NOIP 2016 Day 1 题解

玩具谜题 toy

知识点

模拟

分析

可以发现，本题就是根据要求在环上顺时针或者逆时针走动，那么假设当前的位置是 p，那么逆时针走 x 个到达的位置是 $(p-1+x) \bmod n+1$ ，顺时针走 x 个到达的位置是 $(p-1-x) \bmod n+1$ 。模拟每一次的指令，判断每次是走顺时针还是逆时针即可。

实现上要注意，C++ 中对于模运算的定义与数学中的模运算是不同的，我们知道在数学上 $a \bmod b=a-b \cdot \lfloor \frac{a}{b} \rfloor$ ，问题就在于 $\lfloor \frac{a}{b} \rfloor$ 的计算上。在大部分编译器中，两个 int 相除，是**直接丢弃**小数位得到答案的，这就导致了在 C++ 中两个 int 相除并非总是得到 $\lfloor \frac{a}{b} \rfloor$（例如 $\frac{-3}{2}=-1$），所以 a 为负数的情况下，取模会存在问题，$a \bmod b$ 不一定能够得到一个非负整数，但是我们可以保证这个数在 $(-|b|,|b|)$ 内，在所以实际需要取模的时候，采用 $((a \bmod b)+b) \bmod b$ 来得到非负整数。

代码

```
#include <cstdio>

using namespace std;

const int MAX_N = 100000 + 3, MAX_LEN = 10 + 3;

int n, m, dir[MAX_N];

char str[MAX_N][MAX_LEN];

int main() {

    scanf("%d%d", &n, &m);
```

```
    for (int i = 0; i < n; ++i)
        scanf("%d%s", dir + i, str[i]);
    int pos = 0;
    for (int i = 0, d, a; i < m; ++i) {
        scanf("%d%d", &d, &a);
        int dd = d ^ dir[pos];
        if (dd == 1) (pos += a) %= n;
        else (pos -= a) %= n;
        pos = (pos % n + n) % n;
    }
    printf("%s\n", str[pos]);
    return 0;
}
```

天天爱跑步 running

考点　DFS，LCA（最近公共祖先），差分标记，前缀和

分析

对于 NOIP 初学者来说，这个题在考场上是比较没有办法 A 掉的，但是这并不妨碍我们拿到暴力分。

25 分

对于前 5 个测试点，$n, m \leq 1000$。

可以考虑对于每一个人 (u, v)，都使用 BFS 找出 $u \to v$ 的路径，然后更新路径上有满足条件的点即可。至于找出路径的方法，只需要从 u 开始 BFS，并且在 BFS 中从 a 扩展到 b 时，记录 a 的前继为 b，最后我们从 v 开始，不断地找前继，最终到 u，这样就还原出了路径，你可能还需要记录一个到点 v 的距离来帮助判断路径上的点是否满足条件。

复杂度：$O(n^2)$。

40 分

接下来 3 个测试点，$n, m \leq 10^5$，形成一条链。

本题有 2s 的时限，大概能够承受 10^8 的运算量，如果使用 25 分算法，运算量将达到 10^{10}，绝对无法承受。所以我们不能采用还原路径的方法。

由于链上的点依次是 $1, 2, \ldots, n$，而题目中的路径有从编号小的点走到编号大的点或者从编号大的点走到编号小的点。我们将这两种路径分开考虑，先考虑从编号小的点走到编号大的点 $(u, v)(u \leq v)$。
对于 (u, v)，如果路径上的点 $i(u \leq i \leq v)$ 要满足条件的话，必然是满足 $w_i = i - u$，我们不妨设 $w'_i = w_i - i$，那么现在的条件转化为了 $w'_i = -u$，也就是说，路径上的点 i 答案 + 1 的充分必要条件是满足 w'_i 等于一个定值。

