (x << 3) + ch - '0';
        ch = getchar();
    }
    if (flag)
        return x;
    return ~(x - 1);
}

const int maxn = 2505;
typedef std::pair<int, int> pii;

std::vector<int> G[maxn];
int w[maxn];
bool ok[maxn][maxn]; // u, v 是否可达
std::vector<int> f[maxn]; // f[u] 存放：可达 u 且可达 1 的前三大 v

int k;

int dis[maxn];

void bfs(int x) {
    std::memset(dis, -1, sizeof(dis));
    std::queue<int> q;
    q.push(x);
    dis[x] = 0;
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        for (int v : G[u]) {
            if (dis[v] == -1) {
                dis[v] = dis[u] + 1;
                q.push(v);
            }
        }
    }
    for (int v : G[x]) {
        f[x].push_back(v);
        if (f[x].size() == 3) break;
    }
}

```



应用 >>



题库



题单



比赛



记录



讨论

```

dis[X] = 0;

while (!q.empty()) {
    int u = q.front();
    q.pop();

    if (u != x) {
        ok[x][u] = true;
        if (x != 1 && ok[1][u]) {
            // printf("%lld %lld\n", x, u);
            f[x].push_back(u);
            std::sort(f[x].begin(), f[x].end(), [](int u, int v) {
                return w[u] > w[v];
            }); // 注意这里 sort 元素数量不超过 3，效率可看做常数
            if (f[x].size() > 3)
                f[x].pop_back();
        }
    }

    if (dis[u] == k + 1)
        continue;

    for (int v : G[u]) if (dis[v] == -1) {
        dis[v] = dis[u] + 1;
        q.push(v);
    }
}

inline bool gmx(int &a, int b) {
    return b > a ? a = b, true : false;
}

signed main() {
    int n = read(), m = read();
    k = read();
    for (int u = 2; u <= n; ++u)
        w[u] = read();

    while (m--) {
        int u = read(), v = read();
        G[u].push_back(v);
        G[v].push_back(u);
    }

    for (int u = 1; u <= n; ++u)
        bfs(u);

    int ans = 0;

    for (int b = 2; b <= n; ++b)
        for (int c = 2; c <= n; ++c) if (ok[b][c])
            for (int a : f[b])
                for (int d : f[c])
                    if (a != c && b != d && a != d)
                        // 其他不等关系天然满足了，只有这三组需要检验；
                        // 天然满足的不等关系：a != b, b != c, c != d，请读者自己思考为什么。
                        gmx(ans, w[a] + w[b] + w[c] + w[d]);

    printf("%lld\n", ans);
    return 0;
}

```

*H 不见得会带题解+讨论，请个文心知悉，勿勿！

👍 406



💬 58 条评论

收起 ^

应用 >>



题库



题单



比赛



记录



讨论



RedLycoris



创建时间：2022-11-02 15:26:14

在 Ta 的博客查看

这里是 [CSP-S 2022]假期计划 的另类做法。

- 先暴力bfs把所有能在 k 步内达到的点互相连边，复杂度 $O(n^2)$
- 然后随机染色。对于每个点，随机染上 $1 \dots 4$ 这四种颜色中的任意一种，然后就相当于选出有4个颜色不同的点的包含点1的环。**这样就避免了重复经过某个点的情况。**
- 显然这是原问题的弱化版，每次有 $(\frac{2}{4})$ 的可能性对，随个 200 次就能过洛谷的民间数据了。

```
//start coding at 2:35
//finish coding at 2:47
//finish debugging at 2:55
//Code by paulzrm
/*
虽千万人逆我之我仍执着
侠客行遍九州恩仇奔波
刀光剑影里是谁的明眸粲粲如星火
纵使我双掌倚天苍龙俘获
奈何降不住这人心如魔
所谓天下第一 怎围我天高海阔
*/
/*
此战 我战与我！
*/
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int mxn=2505;
vector<int>g[mxn],ng[mxn];
ll n,m,k;
ll a[mxn];
int dist[mxn][mxn];
inline void bfs(int x){
    queue<int>q;for(;;q.size();q.pop());
    q.push(x);memset(dist[x],63,sizeof(dist[x]));dist[x][x]=0;
    for(;;q.size();){
        int u=q.front();q.pop();
        for(int v:g[u])if(dist[x][v]>dist[x][u]+1){
            dist[x][v]=dist[x][u]+1;
            q.push(v);
        }
    }
}
int col[mxn];
ll dp[mxn],ans;
inline ll dfs(int x,int d){
    if(dp[x]!=-1)return dp[x];
    dp[x]=-5000000000000000000ll;
    if(d==4){
        if(dist[1][x]<=k+1)dp[x]=a[x];
        else dp[x]=-5000000000000000000ll;
        return dp[x];
    }
    for(int y:ng[x])
        if(col[y]==d+1)
            dp[x]=max(dp[x],dfs(y,d+1)+a[x]);
    return dp[x];
}
```



应用 >>



题库



题单



比赛



记录



讨论

```

}
int main() {
    ios_base::sync_with_stdio(false);
    srand(1145141);
    cin>>n>>m>>k;
    for(int i=2;i<=n;++i)cin>>a[i];
    for(int i=1,u,v;i<=m;++i){
        cin>>u>>v;
        g[u].push_back(v);
        g[v].push_back(u);
    }
    for(int i=1;i<=n;++i)bfs(i);
    for(int i=1;i<=n;++i)for(int j=i+1;j<=n;++j)if(dist[i][j]<=k+1){
        ng[i].push_back(j);
        ng[j].push_back(i);
    }
    for(int ee=0;ee<200;++ee){
        if(rand()%3==1)rand();
        for(int i=2;i<=n;++i)col[i]=rand()%4+1;
        memset(dp,-1,sizeof(dp));
        dfs(1,0);
        ans=max(ans,dp[1]);
    }
    cout<<ans<<endl;
    return 0;
}

```

upd: 这份没有卡时的代码在官方数据下获得了整整65pts的好成绩, 但我们可以通过一个clock()来进行卡时, 获得100pts的好成绩。

Code:

