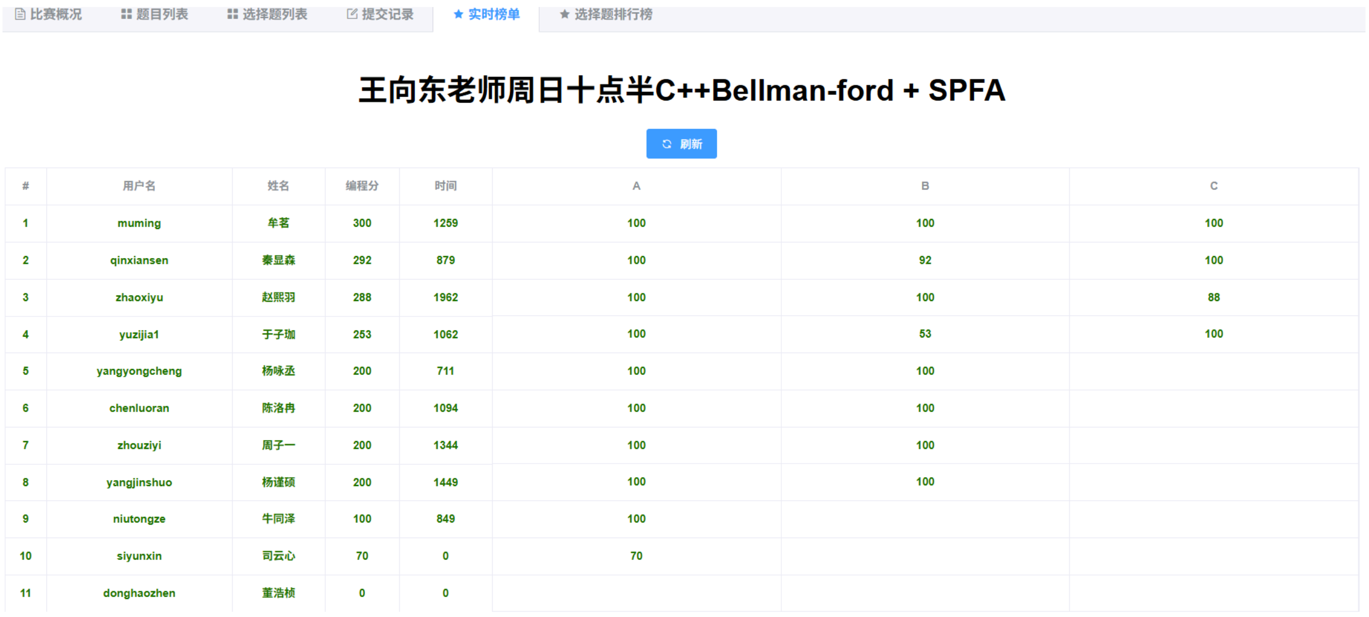
分层图

人员

赵熙羽、于子珈、陈洛冉、杨咏丞、董浩桢、李子瀚、秦显森、周子一、谢亚锴 到课

上周作业检查

上周作业链接: https://cppoj.kids123code.com/contest/2020



王向东老师周日十点半C++Bellman-ford + SPFA

#	用户名	姓名	编程分	时间	A	B	C
1	muming	牟茗	300	1259	100	100	100
2	qinxiansen	秦显森	292	879	100	92	100
3	zhaoxiyu	赵熙羽	288	1962	100	100	88
4	yuzijia1	于子珈	253	1062	100	53	100
5	yangyongcheng	杨咏丞	200	711	100	100	
6	chenluoran	陈洛冉	200	1094	100	100	
7	zhouziyi	周子一	200	1344	100	100	
8	yangjinshuo	杨谨硕	200	1449	100	100	
9	niutongze	牛同泽	100	849	100		
10	siyunxin	司云心	70	0	70		
11	donghaozhen	董浩桢	0	0			

本周作业

https://cppoj.kids123code.com/contest/2170 (课上讲了 A ~ B 题, 课后作业是 C 题)

课堂表现

今天课上给同学们讲了 分层图 这个思想

课上反映的问题是许多同学 dijkstra 写不熟, dijkstra 可以说是图论里面最重要的算法, 这些同学课下要把 dijkstra 多写几遍

课堂内容

[USACO06DEC] Wormholes G (上周作业)

