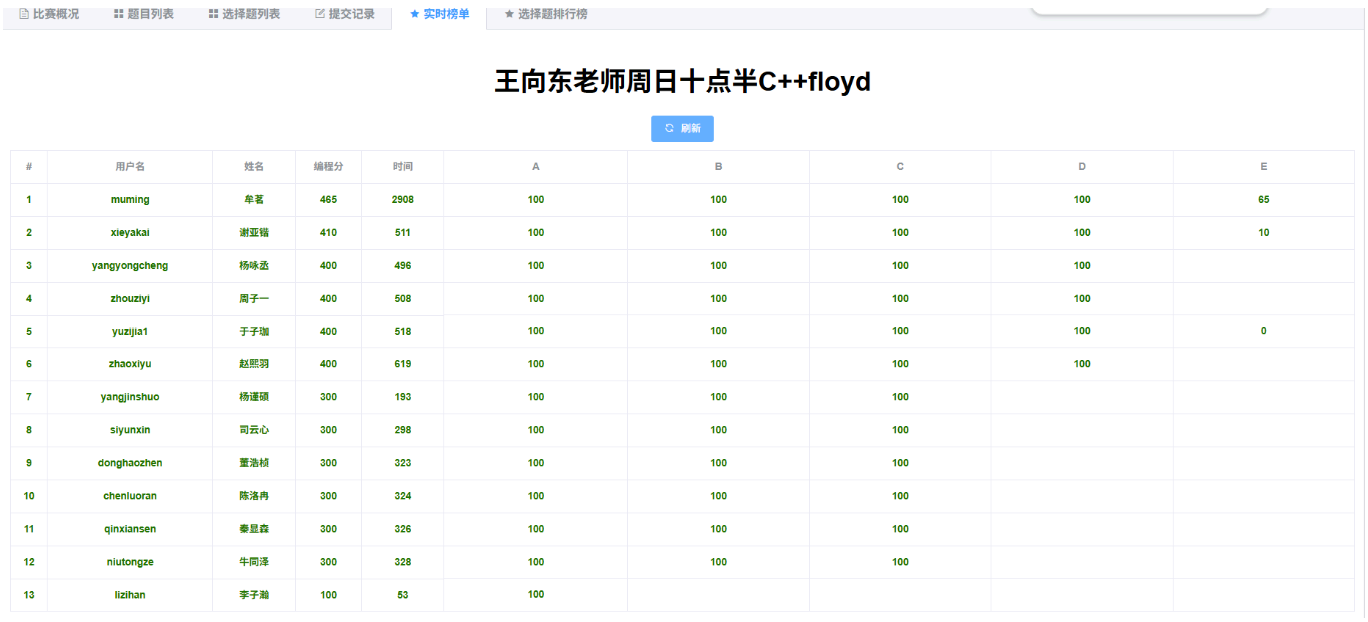
图论综合练习

人员

赵熙羽、司云心、于子珈、陈洛冉、谢亚锴、杨咏丞、杨瑾硕、董浩桢、牟茗、秦显森、牛同泽、隋天乙 到课, 周子一 线上

上周作业检查

上周作业链接: https://cppoj.kids123code.com/contest/1788



王向东老师周日十点半C++floyd

#	用户名	姓名	编程分	时间	A	B	C	D	E
1	muming	牟茗	465	2908	100	100	100	100	65
2	xieyakai	谢亚锴	410	511	100	100	100	100	10
3	yangyongcheng	杨咏丞	400	496	100	100	100	100	
4	zhouziyi	周子一	400	508	100	100	100	100	
5	yuzijia1	于子珈	400	518	100	100	100	100	0
6	zhaoxiyu	赵熙羽	400	619	100	100	100	100	
7	yangjinshuo	杨谨硕	300	193	100	100	100		
8	siyunxin	司云心	300	298	100	100	100		
9	donghaozhen	董浩桢	300	323	100	100	100		
10	chenluoran	陈洛冉	300	324	100	100	100		
11	qinxiansen	秦显森	300	326	100	100	100		
12	niutongze	牛同泽	300	328	100	100	100		
13	lizihan	李子瀚	100	53	100				

本周作业

https://cppoj.kids123code.com/contest/1910 (课上讲了 A ~ C 题, 课后作业是 D 题)

课堂表现

今天的 B 题会比较复杂一些, 课上很多同学都是在老师的帮助下把这道题通过的, 课下需要再好好复习复习 B 题。

课堂内容

[USACO08OPEN] Clear And Present Danger S

先用 floyd 求出来任意两点间最短路, 后续每次移动的长度就可以 O(1) 知道了

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

using namespace std;

const int maxn = 100 + 5, M = 10000 + 5;
const int inf = 0x3f3f3f3f;
```

```
int f[maxn][maxn], w[M];