```

#include<bits/stdc++.h>
#define ll long long
using namespace std;
const int mxn=2505;
vector<int>g[mxn],ng[mxn];
ll n,m,k;
ll a[mxn];
int dist[mxn][mxn];
inline void bfs(int x){
    queue<int>q;for(;;q.size();q.pop());
    q.push(x);memset(dist[x],63,sizeof(dist[x]));dist[x][x]=0;
    for(;;q.size();){
        int u=q.front();q.pop();
        for(int v:g[u])if(dist[x][v]>dist[x][u]+1){
            dist[x][v]=dist[x][u]+1;
            q.push(v);
        }
    }
}
int col[mxn];
ll dp[mxn],ans;
inline ll dfs(int x,int d){
    if(dp[x]!=-1)return dp[x];
    dp[x]=-5000000000000000000ll;
    if(d==4){
        if(dist[1][x]<=k+1)dp[x]=a[x];
        else dp[x]=-5000000000000000000ll;
        return dp[x];
    }
    for(int y:ng[x])

```



应用 >>



题库



题单



比赛



记录



讨论

```

        if (col[y] == d + 1)
            dp[x] = max(dp[x], dfs(y, d + 1) + a[x]);
        return dp[x];
    }

    int main() {
        clock_t st = clock();
        ios_base::sync_with_stdio(false);
        srand(1919810);
        cin >> n >> m >> k;
        for (int i = 2; i <= n; ++i) cin >> a[i];
        for (int i = 1, u, v; i <= m; ++i) {
            cin >> u >> v;
            g[u].push_back(v);
            g[v].push_back(u);
        }
        for (int i = 1; i <= n; ++i) bfs(i);
        for (int i = 1; i <= n; ++i) for (int j = i + 1; j <= n; ++j) if (dist[i][j] <= k + 1) {
            ng[i].push_back(j);
            ng[j].push_back(i);
        }
        for (int ee = 0; ee < 10000; ++ee) {
            if (clock() - st > 1.98 * CLOCKS_PER_SEC) break;
            if (rand() % 3 == 1) rand();
            for (int i = 2; i <= n; ++i) col[i] = rand() % 4 + 1;
            memset(dp, -1, sizeof(dp));
            dfs(1, 0);
            ans = max(ans, dp[1]);
        }
        cout << ans << endl;
        return 0;
    }

```

👍 86



💬 30 条评论

收起 ^



StayAlone



创建时间: 2022-10-31 16:40:52

[在 Ta 的博客查看](#)

考场上磕了一个多钟，并没有做出来，但我仍然认为是绿题。

属于是今年 CSP 的一大遗憾了。

题意

给定一个 $n \leq 2500, m \leq 10000$ 的无向图，有点权。求一条点权和最大的路径 $1 \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow 1$ ，满足：

- A, B, C, D 均不为 1，且互不相同；
- 每一段路径上经过的点的数量小于等于 k 。

思路

如果想到枚举 B, C ，此题基本上就没有难度了。

先 n^2 bfs 预处理全源最短路。

定义“一步可以走到”指这两个点之间经过的点小于等于 k 。

枚举了 B, C ，容易想到， A 可以一步走到 1，也可以一步走到 B ；同理， D 可以一步走到 1，也可以一步走到 C 。

因此，对于每个点 v ，预处理出所有点 u ，其中 u 能一步走到 1 和 v ，注意 $u \neq 1 \wedge u \neq v$ 。因为我们只需要求出答案的最大值，而 u 可能和 C, D 重合，所以拿一个 set 维护点权前 3 大即可。

下面暴力枚举满足条件的 B, C ，并从预处理出的 set 里面取 A, D ，若互不相同就更新答案。时间复杂度 $O(n^2 s^2 \log s)$ ，其中 $s = 3$ 。而且极其跑不满啊。

接下来暴力枚举满足条件的 B, C ，并从预处理出的 set 里面取 A, D ，若互不相同就更新答案。时间复杂度 $O(n^2 s^2 \log s)$ ，其中 $s = 3$ 。而且极其跑不满啊。

记得开上点权的 long long 。

```
#include <bits/stdc++.h>
#define repl(i, l, r) for (int i = l; i <= int(r); ++i)
#define rep2(i, l, r) for (int i = l; i >= int(r); --i)
#define fst first
#define snd second
#define mp make_pair
#define eb emplace_back
#define ptc putchar
using namespace std;
typedef long long ll;
typedef pair<ll, ll> pll;
const int MAXN = 2.5e3 + 10, INF = ~0U >> 1, inf = ~0U >> 2;
namespace stupid_lrc {

};
using namespace stupid_lrc;
int n, m, k, d[MAXN][MAXN]; ll w[MAXN], ans;
bool vis[MAXN]; vector<int> lnk[MAXN];
set<pll> bs[MAXN]; // 1 与 i 都能一步到达的点

il void solve(int s, int *d) {
    queue<int> q; memset(vis, 0, sizeof vis);
    repl(i, 1, n) d[i] = INF;
    d[s] = 0; q.push(s);
    while (q.size()) {
        int now = q.front(); q.pop();
        if (vis[now]) continue;
        vis[now] = true;
        for (auto v : lnk[now]) if (d[v] > d[now] + 1) d[v] = d[now] + 1, q.push(v);
    }
}

