差分约束

人员

袁晨峻、李锦澍 到课

上周作业检查

上周作业链接: https://cppoj.kids123code.com/contest/2583

比赛概况 题目列表 选择题列表 提交记录 ★ 实时榜单

王向东老师周日八点半C++综合练习

刷新

#	用户名	姓名	编程分	时间	A	B	C
1	yangjunyan	杨俊彦	300	4322	100	100	100
2	lijinshu	李锦澍	300	4441	100	100	100
3	yuanchenjun	袁晨峻	300	6512	100	100	100
4	chenxinmiao	陈欣妙	200	4546	100	100	0
5	liuyichen	刘奕辰	200	10342	100	100	

本周作业

https://cppoj.kids123code.com/contest/2683 (课上讲了 A ~ C 题, 课后作业是 E 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 xi - xj <= c

转化为 xi <= xj + 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

```cpp
#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~i 时, 一共有多少方案

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

using namespace std;

typedef long long LL;
```

```cpp
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;
}
```