现在问题转化成了支持两种操作：

1. 将一段链上的每个点插入一个数。
2. 查询点 i 上有多少个数是 w'_i。

我们考虑使用差分化标记实现，这个思想在 NOIP 2012 借教室 出现过。
对于将链上的 $[l, r)$ 同时插入一个数 v，差分后转化为在 l 位置插入一个 $(v, 1)$（表示增加一个 v），在 r 位置插入一个 $(v, -1)$（表示删除一个 v）。由于差分的逆运算是前缀和，我们顺序遍历这条链，维护 cnt_v 为数 v 出现的次数（由于 i 有负数，所以我们考虑将这个数组向右平移 n 个单位来方便存储），每遍历到一个点 i，我们就处理它上面所有的标记 (v, t)，将 cnt_v 加上 t。这样遍历到每个点的时候，cnt_v 就是在它上面数 v 的出现次数。

对于反过来的路径 (u, v)，同理可以发现点 $i(v \leq i \leq u)$ 答案 + 1 的充分必要条件是 $w_i = u - i$，设 $w''_i = w - (n - i)$，那么条件转化为 $w''_i = u - n$，也是一个常数，可以用类似的方法处理。

复杂度：$O(n + m)$。这个做法比较重要，对于初学者建议实现这个程序。

树链剖分

刚刚我们得出了链上的算法，树链剖分可以让我们将树剖分成链，这样树上的问题就可以转化为链上的问题，套用链上的算法即可。每条路径产生 $O(\log n)$ 个标记，处理每条路径的复杂度也是 $O(\log n)$，最后结算标记的复杂度 $O(n \log n)$，所以总复杂度 $O(n \log n)$。依据实现的情况，得到 95 或 100 分。

100 分

NOIP 显然不会考树链剖分，这题有更优美的解法。我们首先求出每条路径的 LCA，使用 Tarjan 算法 在 $O(n+m)$ 的时间内预处理出 LCA。对于每一条路径，我们拆成直上直下的两条链（如果本身就是这种链就不拆），设 d_i 为 i 点的深度。我们考虑链 $u \to v$，对于向下的链，路径上满足条件的点 i 必须满足 $d_i - d_u = w_i$，即 $w_i - d_i$ 是一个定值。对于向上的链，有 $d_u - d_i = w_i$，即 $w_i + d_i$ 是一个定值，问题同样转化为了支持两种操作（对于两种链分别考虑，这里只说了第一种链）：

1. 将一段路径（没有弯折）上的每个点插入一个数。
2. 查询点 i 上有多少个数是 w'_i。

如果在路径 $u \to v(d_u \le d_v)$ 上插入一个数 x，可以差分转化为在 v 这个地方插入一个 x，在 fa_u 这个地方删除一个数 x，则点 i 上的数就是其子树的和（一个常用技巧，建议画图验证）。

显然对于每一个点都求一次子树是 $O(n^2)$ 的，考虑如何快速计算。我们可以弄出树的 DFS 序，对于一棵子树，它在 DFS 序中必然是连续的一段。那么对于一个点 i，我们只需要查询它的子树代表的一段连续的区间中有多少个数 w'_i。假设其子树的区间为 $[l,r]$，$T_{j,v}$ 为 $[0,r]$ 中有多少个 v，那么 i 的答案等于 $T_{r,w'_i} - T_{l-1,w'_i}$。我们先算出所有点的 l 和 r，然后离线，顺序遍历 DFS 序列来结算每个前缀和。这一步比较抽象，大体的思想就是每遍历到一个位置 i，结算所有包含 T_i 项的区间，由于只有 $2n$ 个区间，所以结算的复杂度是 $O(n+m)$ 的。