int main() {
    read(n, m, k); repl(i, 2, n) read(w[i]);
    repl(i, 1, m) {
        int u, v; read(u, v);
        lnk[u].eb(v); lnk[v].eb(u);
    }
    repl(i, 1, n) solve(i, d[i]);
    repl(i, 2, n) repl(j, 2, n) {
        if (j == i) continue;
        if (d[i][j] <= k + 1 && d[1][j] <= k + 1) bs[i].insert(mp(w[j], j));
        if (bs[i].size() > 3) bs[i].erase(bs[i].begin());
    }
    repl(b, 2, n) repl(c, 2, n) {
        if (d[b][c] > k + 1 || b == c) continue;
        for (auto a : bs[b]) {
            if (a.snd == b || a.snd == c) continue;
            for (auto d : bs[c]) {
                if (d.snd == b || d.snd == c || d.snd == a.snd) continue;
                ans = max(ans, w[b] + w[c] + w[a.snd] + w[d.snd]);
            }
        }
    }
    cout << ans << endl;
    return 0;
}
```




最大点 550ms 以内。

在这个结尾，不要脸地放上我 CSP2022 的思考与汗泪。

👍 51

💬 11 条评论

收起 ^

Dregen_Yor

🏆

创建时间：2022-11-01 17:16:33

在 Ta 的博客查看

更好的阅读体验。

思路

我们考虑以每个点为起点跑一遍 dijkstra，以 1 为边权，只记录所有最短距离小于等于 k 的点，并重重新建一张图，在能经过至多 k 次中转后到达的点之间连一条边。因为 $5 \leq n \leq 2500$ ，所以我们可以用一个二维数组来储存两个点之间是否能相互到达。

这样我们可以通过枚举 4 个不同的点，判断他们是否连通且起始位置与结束位置与 1 号点是否连通即可。但是复杂度为 $\mathcal{O}(n^4)$ ，这样的复杂度在本题的条件下显然是不行的，我们考虑如何进行优化。

我们用一个**二元组** (i, j) 来表示从 1 号点出发到 i 号点再到 j 号点得到的分数（即 $a_i + a_j$ ）。

对于每个确定的 j ，我们处理出**前三大的可行的旅行计划**，分别表示为 $(i_1, j), (i_2, j), (i_3, j)$ 。

我们考虑如何将 2 个二元组合并。

我们枚举可以在 k 次转车内到达的两个点 u, v 。分别考虑两者中的 $(i_1, u), (i_2, u), (i_3, u)$ 以及 $(j_1, v), (j_2, v), (j_3, v)$ 进行合并，合并时注意判断 4 个点是否全部不同。

为什么是前 3 大的点

因为要从所有点中选取 4 个不同的点，我们将这 4 个不同的点分别记为 i, j, u, v ，假设 j, u, v 三个点是确定的，并且 u, v 在一个二元组中。

我们已经记录了 j 的前三大的可行的旅行计划，即 $(i_1, j), (i_2, j), (i_3, j)$ ，若 i 中存在与 u, v 重复的点，则最多有 2 个重复的，第三个可以保证 4 个点不重复。至于后面的点，无法对答案产生贡献，在这里直接忽略不计即可。

代码

```
#include<bits/stdc++.h>
#include<cstdio>
#define int long long
using namespace std;
int n,m,k,a[3000],dis[2510][2510];
vector<int> G[3000],edg[2500];
struct nod{
    int dis,num;
    bool operator <(const nod &b)const{
        return dis>b.dis;
    }
};
inline int read(){
    int x=0;
    char ch=getchar();
    while(ch<'0' || ch>'9'){
        ch=getchar();
    }
    while(ch>='0' && ch<='9'){
        x=x*10+ch-'0';
        ch=getchar();
    }
    return x;
}
```



应用 >>



题库



题单



比赛



记录



讨论

```

while(ch>='0' &&ch<='9') {
    x=(x<<3)+(x<<1)+(ch^48);
    ch=getchar();
}
return x;
}

void write(int x){
    if(x>9){
        write(x/10);
    }
    putchar('0'+x%10);
}

void dijkstra(int st){
    for(int i=1;i<=n;i++){
        dis[st][i]=1e9;
    }
    priority_queue<nod> q;
    q.push(nod{0,st});
    while(!q.empty()){
        nod head=q.top();
        q.pop();
        if(dis[st][head.num]<head.dis){
            continue;
        }
        if(st!=head.num){
            edg[st].push_back(head.num);
        }
        dis[st][head.num]=head.dis;
        for(auto to:G[head.num]){
            if(dis[st][to]>head.dis+1&&head.dis+1<=k+1){
                dis[st][to]=head.dis+1;
                q.push(nod{dis[st][to],to});
            }
        }
    }
}

bool cmp(int A,int B){
    return a[A]>a[B];
}

int dist[2510][4],ans,tmp[2510];
void getans(){
    for(int i=2;i<=n;i++){
        int tot=0;
        for(int j=1;j<=n;j++){
            tmp[j]=0;
        }
        for(auto j:edg[i]){
            if(j==1||j==i){
                continue;
            }
            if(dis[1][j]<1e9){
                tmp[++tot]=j;
            }
        }
        sort(tmp+1,tmp+1+tot,cmp);
        dist[i][1]=tmp[1],dist[i][2]=tmp[2],dist[i][3]=tmp[3];
    }
    for(int i=2;i<=n;i++){
        for(auto j:edg[i]){
            if(j==1){
                continue;
            }
            for(int x=1;x<=3;x++){
                for(int y=1;y<=3;y++){

```



应用 >>



题库



题单



比赛



记录



讨论

```

        if(dist[i][x]!=j&&dist[i][x]!=dist[j][y]&&dist[j][y]!=i&&i!=j&&dist[i][x]&&dist[j][y]){
            ans=max(ans,a[i]+a[j]+a[dist[i][x]]+a[dist[j][y]]);
        }
    }
}

signed main(){
    n=read(),m=read(),k=read();
    for(int i=2;i<=n;i++){
        a[i]=read();
    }

    for(int i=1;i<=m;i++){
        int u,v;
        u=read(),v=read();
        G[u].push_back(v);
        G[v].push_back(u);
    }

    for(int i=1;i<=n;i++){
        dijkstra(i);
    }

    getans();
    write(ans);
    return 0;
}

```

👍 22



💬 1 条评论

收起 ^



Leasier



创建时间: 2022-10-30 18:51:40

[在 Ta 的博客查看](#)

首先 bfs 预处理出任意两点间的距离。

然后就可以写出一个 $O(nm + n^4)$ 暴力了：直接枚举 A, B, C, D 并判断是否合法。

此时注意到 $n \leq 2.5 \times 10^3$ ，也就是说我们**至多可以枚举两个点**。

于是我们折半一下，预处理对于每个 B ，所有 A （即 $1 \rightarrow A \rightarrow B$ ）的最大、次大、次次大的值，求值时枚举 B, C 并统计即可。

时间复杂度为 $O(n(n + m))$ 。

代码：

