

# 差分约束

## 人员

阮文璋、褚锦轩、王承周、王毅博、许睿谦 到课

## 上周作业检查

上周作业链接: <https://cppoj.kids123code.com/contest/2585>

#	用户名	姓名	编程分	时间	A	B	C
1	wangchengzhou	王承周	300	4909	100	100	100
2	wangyibo	王毅博	200	1864		100	100
3	ruanwenzhang	阮文璋	200	4374	100	100	
4	chujinxuan	褚锦轩	200	9812	100	100	

## 本周作业

<https://cppoj.kids123code.com/contest/2685> (课上讲了 A ~ C 题, 课后作业是 D 题)

## 课堂表现

今天课上讲了 差分约束 这个知识点, 并且讲解了一道前缀和优化 dp 的题目, 前缀和优化dp 是 dp 中比较常用的技巧, 需要同学们掌握。

## 课堂内容

### 【模板】差分约束

把  $x_i - x_j \leq c$

转化为  $x_i \leq x_j + c$

转化为一条点 j 到点 i 的, 边权为 c 的边即可

设定一个超级源点, 从超级源点往所有点连边, 跑一遍最短路即可, 因为有负边, 所以需要用 SPFA 求最短路

如果图中有负环, 说明无解

```
#include <bits/stdc++.h>

using namespace std;

const int maxn = 5e3 + 5;
const int inf = 0x3f3f3f3f;
```

```

struct node {
    int to, value;
};
vector<node> vec[maxn];
int dis[maxn], cnt[maxn];
bool st[maxn];

int main()
{
    int n, m; cin >> n >> m;
    while (m -- ) {
        int a, b, c; cin >> a >> b >> c;
        vec[b].push_back({a,c});
    }
    for (int i = 1; i <= n; ++i) vec[0].push_back({i,0});

    memset(dis, 0x3f, sizeof(dis));
    queue<int> q; q.push(0); dis[0] = 0; st[0] = true;
    while (!q.empty()) {
        int u = q.front(); q.pop();
        st[u] = false;
        for (node it : vec[u]) {
            if (dis[u]+it.value < dis[it.to]) {
                dis[it.to] = dis[u]+it.value; ++cnt[it.to];
                if (cnt[it.to] == n) { cout << "NO" << endl; return 0; }
                if (!st[it.to]) { q.push(it.to); st[it.to] = true; }
            }
        }
    }

    for (int i = 1; i <= n; ++i) cout << dis[i] << " ";
    cout << endl;
    return 0;
}

```

## 小 K 的农场

按照题目的要求建边, 可以把原问题转化为一个差分约束问题

```

#include <bits/stdc++.h>

using namespace std;

const int maxn = 5e3 + 5;
const int inf = 0x3f3f3f3f;
struct node {
    int to, value;
};
vector<node> vec[maxn];
int dis[maxn], cnt[maxn];
bool st[maxn];

```

```

int main()
{
    int n, m; cin >> n >> m;
    while (m -- ) {
        int op; cin >> op;
        if (op == 1) {
            int a, b, c; cin >> a >> b >> c;
            vec[a].push_back({b,-c});
        } else if (op == 2) {
            int a, b, c; cin >> a >> b >> c;
            vec[b].push_back({a,c});
        } else {
            int a, b; cin >> a >> b;
            vec[a].push_back({b,0}), vec[b].push_back({a,0});
        }
    }
    for (int i = 1; i <= n; ++i) vec[0].push_back({i,0});

    memset(dis, 0x3f, sizeof(dis));
    queue<int> q; q.push(0); dis[0] = 0; st[0] = true;
    while (!q.empty()) {
        int u = q.front(); q.pop();
        st[u] = false;
        for (node it : vec[u]) {
            if (dis[u]+it.value < dis[it.to]) {
                dis[it.to] = dis[u]+it.value; ++cnt[it.to];
                if (cnt[it.to] == n) { cout << "No" << endl; return 0; }
                if (!st[it.to]) { q.push(it.to); st[it.to] = true; }
            }
        }
    }

    cout << "Yes" << endl;
    return 0;
}

```

## Yet Another Grid Task

前缀和优化 dp

$f[j][i]$ : 第  $j$  列, 黑色高度为  $i$  时, 有多少方案

$p[j][i]$ : 第  $j$  列, 黑色高度为  $1 \sim i$  时, 一共有多少方案

```

#include <bits/stdc++.h>

using namespace std;

typedef long long LL;
const int maxn = 2000 + 5;

```

```
const int mod = 998244353;
char s[maxn][maxn];
int h[maxn];
int f[maxn][maxn]; // f[j][i]: 第 j 列, 黑色高度为 i 时, 有多少方案
int p[maxn][maxn];

int main()
{
    int n, m; cin >> n >> m;
    for (int i = 1; i <= n; ++i) cin >> (s[i]+1);

    for (int j = 1; j <= m; ++j) {
        for (int i = 1; i <= n; ++i) {
            if (s[i][j] == '#') { h[j] = n-i+1; break; }
        }
    }

    for (int j = 1; j <= m; ++j) {
        for (int i = h[j]; i <= n; ++i) {
            if (j == 1) f[j][i] = 1;
            else f[j][i] = p[j-1][i+1];
        }
    }

    p[j][0] = f[j][0];
    for (int i = 1; i <= n+1; ++i) p[j][i] = (p[j][i-1] + f[j][i]) % mod;
}

int res = 0;
for (int i = h[m]; i <= n; ++i) res = (res + f[m][i]) % mod;
cout << res << endl;
return 0;
}
```