具体的实现见代码，复杂度：$O(n+m)$。

代码

树链剖分

```
//    Created by Sengxian on 2016/11/23.
//    Copyright (c) 2016 年  Sengxian. All rights reserved.
#include <bits/stdc++.h>
using namespace std;

const int MAX_N = 300000 + 3;

struct edge {
    edge *next;
```

```
        int to;
        edge(edge *next = NULL, int to = 0): next(next), to(to) {}
} pool[MAX_N * 2], *pit = pool, *first[MAX_N];

int n, m, w[MAX_N], ans[MAX_N];
int dfn[MAX_N], fa[MAX_N], s[MAX_N], bel[MAX_N], dep[MAX_N],
chainDep[MAX_N];

void dfs1(int u, int f) {
        fa[u] = f, s[u] = 1;
        for (edge *e = first[u]; e; e = e->next) if (e->to != f) {
                dep[e->to] = dep[u] + 1;
                dfs1(e->to, u);
                s[u] += s[e->to];
        }
}

vector<int> chains[MAX_N];
void dfs2(int u, int num) {
        static int tsp = 0;
        dfn[u] = tsp++, bel[u] = num;
        chains[num].push_back(u);
        int mx = -1, id = 0;
        for (edge *e = first[u]; e; e = e->next)
```

```
            if (e->to != fa[u] && s[e->to] > mx) mx = s[id = e->to];
        if (mx == -1) return;
        chainDep[id] = chainDep[u] + 1;
        dfs2(id, num);
        for (edge *e = first[u]; e; e = e->next)
            if (e->to != fa[u] && e->to != id) chainDep[e->to] = 0, dfs2(e->to, e->to);
}

typedef pair<int, int> state;
vector<state> mark1[MAX_N], mark2[MAX_N];
#define sz(x) ((int)x.size())

int dis(int u, int v) {
        int d = 0;
        while (bel[u] != bel[v]) {
            if (dep[bel[u]] < dep[bel[v]]) swap(u, v);
            d += chainDep[u] + 1;
            u = fa[bel[u]];
        }
        if (dep[v] < dep[u]) swap(u, v);
        d += chainDep[v] - chainDep[u] + 1;
        return d - 1;
}
```

```
void solve(int u, int v) {
	int disU = 0, disV = dis(u, v), d, t;
	while (bel[u] != bel[v]) {
		if (dep[bel[u]] > dep[bel[v]]) {
			mark1[u].push_back(state(disU - (sz(chains[bel[u]]) - 1 - chainDep[u]), 1));
			disU += chainDep[u] + 1;
			u = fa[bel[u]];
		} else {
			d = disV - chainDep[v];
			mark2[bel[v]].push_back(state(d, 1));
			if (chainDep[v] + 1 != sz(chains[bel[v]])) {