```

#include <iostream>
#include <queue>

using namespace std;

typedef long long ll;

typedef struct {
    int nxt;
    int end;
} Edge;

int cnt = 0;
int dis[2507][2507], head[2507], pos[2507][7];
ll s[2507], f[2507][2507];
Edge edge[20007];
queue<int> q;

```



```

queue<int> q;

inline void init(int n){
    for (register int i = 1; i <= n; i++){
        for (register int j = 1; j <= n; j++){
            dis[i][j] = 0x7fffffff;
        }
    }
}

inline void add_edge(int start, int end){
    cnt++;
    edge[cnt].nxt = head[start];
    head[start] = cnt;
    edge[cnt].end = end;
}

void bfs(int start, int n){
    dis[start][start] = 0;
    q.push(start);
    while (!q.empty()){
        int cur = q.front();
        q.pop();
        for (register int i = head[cur]; i != 0; i = edge[i].nxt){
            int x = edge[i].end;
            if (dis[start][x] == 0x7fffffff){
                dis[start][x] = dis[start][cur] + 1;
                q.push(x);
            }
        }
    }
}

int main(){
    int n, m, k;
    ll ans = 0;
    cin >> n >> m >> k;
    k++;
    init(n);
    for (register int i = 2; i <= n; i++){
        cin >> s[i];
    }
    for (register int i = 1; i <= m; i++){
        int x, y;
        cin >> x >> y;
        add_edge(x, y);
        add_edge(y, x);
    }
    for (register int i = 1; i <= n; i++){
        bfs(i, n);
    }
    for (register int i = 1; i <= n; i++){
        for (register int j = 1; j <= n; j++){
            if (i != j && dis[1][i] <= k && dis[i][j] <= k){
                f[i][j] = s[i] + s[j];
            } else {
                f[i][j] = -4e18;
            }
        }
    }
    for (register int i = 2; i <= n; i++){
        pos[i][1] = pos[i][2] = pos[i][3] = i;
        for (register int j = 2; j <= n; j++){
            if (f[pos[i][1]][i] < f[j][i]){

```

应用 >>

题库

题单

比赛


记录

讨论

```
pos[i][3] = pos[i][2];
pos[i][2] = pos[i][1];
pos[i][1] = j;
} else if (f[pos[i][2]][i] < f[j][i]){
    pos[i][3] = pos[i][2];
    pos[i][2] = j;
} else if (f[pos[i][3]][i] < f[j][i]){
    pos[i][3] = j;
}
}
}
for (register int i = 2; i <= n; i++){
    for (register int j = 2; j <= n; j++){
        if (i != j && dis[i][j] <= k){
            for (register int x = 1; x <= 3; x++){
                for (register int y = 1; y <= 3; y++){
                    if (pos[i][x] != j && pos[i][x] != pos[j][y] && pos[j][y] != i){
                        ans = max(ans, f[pos[i][x]][i] + f[pos[j][y]][j]);
                        break;
                    }
                }
            }
        }
    }
}
cout << ans;
return 0;
}
```

👍 19 💬 6 条评论

收起 ^

 **cyffff**  创建时间：2022-10-31 13:45:29 在 Ta 的博客查看

Link

Update2022.11.8：我的做法居然被官方数据卡了 5 分，发现一个小锅，修掉了。

To 审核：在第一次审核中，本题解因“非本题真正意义上的正解”而被拒绝，事实上，本题解第二种做法拥有正确性。本次提交还增加了正解的说明。

题意

给你一个图， n 个点， m 条边，两个点之间通过一次操作可达当且仅当图上存在一条不算端点，长度 $\leq k$ 的路径。

每个点有点权 s_i ，你需要选出四个**互不相同**的点 a, b, c, d 使得相邻两个点可达且 1 与 a, d 都可达。

$n \leq 2500, m \leq 10^4$ ，时限 $2s$ 。

思路

正解怎么是折半，没想到啊。。

会卡乱搞首先要会写乱搞，所以我来提供一些乱搞做法。

记录前 L 优路径

这个是我赛时做法。

首先对每个点 bfs 一遍找出所有可达的点对。



考虑维护 $dp_{i,j}$ 表示第 i 个点选 j 的答案的最大值。但是这样子没法记录互不重复的。

考虑记录路径，但是走最优路径可能无法转移，同理，只记录前 2 个等不太行。

我们考虑记录前 L 优的路径和答案。转移的时候取出对方记录的路径中不含 j 的方案，与目前的答案归并即可。

写完测一测，取 $L = 500$ ，跑 $n \leq 300$ 很快，但是跑 $n \leq 2500$ 时很慢。考虑数据分治，当 $n > 2500$ 时取 $L = 5$ ，实测能跑。实现的时候需要分开开数组，否则赋值一次过慢。

赛后实测民间数据取 $L = 3$ 都能过，造得不是很强。。

时间复杂度 $O(nm + n^2Lc^2)$ ，其中 $c = 3$ ，为需要选的点数。常数挺大的。

随机转移顺序

这个是 Ntokisq 赛时做法。

首先对每个点 bfs 一遍找出所有可达的点对。

考虑随机一个排列 p_i ，记 $dp_{i,j}$ 表示第 i 个点选 j 的答案，且前 $i - 1$ 个点都必须选 $p_k < p_j$ 的点 k 的情况下答案的最大值。

考虑随机 L 次，每次进行 dp，答案取最大值即可。

分析一下正确性：假设没取到答案，设答案为 a, b, c, d ，则 L 次中每次都不能出现 $p_a < p_b < p_c < p_d$ 或 $p_a > p_b > p_c > p_d$ ，概率为 $\left(\frac{11}{12}\right)^L$ ，则正确率为 $1 - \left(\frac{11}{12}\right)^L$ ，取 $L = 200$ ，则错误率为 2.7×10^{-8} ，可以接受。

时间复杂度 $O(nm + n^2Lc)$ ，其中 $c = 4$ ，为选的点数。

做法 1 看起来没有正确性保障（有没有神仙能证一下或者求出保证能取到答案的最小 L 或者给出较小 L 的 hack 数据），做法 2 是极高概率正确。

正解

看起来不写这个过不了审核啊。

考虑第一二个点和第三四个点的地位相同，只枚举一半。

处理出 $f_{i,1/2/3}$ 表示枚举的第二个点选 i ，第 1/2/3 大的答案和相应的转移位置。处理前三大是因为可能 c, d 会与 a, b 中的任意一到两个重合。

枚举 b, c ，直接计算，时间复杂度 $O(nm + n^2)$ 。

做法 1 的代码（也是我的考场代码，修正了一个小锅）：