虫洞, 判断是否有负环

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

using namespace std;

const int maxn = 2e3 + 5;
const int inf = 0x3f3f3f3f;
struct Edge {
  int from, to, value;
};
vector<Edge> edges;
int dis[maxn], f[maxn];

void solve() {
  int n, m, W; cin >> n >> m >> W;
  edges.clear();

  while (m -- ) {
    int a, b, c; cin >> a >> b >> c;
    edges.push_back({a,b,c}), edges.push_back({b,a,c});
  }
  while (W -- ) {
    int a, b, c; cin >> a >> b >> c; edges.push_back({a,b,-c});
  }

  memset(dis, 0x3f, sizeof(dis));
  memset(f, 0, sizeof(f));

  for (int i = 1; i <= n; ++i) {
    for (Edge it : edges) dis[it.to] = min(dis[it.to], dis[it.from]+it.value);
    if (i == n-1) {
      for (int j = 1; j <= n; ++j) f[j] = dis[j];
    }
  }

  for (int i = 1; i <= n; ++i) {
    if (f[i] != dis[i]) { cout << "YES" << endl; return; }
  }
  cout << "NO" << endl;
}

int main()
{
  int T; cin >> T;
  while (T -- ) solve();
  return 0;
}
```

[USACO09FEB] Revamping Trails G

分层图模板题, 可以把原图分成 K+1 层, 第 i 层往第 i+1 层的边权为 0, 然后跑最短路即可

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

using namespace std;
```

```cpp
const int maxn = 4e5 + 5;
const int inf = 0x3f3f3f3f3f3f3f3f;
struct edge {
  int to, value;
};
vector<edge> vec[maxn];

int n, m, K;
int get_id(int a, int b) { return a*(K+1)+b; }

struct node {
  int d, id;
  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) {
  memset(dis, 0x3f, sizeof(dis));
  priority_queue<node, vector<node>, greater<node>>q;
  q.push({0, _st}); dis[_st] = 0;
  while (!q.empty()) {
    node u = q.top(); q.pop();
    int d = u.d, id = u.id;
    if (st[id]) continue;
    st[id] = true;

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

signed main()
{
  cin >> n >> m >> K;
  while (m -- ) {
    int a, b, c; cin >> a >> b >> c;
    for (int i = 0; i <= K; ++i) {
      int a1 = get_id(a, i), b1 = get_id(b, i);
      int a2 = get_id(a, i+1), b2 = get_id(b, i+1);
      vec[a1].push_back({b1,c}), vec[b1].push_back({a1,c});
      if (i != K) vec[a1].push_back({b2,0}), vec[b1].push_back({a2,0});
    }
  }

  dijkstra(get_id(1,0));

  int res = inf;
  for (int i = 0; i <= K; ++i) res = min(res, dis[get_id(n,i)]);
  cout << res << endl;
  return 0;
}
```

[CCC 2015 S4] Convex Hull

在 总h < k 的前提下, 让 t 的总和最小, 相当于在跨越不超过 k 层的前提下, 求最短路

当输入 u v t h 时, 建边关系为 {u,i} -> {v,h+i} 和 {v,i} -> {u,h+i}, 边权为 t

然后跑最短路即可

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

using namespace std;

const int maxn = 8e5 + 5;
const int inf = 0x3f3f3f3f3f3f3f3f;
struct edge {
  int to, value;
};
vector<edge> vec[maxn];

int n, m, K;
int get_id(int a, int b) { return a*K+b; }

struct node {
  int d, id;
  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) {
  memset(dis, 0x3f, sizeof(dis));
  priority_queue<node, vector<node>, greater<node>>q;
  q.push({0, _st}); dis[_st] = 0;
  while (!q.empty()) {
    node u = q.top(); q.pop();
    int d = u.d, id = u.id;
    if (st[id]) continue;
    st[id] = true;

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

```cpp
signed main()
{
  cin >> K >> n >> m;
  while (m -- ) {
    int u, v, t, h; cin >> u >> v >> t >> h;
    for (int i = 0, j = h; j <= K-1; ++i, ++j) {
      int u1 = get_id(u, i), v1 = get_id(v, i);
      int u2 = get_id(u, j), v2 = get_id(v, j);
      vec[u1].push_back({v2,t}), vec[v1].push_back({u2,t});
    }
  }

  int _st, _ed; cin >> _st >> _ed;
  dijkstra(get_id(_st,0));

  int res = inf;
  for (int i = 0; i <= K-1; ++i) res = min(res, dis[get_id(_ed,i)]);
  cout << (res==inf ? -1 : res) << endl;
  return 0;
}
```