				t = chains[bel[v]][chainDep[v] + 1];
				mark2[t].push_back(state(d, -1));
			}
			disV -= chainDep[v] + 1;
			v = fa[bel[v]];
		}
	}
	if (dep[u] < dep[v]) {
		mark2[u].push_back(state(disU - chainDep[u], 1));
		if (chainDep[v] + 1 != sz(chains[bel[v]])) {
			t = chains[bel[v]][chainDep[v] + 1];
```

```
                mark2[t].push_back(state(disU - chainDep[u], -1));
            }
        } else {
            mark1[u].push_back(state(disU - (sz(chains[bel[u]]) - 1 - chainDep[u]),
1));
            if (v != bel[v]) {
                mark1[fa[v]].push_back(state(disU - (sz(chains[bel[u]]) - 1 -
chainDep[u]), -1));
            }
        }
}

static int cnt[MAX_N * 2], ti[MAX_N * 2];
int now_t = 0;

inline int get(int x) {
    if (ti[x] != now_t) ti[x] = now_t, cnt[x] = 0;
    return cnt[x];
}

inline void add(int x, int v) {
    if (ti[x] != now_t) ti[x] = now_t, cnt[x] = 0;
    cnt[x] += v;
}
```

```
void cal() {
	for (int tt = 0; tt < n; ++tt) if (bel[tt] == tt) {
		int len = chains[tt].size();
		now_t++;
		for (int i = 0; i < len; ++i) {
			for (int j = 0; j < (int)mark2[chains[tt][i]].size(); ++j)
				add(mark2[chains[tt][i]][j].first + n,
mark2[chains[tt][i]][j].second);
			ans[chains[tt][i]] += get(w[chains[tt][i]] - i + n);
		}
		now_t++;
		for (int i = len - 1; i >= 0; --i) {
			for (int j = 0; j < (int)mark1[chains[tt][i]].size(); ++j) {
				add(mark1[chains[tt][i]][j].first + n,
mark1[chains[tt][i]][j].second);
			}
			ans[chains[tt][i]] += get(w[chains[tt][i]] - (len - 1 - i) + n);
		}
	}
}

int main() {
	scanf("%d%d", &n, &m);
```

```
    for (int i = 0, u, v; i < n - 1; ++i) {
        scanf("%d%d", &u, &v), --u, --v;
        first[u] = new (pit++) edge(first[u], v);
        first[v] = new (pit++) edge(first[v], u);
    }
    for (int i = 0; i < n; ++i) scanf("%d", w + i);
    dfs1(0, -1), dfs2(0, 0);
    for (int i = 0, u, v; i < m; ++i) {
        scanf("%d%d", &u, &v), u--, v--;
        solve(u, v);
    }
    cal();
    for (int i = 0; i < n; ++i) printf("%d%c", ans[i], i + 1 == n ? '\n' : ' ');
    return 0;
}
```