```
#include<bits/stdc++.h>
using namespace std;
//cyffff
//[70, 100]+100+[60, 100-]+36=[266, 336-]
#define ll long long

//read(), write()

const int K=500+10, N=2500+10, M=1e4+10;
int n, m, k, bx, dis[N], cnt, head[N], L;
ll v[N], ans;
vector<int>lk[N];
struct Edge{
    int to, nxt;
    ll v[M/4+1];
```



应用 >>



题库



题单



比赛



记录



讨论

```

    fa[m][l],
    struct node{
        int s,a[5];
        ll val;
        inline void clear(){
            s=val=0;
        }
        inline friend bool operator<(const node &a,const node &b){
            return a.val>b.val;
        }
        inline void pushback(int p){
            a[++s]=p,val+=v[p];
        }
        inline bool check(int p){
            for(int i=1;i<=s;i++)
                if(a[i]==p) return 0;
            return 1;
        }
        inline void print(){
            printf("%d %d\n",s,val);
            for(int i=1;i<=s;i++)
                printf("%d ",a[i]);puts("");
        }
    }st[K*2];
    struct DPT{
        node t[510];
        int sz;
        inline void clear(){
            sz=0;
        }
        inline void pushback(node tmp){
            t[++sz]=tmp;
        }
        inline void ad(int x){
            for(int i=1;i<=sz;i++)
                t[i].pushback(x);
        }
        inline void print(){
            for(int i=1;i<=sz;i++)
                t[i].print();
            puts("----");
        }
        inline friend DPT operator+(DPT a,DPT b){
            int top=0;
            int i=1,j=1;
            while(i<=a.sz&&j<=b.sz){
                if(!b.t[j].check(b.x)){
                    j++;continue;
                }
                if(a.t[i]<b.t[j]) st[++top]=a.t[i],i++;
                else st[++top]=b.t[j],j++;
            }
            while(i<=a.sz) st[++top]=a.t[i],i++;
            while(j<=b.sz){
                if(!b.t[j].check(b.x)) j++;
                else st[++top]=b.t[j],j++;
            }
            DPT C;
            C.sz=min(top,L);
            for(int i=1;i<=C.sz;i++)
                C.t[i]=st[i];
            return C;
        }
    }dp[2][310];

```



```

struct DPT2{
    node t[15];
    int sz;
    inline void clear(){
        sz=0;
    }
    inline void pushback(node tmp){
        t[++sz]=tmp;
    }
    inline void ad(int x){
        for(int i=1;i<=sz;i++)
            t[i].pushback(x);
    }
    inline void print(){
        for(int i=1;i<=sz;i++)
            t[i].print();
        puts("----");
    }
    inline friend DPT2 operator+(DPT2 a,DPT2 b){
        int top=0;
        int i=1,j=1;
        while(i<=a.sz&&j<=b.sz){
            if(!b.t[j].check(b.x)){
                j++;continue;
            }
            if(a.t[i]<b.t[j]) st[++top]=a.t[i],i++;
            else st[++top]=b.t[j],j++;
        }
        while(i<=a.sz) st[++top]=a.t[i],i++;
        while(j<=b.sz){
            if(!b.t[j].check(b.x)) j++;
            else st[++top]=b.t[j],j++;
        }
        DPT2 C;
        C.sz=min(top,L);
        for(int i=1;i<=C.sz;i++)
            C.t[i]=st[i];
        return C;
    }
}dp2[2][N];
inline void add(int u,int v){
    cnt++;
    a[cnt].to=v;
    a[cnt].nxt=head[u];
    head[u]=cnt;
}
inline void bfs(int x){
    for(int i=1;i<=n;i++)
        dis[i]=2e9;
    queue<int>q;
    q.push(x);
    dis[x]=0;
    while(!q.empty()){
        int x=q.front();
        q.pop();
        for(int i=head[x];i;i=a[i].nxt){
            int t=a[i].to;
            if(dis[t]>dis[x]+1){
                dis[t]=dis[x]+1;
                if(dis[t]<=k)
                    q.push(t);
            }
        }
    }
}

```




应用 >>



题库



题单



比赛



记录



讨论