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

  for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= n; ++j) cin >> f[i][j];
  }

  for (int k = 1; k <= n; ++k) {
    for (int i = 1; i <= n; ++i) {
      for (int j = 1; j <= n; ++j) f[i][j] = min(f[i][j], f[i][k]+f[k][j]);
    }
  }

  int res = 0;
  for (int i = 2; i <= m; ++i) {
    int last = w[i-1], now = w[i];
    res += f[last][now];
  }
  cout << res << endl;
  return 0;
}
```

Road Blocked

考虑删边操作是比较复杂的, 因为删完边后点与点之间的最短路变化可能变化很大

因此, 可以把整个问题反过来思考, 考虑加边操作

每次加完一条边后, 最短路的更新只可能通过这条边进行跟新, 所以最短路的更新是 O(n^2) 级别的

题目保证, 最多添加 300 条边, 那么这个题就解决了

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

const int N = 300 + 5, M = 2e5 + 5;
const int inf = 0x3f3f3f3f3f3f3f3f;
struct Edge {
  int a, b, c;
} w[N*N];
bool st[N*N];
int f[N][N];

struct node {
  int op, id, x, y;
} q[M];
```

```
signed main()
{
  int n, m, Q; cin >> n >> m >> Q;
  for (int i = 1; i <= m; ++i) cin >> w[i].a >> w[i].b >> w[i].c, st[i] = true;
  for (int i = 1; i <= Q; ++i) {
    cin >> q[i].op;
    if (q[i].op == 1) cin >> q[i].id, st[q[i].id] = false;
    else cin >> q[i].x >> q[i].y;
  }

  memset(f, 0x3f, sizeof(f));
  for (int i = 1; i <= n; ++i) f[i][i] = 0;
  for (int i = 1; i <= m; ++i) {
    int a = w[i].a, b = w[i].b, c = w[i].c;
    if (st[i]) f[a][b] = f[b][a] = c;
  }

  for (int k = 1; k <= n; ++k) {
    for (int i = 1; i <= n; ++i) {
      for (int j = 1; j <= n; ++j) f[i][j] = min(f[i][j], f[i][k]+f[k][j]);
    }
  }

  vector<int> ans;
  for (int i = Q; i >= 1; --i) {
    int op = q[i].op, id = q[i].id, x = q[i].x, y = q[i].y;
    if (op == 1) {
      int a = w[id].a, b = w[id].b, c = w[id].c;
      f[a][b] = min(f[a][b], c);
      for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= n; ++j) {
          int value = min(f[i][a]+f[b][j], f[i][b]+f[a][j]) + f[a][b];
          f[i][j] = min(f[i][j], value);
        }
      }
    } else {
      int dis = (f[x][y]==inf ? -1 : f[x][y]);
      ans.push_back(dis);
    }
  }

  reverse(ans.begin(), ans.end());
  for (int i : ans) cout << i << endl;
  return 0;
}
```

[CERC1998] 请柬

需要求所有 1 -> i 的最短路和所有 i -> 1 的最短路

1 -> i 的最短路: 可以以 1 为起点用 dijkstra 跑一遍就能求出来

i -> 1 的最短路: 通过建反图, 然后以 1 为起点跑一遍 dijkstra 即可

```
#include <bits/stdc++.h>
#define int long long

using namespace std;

const int maxn = 2e6 + 5;
const int inf = 0x3f3f3f3f3f3f3f3f;
struct eInfo {
  int to, value;
};
vector<eInfo> vec[maxn];

struct node {
  int id, d;
  bool operator < (const node& p) const { return d < p.d; }
  bool operator > (const node& p) const { return d > p.d; }
};
int dis[maxn];
bool st[maxn];

void dijkstra(int _st) {
  priority_queue<node, vector<node>, greater<node>> q;
  q.push({_st, 0}); dis[_st] = 0;
  while (!q.empty()) {
    node u = q.top(); q.pop();
    int id = u.id, d = u.d;
    if (st[id]) continue;
    st[id] = true;

    for (eInfo it : vec[id]) {
      if (dis[it.to] > d+it.value) {
        dis[it.to] = d+it.value; q.push({it.to, dis[it.to]});
      }
    }
  }
}

signed main()
{
  int n, m; cin >> n >> m;
  while (m -- ) {
    int a, b, c; cin >> a >> b >> c;
    vec[a].push_back({b,c}), vec[n+b].push_back({n+a,c});
  }

  memset(dis, 0x3f, sizeof(dis)), memset(st, false, sizeof(st));
  dijkstra(1), dijkstra(n+1);
  int res = 0;
  for (int i = 2; i <= n; ++i) res += dis[i] + dis[n+i];
  cout << res << endl;
```

```
    return 0;
}
```