正解

```
//    Created by Sengxian on 2016/11/23.
//    Copyright (c) 2016 年  Sengxian. All rights reserved.
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
```

```
inline int readInt() {
	static int n, ch;
	n = 0, ch = getchar();
	while (!isdigit(ch)) ch = getchar();
	while (isdigit(ch)) n = n * 10 + ch - '0', ch = getchar();
	return n;
}

const int MAX_N = 300000 + 3, MAX_M = 300000 + 3;

struct edge {
	edge *next;
	int to;
	edge(edge *next = NULL, int to = 0): next(next), to(to) {}
} pool[(MAX_N + MAX_M) * 2], *pit = pool, *first[MAX_N], *qFirst[MAX_N];
int n, m, w[MAX_N], w1[MAX_N], w2[MAX_N];
int queryU[MAX_M], queryV[MAX_M], queryLCA[MAX_N], queryAns[MAX_M];
int fa[MAX_N], s[MAX_N], d[MAX_N], dfn[MAX_N], seq[MAX_N];

namespace disjoint_union {
	int ufa[MAX_N];

	inline void init(int n) {
```

```
        for (int i = 0; i < n; ++i) ufa[i] = i;
    }

    inline int find(int x) {
        return ufa[x] == x ? x : ufa[x] = find(ufa[x]);
    }

    inline void unite(int x, int y) {
        x = find(x), y = find(y);
        ufa[x] = y;
    }
}

using namespace disjoint_union;

void dfs(int u, int fa) {
    static bool vis[MAX_N];
    static int tsp = 0;
    vis[u] = true, ::fa[u] = fa, s[u] = 1;
    dfn[u] = tsp, seq[tsp++] = u;
    for (edge *e = first[u]; e; e = e->next) if (e->to != fa) {
        d[e->to] = d[u] + 1;
        dfs(e->to, u);
```

```
            unite(e->to, u);
            s[u] += s[e->to];
        }
        for (edge *q = qFirst[u]; q; q = q->next) {
            int v = queryU[q->to] == u ? queryV[q->to] : queryU[q->to];
            if (vis[v]) queryLCA[q->to] = find(v);
        }
}

typedef pair<int, int> state;
vector<state> mark1[MAX_N], mark2[MAX_N], pos[MAX_N];
int cnt1[MAX_N * 4], cnt2[MAX_N * 4];

inline void giveTag(int u, int v, int s, bool flag = false) {
        if (d[u] <= d[v]) {
            mark1[v].push_back(state(s - d[u], 1));
            if (fa[u] >= 0) mark1[fa[u]].push_back(state(s - d[u], -1));
        } else {
            mark2[u].push_back(state(s + d[u], 1));
            if (flag) mark2[v].push_back(state(s + d[u], -1));
            else if (fa[v] >= 0) mark2[fa[v]].push_back(state(s + d[u], -1));
        }
}
```

```
void solve() {
    for (int i = 0; i < n; ++i) w1[i] = w[i] - d[i];
    for (int i = 0; i < n; ++i) w2[i] = w[i] + d[i];
    for (int i = 0, u, v, LCA; i < m; ++i) {
        u = queryU[i], v = queryV[i], LCA = queryLCA[i];
        if (u == LCA || v == LCA) giveTag(u, v, 0);
        else giveTag(u, LCA, 0, true), giveTag(LCA, v, d[u] - d[LCA]);
    }
    for (int i = 0; i < n; ++i) {
        pos[dfn[i]].push_back(state(i, -1));
        pos[dfn[i] + s[i]].push_back(state(i, 1));
    }
    int del = n * 2;
    for (int i = 0, u; i < n; ++i) {
        u = seq[i];
        for (int j = 0; j < (int)mark1[u].size(); ++j)
            cnt1[mark1[u][j].first + del] += mark1[u][j].second;
        for (int j = 0; j < (int)mark2[u].size(); ++j)
            cnt2[mark2[u][j].first + del] += mark2[u][j].second;
        for (int j = 0; j < (int)pos[i + 1].size(); ++j) {
            state st = pos[i + 1][j];
            queryAns[st.first] += cnt1[w1[st.first] + del] * st.second;
```

```
                queryAns[st.first] += cnt2[w2[st.first] + del] * st.second;
            }
        }
}

int main() {
    n = readInt(), m = readInt();
    for (int i = 0, u, v; i < n - 1; ++i) {
        u = readInt() - 1, v = readInt() - 1;
        first[u] = new (pit++) edge(first[u], v);
        first[v] = new (pit++) edge(first[v], u);
    }
    for (int i = 0; i < n; ++i) w[i] = readInt();
    for (int i = 0; i < m; ++i) {
        queryU[i] = readInt() - 1, queryV[i] = readInt() - 1;
        qFirst[queryU[i]] = new (pit++) edge(qFirst[queryU[i]], i);
        qFirst[queryV[i]] = new (pit++) edge(qFirst[queryV[i]], i);
    }

    init(n), dfs(0, -1);

    solve();
    for (int i = 0; i < n; ++i)
```

```
            printf("%d%c", queryAns[i], i + 1 == n ? '\n' : ' ');

    return 0;

}
```