```

for(int i=1;i<=n;i++)
    if(dis[i]<=k&&v[i]!=x)
        lk[x].push_back(i);
}
int main() {
    // freopen("holiday.in","r",stdin);
    // freopen("holiday.out","w",stdout);
    n=read(),m=read(),k=read()+1;
    if(n<=300) L=500;
    else L=5;
    for(int i=2;i<=n;i++)
        v[i]=read();
    for(int i=1;i<=m;i++){
        int u=read(),v=read();
        add(u,v),add(v,u);
    }
    for(int i=1;i<=n;i++)
        bfs(i);
    if(n<=300){
        for(auto x:lk[1]){
            node tmp;
            tmp.clear();
            tmp.pushback(x);
            dp[1][x].pushback(tmp);
        }
        for(int t=2;t<=4;t++){
            int c=t&1;
            for(int i=2;i<=n;i++){
                dp[c][i].clear();
                bx=i;
                for(auto j:lk[i])
                    dp[c][i]=dp[c][i]+dp[!c][j];
                dp[c][i].ad(i);
            }
        }
        for(auto x:lk[1])
            if(dp[0][x].sz)//小锅: sz=0 时不需访问第一位
                ans=max(ans,dp[0][x].t[1].val);
    }else{
        for(auto x:lk[1]){
            node tmp;
            tmp.clear();
            tmp.pushback(x);
            dp2[1][x].pushback(tmp);
        }
        for(int t=2;t<=4;t++){
            int c=t&1;
            for(int i=2;i<=n;i++){
                dp2[c][i].clear();
                bx=i;
                for(auto j:lk[i])
                    dp2[c][i]=dp2[c][i]+dp2[!c][j];
                dp2[c][i].ad(i);
            }
        }
        for(auto x:lk[1])
            if(dp2[0][x].sz)//同上
                ans=max(ans,dp2[0][x].t[1].val);
    }
    write(ans);
}

```

做法 2 的代码 (Ntokisq 考场代码, 事实上这份代码不是稳过, 需要改成运行 $T = 80$ 次或者卡时 $1.8s$ 之类

的) :

```

#include<bits/stdc++.h>
#define Yukinoshita namespace
#define Yukino std
#define ll long long
using Yukinoshita Yukino;
vector<int> a[2505], nd[2505];
bool t[2505];
ll v[2505];
int n, cnt;
int p[2505];
ll dp[2505][5], mx;
void init(int d, int k)
{
    queue<pair<int, int> > q;
    q.push({d, 0});
    memset(t, 0, n+1), t[d]=1;
    while(q.size())
    {
        pair<int, int> x=q.front();
        q.pop();
        if(x.first!=d)
            nd[d].push_back(x.first);
        if(x.second==k) continue;
        for(int i=0; i<a[x.first].size(); i++)
            if(!t[a[x.first][i]])
                t[a[x.first][i]]=1, q.push({a[x.first][i], x.second+1});
    }
}
void solve()
{
    int i, j, k, x;
    for(i=1; i<=n; i++)
        memset(dp[i], 128, sizeof(dp[i]));
    cnt+=n*4;
    dp[1][0]=0;
    for(i=2; i<=n; i++)
        for(x=p[i], j=1; j<=4; j++)
        {
            for(k=0; k<nd[x].size(); k++)
                dp[x][j]=max(dp[x][j], dp[nd[x][k]][j-1]);
            dp[x][j]+=v[x];
            cnt+=nd[x].size();
        }
    for(i=0; i<nd[1].size(); i++)
        mx=max(mx, dp[nd[1][i]][4]);
}
int main()
{
    srand(time(0));
    // freopen("holiday.in", "r", stdin);
    // freopen("holiday.out", "w", stdout);
    int m, k, x, y, i, j;
    scanf("%d%d%d", &n, &m, &k);
    for(i=2; i<=n; i++)
        scanf("%lld", &v[i]);
    while(m--)
        scanf("%d%d", &x, &y),
        a[x].push_back(y),
        a[y].push_back(x);
    for(i=1; i<=n; i++)
        init(i, k+1);
}

```



```
for (i=1; i<=n; i++)
    p[i]=i;
solve();
reverse(p+2, p+1+n);
solve();
while (cnt<1e8*1.5)
    random_shuffle(p+2, p+1+n), solve();
cout<<mx;
}
```

没有看出来正解，还是水平不太足，当然，乱搞能力也是一种能力。

再见 qwq~

👍 20 💬 7 条评论

收起 ^



happy_dengziyue 创建时间：2022-11-05 22:47:00

[在 Ta 的博客查看](#)

1 视频题解

2 思路

首先 n 次广搜搞定最短路。

我们可以发现，因为这张图是无向图，所以 $X \rightarrow Y$ 和 $Y \rightarrow X$ 意义几乎一样。

然后预处理：

枚举，从 1 到 X 点存在什么**中途恰好去 1 个景区游玩**的方案，记录下**前 3 优**的满足距离要求的方案。

题目要求我们 $1 \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow 1$ ，那么我们就枚举 B 和 C 。

至于 A 和 D ？我们可以发现，如果 $1 \rightarrow X \rightarrow B$ 和 $C \rightarrow X \rightarrow 1$ 都分别有至少 3 种满足距离要求的方案时，枚举前 3 优解，产生的 9 种可能中，肯定有一种是四个景点互不相同的。其他的次优解根本没有意义。

通过枚举即可枚举出 A 和 D 。不断更新答案即可。

3 代码



本代码已通过洛谷测试和 InfoJ 测试。

应用 >>



题库



题单



比赛



记录



讨论

```

#include<stdio>
#include<cstring>
#include<algorithm>
#include<queue>
using namespace std;
#define max_n 2500
#define max_m 20000
#define inf 0x3f3f3f3f
int n;
int m;
int k;
long long a[max_n+2];
struct E{
    int v,nx;
}e[20002];
int ei=0;
int fir[max_n+2];
int dis[max_n+2][max_n+2];
struct W{
    int u;
    bool operator<(const W&x)const{
        return a[x.u]>a[u];
    }
};
int c[max_n+2][6];
int ci[max_n+2];
long long ans=0;
void addedge(int u,int v){
    e[++ei]=(E){v,fir[u]}; fir[u]=ei;
}
void bfs(){
    memset(dis,inf,sizeof(dis));
    queue<int>q;
    for(int f=1;f<=n;++f){
        while(!q.empty())q.pop();
        dis[f][f]=0;
        q.push(f);
        while(!q.empty()){
            int u=q.front();
            q.pop();
            for(int i=fir[u],v;i=i=e[i].nx){
                v=e[i].v;
                if(dis[f][u]+1<dis[f][v]){
                    dis[f][v]=dis[f][u]+1;
                    if(dis[f][v]>=k)continue;//不需要继续了
                    q.push(v);
                }
            }
        }
    }
}
int main(){
    scanf("%d%d%d",&n,&m,&k);
    ++k;
    for(int i=2;i<=n;++i)scanf("%lld",a+i);
    for(int i=1,u,v;i<=m;++i){
        scanf("%d%d",&u,&v);
        addedge(u,v);
        addedge(v,u);
    }
    bfs();
}

```