换教室 classroom

知识点

数学期望，动态规划，最短路

分析

首先有 $v \leq 300$，使用 floyd 算法求出两两点对之间的最短路。接着可以发现，这是一个经典的序列 DP 的模型，每个点有选或者不选两种决策，那么根据期望的线性性，容易发现每次 DP 只与最后两个选不选有关，考虑 $dp_{i,j,k}$ 为到第 i 个点，已经选了 j 个，k 表示第 i 个点是否选择换教室，那么转移就是根据后两个是否选择更换，来分情况讨论（设 p_i 为更换成功的概率，c_i 为不换的教室，d_i 为换的教室，$G_{i,j}$ 为 i 与 j 之间的最短路）：

如果点 i 不选，那么根据上一个选不选来计算期望（期望的线性告诉我们可以这样做）。

$$dp_{i,j,0} = \min \begin{cases} dp_{i-1,j,0} + G_{c_{i-1},c_i} \\ dp_{i-1,j,1} + p_{i-1} \cdot G_{d_{i-1},c_i} + (1 - p_{i-1}) \cdot G_{c_{i-1},c_i} \end{cases}$$

如果点 i 选，也是根据上一个选不选来计算期望，第二个式子会比较麻烦，有四种情况。

$$dp_{i,j,1} = \min \begin{cases} dp_{i-1,j-1,0} + p_i \cdot G_{c_{i-1},d_i} + (1 - p_i) \cdot G_{c_{i-1},c_i} \\ dp_{i-1,j-1,1} + p_{i-1} \cdot p_i \cdot G_{d_{i-1},d_i} + (1 - p_{i-1}) \cdot p_i \cdot G_{c_{i-1},d_i} \quad + p_{i-1} \cdot (1 - p_i) \cdot G_{d_{i-1},c_i} + (1 - p_{i-1}) \cdot (1 - p_i) \cdot G_{d_{i-1},d_i} \end{cases}$$

代码

```
//    Created by Sengxian on 2016/11/23.

//    Copyright (c) 2016 年  Sengxian. All rights reserved.

#include <bits/stdc++.h>

using namespace std;
```

```
typedef long long ll;
inline int readInt() {
	static int n, ch;
	n = 0, ch = getchar();
	while (!isdigit(ch)) ch = getchar();
	while (isdigit(ch)) n = n * 10 + ch - '0', ch = getchar();
	return n;
}

const int MAX_N = 2000 + 3, MAX_M = 2000 + 3, MAX_V = 300 + 3;
int n, m, v, e, c[MAX_N], d[MAX_N];
double p[MAX_N];
int G[MAX_V][MAX_V];

void floyd() {
	for (int k = 0; k < v; ++k)
		for (int i = 0; i < v; ++i)
			for (int j = 0; j < v; ++j)
				G[i][j] = min(G[i][j], G[i][k] + G[k][j]);
}

double dp[MAX_N][MAX_M][2];
void solve() {
```

```
	for (int i = 0; i < n; ++i)

		for (int j = 0; j <= m; ++j)

			dp[i][j][0] = dp[i][j][1] = 1e30;

	dp[0][0][0] = 0.0, dp[0][1][1] = 0.0;



	for (int i = 1; i < n; ++i)

		for (int j = 0; j <= m; ++j) {

			dp[i][j][0] = min(dp[i - 1][j][0] + G[c[i - 1]][c[i]], dp[i - 1][j][1] + p[i - 1] * G[d[i - 1]][c[i]] + (1 - p[i - 1]) * G[c[i - 1]][c[i]]);

			if (j) dp[i][j][1] = min(dp[i - 1][j - 1][0] + p[i] * G[c[i - 1]][d[i]] + (1 - p[i]) * G[c[i - 1]][c[i]], dp[i - 1]j - 1][1] + p[i] * p[i - 1] * G[d[i - 1]][d[i]] + (1 - p[i]) * p[i - 1] * G[d[i - 1]][c[i]] + p[i] * (1 - p[i - 1]) * G[c[i - 1]][d[i]] + (1 - p[i]) * (1 - p[i - 1]) * G[c[i - 1]][c[i]]);

		}



	double ans = 1e30;

	for (int i = 0; i <= m; ++i) ans = min(ans, min(dp[n - 1][i][0], dp[n - 1][i][1]));

	printf("%.2f\n", ans);

}



int main() {

	n = readInt(), m = readInt(), v = readInt(), e = readInt();

	for (int i = 0; i < n; ++i) c[i] = readInt() - 1;

	for (int i = 0; i < n; ++i) d[i] = readInt() - 1;
```

```
        for (int i = 0; i < n; ++i) scanf("%lf", p + i);
        memset(G, 0x3f, sizeof G);
        for (int i = 0, f, t, c; i < e; ++i) {
            f = readInt() - 1, t = readInt() - 1, c = readInt();
            G[f][t] = G[t][f] = min(G[f][t], c);
        }
        for (int i = 0; i < v; ++i) G[i][i] = 0;
        floyd();
        solve();
        return 0;
}
```

总结

Day 1 的题还是有一定难度的，第二题是一个不错的题，放在这个位置有不小的震慑力，而且可以意外地让一批伪高手栽下跟头——想不出来正解，也打不出高级数据结构，结果爆 0。第三题如果熟悉期望 DP 的话，是一道比较简单的题，可见此题的目的主要还是用于普及一下数学期望，以便在后续比赛中出现。