```

memset(ci, 0, sizeof(ci));
for(int i=2; i<=n; ++i) {
    priority_queue<W> q;
    for(int u=2; u<=n; ++u) {
        if(u==i) continue;
        if(dis[1][u]<=k&&dis[u][i]<=k) q.push((W){u});
    }
    while(!q.empty()&&ci[i]<3) {
        c[i][++ci[i]]=q.top().u;
        q.pop();
    }
}
for(int i=2; i<=n; ++i) {
    for(int j=i+1; j<=n; ++j) {
        if(dis[i][j]>k) continue;
        for(int ii=1; ii<=ci[i]; ++ii) {
            for(int jj=1; jj<=ci[j]; ++jj) {
                int u=c[i][ii], v=c[j][jj];
                if(u!=j&&v!=i&&u!=v) ans=max(ans, a[u]+a[i]+a[j]+a[v]); //1->u->i->j->v->1
            }
        }
    }
}
printf("%lld\n", ans);
}

```

4 暴力枚举

上述解法并不是我的考场解法。

我的考场上的暴力枚举加上玄学优化的方法，在洛谷和 InfOJ 分别能拿 100 分和 90 分。

By dengziyue

👍 19 💬 1 条评论

收起 ^



ReKoj 创建时间: 2022-10-30 16:48:43

在 Ta 的博客查看

link

$1 - A - B - C - D - 1$ 非常对称，我们断开来，分成 $1 - A - B$ 和 $C - D - 1$ 两部分，不难发现这两块是完全一致的。

首先对于每个景点 x 求出距离它不过 K 、且距离 1 不超过 K 的所有点（即对于每个 B ，找到满足条件的 A ），设这些点形成的集合为 S_B 。由于边权为 1，对于每个 B ，我们只需要 bfs 即可 $O(n + m)$ 得到 S_B ，整个过程的复杂度为 $O(n(n + m))$ 。

接下来，枚举 $f_{B,C} \leq K$ 的 (B, C) ，然后暴力枚举 S_B, S_C 中的点，取最优且满足限制的组合即可。

上面求答案的复杂度是 $O(n^4)$ 的，需要优化。

不难贪心想到我们只需要取 S_B, S_C 中最大的值，但是我们可能会从 S_B 取到 C ， S_C 取到 B ，或者从 S_B, S_C 中取到相同的值，我们需要规避掉这种情况。

考虑先满足从 S_B 取出来的值与 C 不同，那么我们顺便维护 S_B 中的次大值，若最大值与 C 相同就取次大值，否则取最大值即可。

然后考虑 S_C 取出的值满足条件，不难发现需要不同的值有 2 个（ B, S_B 取出的值），那么我们还要多维护次次大值，这样就可以保证最优解必然被我们枚举到了。

总复杂度 $O(nm + n^2)$ 。



code

👍 16



💬 0 条评论

收起 ^



yzy1

创建时间: 2022-10-31 09:30:39

在 Ta 的博客查看

题目大意

给定 n ($5 \leq n \leq 2500$) 个点 m ($1 \leq m \leq 10^4$) 条边的无向简单图, 结点依次编号为 $1 \sim n$ 的整数. 编号为 i 的结点有 a_i ($1 \leq a_i \leq 10^{18}$) 的整数点权. 定义一个五元组 $(c_1, c_2, c_3, c_4, c_5)$ 是「符合条件的」的, 当且仅当:

- $c_1 = 1$.
- c_1, c_2, c_3, c_4, c_5 互不相等.
- \forall 整数 $i \in [1, 5]$, 均有 $\text{dis}(c_i, c_{i \bmod 5 + 1}) \leq K$. 其中 $\text{dis}(i, j)$ 表示连接结点 i 与 j 的路径中包含边数最少的路径所包含的边数, K 为一给定常量.

定义一个五元组的权值为五个点的点权之和, 求出权值最大且符合条件的五元组的权值, 保证存在至少一组符合条件的五元组.

简要做法

首先以每个点为开始跑 BFS, 求出任意两个点之间的最短路.

考虑在固定 c_3, c_4 两个点的条件下找出权值最大的 c_2 和 c_5 . 发现 c_2 与 $1, c_3$ 的距离, c_4 与 $1, c_5$ 的距离不能超过 K . 考虑对于每个结点, 预处理出一个集合代表该点作为 c_3 (或 c_4) 时, 有哪些点可以作为 c_2 (或 c_5). 然后枚举预处理出的集合, 依次检查选出的 c_2 和 c_5 是否与已有的点重复, 并更新答案.

上述做法时间复杂度为 $O(nm + n^4)$. 无法通过本题. 需要进一步优化.

我们发现处理 c_2 与 c_5 的时候, 没有必要依次检查集合中的所有元素. 根据鸽巢原理, 不妨设当前处理的是 c_2 , 则该点只可能会和 c_3, c_4, c_5 重复. 因此我们可以只保留集合中权值最大的四个不同的结点, 则一定存在一个结点不与这三个结点重复. 处理 c_5 时的情况同理. 由于对于每对 c_3, c_4 只需枚举常数对 c_2, c_5 , 时间复杂度为 $O(nm + n^2)$. 可以通过本题.

👍 8



💬 0 条评论

收起 ^



yukimianyan

创建时间: 2022-10-29 23:32:15

在 Ta 的博客查看

problem

n 个点 m 条边的无向无权图, 令 $to(i, j) = [\text{dist}(i, j) \leq k + 1]$, 点带权 a_i , 求:

$$[*] = \max_{A \neq B \neq C \neq D \neq 1} to(1, A) \cdot to(A, B) \cdot to(B, C) \cdot to(C, D) \cdot to(D, 1) \cdot (a_A + a_B + a_C + a_D).$$

$n, m \leq 3000$.

solution

以每个点为源点跑 bfs, 求出 $to(i, j)$, 复杂度 $O(nm)$.

我们弱化一下这个问题: 如果只有两个景点 A, B , 怎么做?

我们可以预处理一个 $F_{i,j}$ 表示一个权值最大的景点 k 满足 $to(i, k) \wedge to(k, j)$, 我们先不说怎么做. 然后枚举景点 A , 那么我们确信景点 $F_{1,A}$ 是最优的.

现在我们有三个景点 A, B, C , 我仍然可以枚举 B , 转移 $F_{1,i}$ 的时候记录最大值和次大值, 那么我们确信这两个值且 A, C 的景点取值

我们固定 A, C 的最优取值。

现在我们有四个景点 A, B, C, D ，我们可以枚举 B, C ，用 $F_{1,B}, F_{C,1}$ 分别求出 A, D 的最优取值。注意当我们确定 $F_{1,B}$ 时，有两个景点 $(B, F_{1,B})$ 已经被选，所以使用 $F_{C,1}$ 时我们要记录最大值、次大值、次次大值，确保没有重复才能转移。反过来也要判一次。枚举的复杂度是 $O(n^2)$ 的。

现在回到 F ，我们发现，由于图是无向图，因此我们断定 $F_{i,j} = F_{j,i}$ ，这样一来我们所有用到的 F 都形如 $F_{1,i}$ ，状态数 $O(n)$ ，转移暴力 $O(n)$ 就好了。

code

```
#include <queue>
#include <cstdio>
#include <cstring>
#include <algorithm>
using namespace std;
typedef long long LL;
template<int N,int M,class T=int> struct graph{
    int head[N+10],nxt[M*2+10],cnt;
    struct edge{
        int u,v; T w;
        edge(int u=0,int v=0,T w=0):u(u),v(v),w(w){}
    } e[M*2+10];
    graph(){memset(head,cnt=0,sizeof head);}
    edge& operator[] (int i){return e[i];}
    void add(int u,int v,T w=0){e[++cnt]=edge(u,v,w),nxt[cnt]=head[u],head[u]=cnt;}
    void link(int u,int v,T w=0){add(u,v,w),add(v,u,w);}
};
LL a[3010];
bool cmp(int i,int j){return a[i]>=a[j];}
struct node{
    int v[4];
    node(){v[0]=v[1]=v[2]=v[3]=0;}
    void update(int k){
        if(cmp(k,v[0])) v[3]=v[2],v[2]=v[1],v[1]=v[0],v[0]=k;
        else if(cmp(k,v[1])) v[3]=v[2],v[2]=v[1],v[1]=k;
        else if(cmp(k,v[2])) v[3]=v[2],v[2]=k;
        else if(cmp(k,v[3])) v[3]=k;
    }
    int operator[] (int i){return v[i];}
};
int n,m,sshwy,dis[3010];
bool to[3010][3010];
graph<3010,10010> g;
void bfs(int s){
    queue<int> q;
    memset(dis,0x3f,sizeof dis);
    for(q.push(s),dis[s]=0;!q.empty();){
        int u=q.front(); q.pop();
        if(dis[u]>sshwy+1) continue;
        to[s][u]=1;
        if(dis[u]==sshwy+1) continue;
        for(int i=g.head[u];i;i=g.nxt[i]){
            int v=g[i].v;
            if(dis[v]>dis[u]+1) dis[v]=dis[u]+1,q.push(v);
        }
    }
}
node f[3010];
void dp(){
    for(int i=2;i<=n;i++){
        for(int j=2;j<=n;j++){
            if(i==j||!to[1][i]||!to[i][j]) continue;
```



应用 >>



题库



题单



比赛



记录



讨论

```

        f[j].update(i);
    }
}
// for(int i=1;i<=n;i++){
//     fprintf(stderr,"f[%d]={%d,%d,%d,%d}\n",i,f[i][0],f[i][1],f[i][2],f[i][3]);
// }
}
LL solve(){
    LL res=-1;
    for(int u=2;u<=n;u++){
        for(int v=2;v<=n;v++){
            if(!to[u][v]) continue;
            node &be=f[u],&ce=f[v];
            for(int i=0;i<4;i++){
                for(int j=0;j<4;j++){
                    if(!be[i]||!ce[j]) continue;
                    int pos[]={u,v,be[i],ce[j]};
                    if(sort(pos,pos+4),unique(pos,pos+4)==pos+4){
                        res=max(res,a[u]+a[v]+a[be[i]]+a[ce[j]]);
                        break;//后面的决策不会更优
                    }
                }
            }
        }
    }
    //考场并没有发现只需要 3 个值，写的比较暴力
}
return res;
}
int main(){
    a[0]=a[1]=-1e18;
    freopen("holiday.in","r",stdin),freopen("holiday.out","w",stdout);
    scanf("%d%d%d",&n,&m,&sshwy);
    for(int i=2;i<=n;i++) scanf("%lld",&a[i]);
    for(int i=1,u,v;i<=m;i++) scanf("%d%d",&u,&v),g.link(u,v);
    for(int i=1;i<=n;i++) bfs(i);
    dp();
    printf("%lld\n",solve());
    return 0;
}

```



7



0 条评论

收起 ^



在洛谷，
享受 Coding 的快乐



关于洛谷 | 帮助中心 | 用户协议 | 联系我们

小黑屋 | 陶片放逐 | 社区规则 | 招贤纳士

Developed by the Luogu Dev Team

2013-2023, © 洛谷

增值电信业务经营许可证 沪B2-20200477

沪ICP备18008322号 All rights reserved.