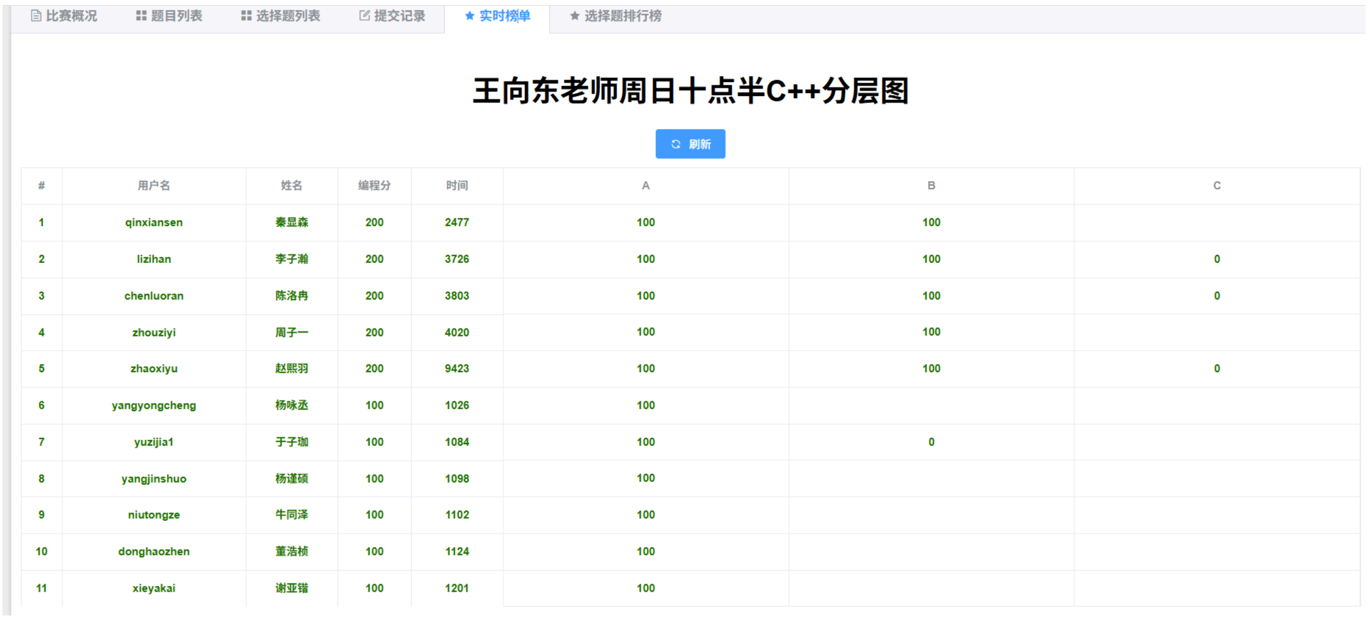
综合练习

人员

赵熙羽、陈洛冉、李子瀚、谢亚锴、隋天乙、牛同泽、杨瑾硕 到课, 周子一、司云心、于子珈 线上

上周作业检查

上周作业链接: https://cppoj.kids123code.com/contest/2170



王向东老师周日十点半C++分层图

#	用户名	姓名	编程分	时间	A	B	C
1	qinxiansen	秦显森	200	2477	100	100	
2	lizihan	李子瀚	200	3726	100	100	0
3	chenluoran	陈洛冉	200	3803	100	100	0
4	zhouziyi	周子一	200	4020	100	100	
5	zhaoxiyu	赵熙羽	200	9423	100	100	0
6	yangyongcheng	杨咏丞	100	1026	100		
7	yuzijia1	于子珈	100	1084	100	0	
8	yangjinshuo	杨瑾硕	100	1098	100		
9	niutongze	牛同泽	100	1102	100		
10	donghaozhen	董浩桢	100	1124	100		
11	xieyakai	谢亚锴	100	1201	100		

本周作业

https://cppoj.kids123code.com/contest/2239 (课上讲了 A ~ C 题, 课后作业是 D 题必做, E 题选做)

课堂表现

今天题目整体不难, 同学们课上完成的都普遍不错。

课堂内容

[JLOI2011] 飞行路线 (上周作业)

最多可以 k 条边免费, 分为 k+1 层图即可

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

using namespace std;

const int maxn = 2e5 + 5;
const int inf = 0x3f3f3f3f;
struct node {
  int to, value;
  bool operator > (const node& p) const { return value > p.value; }
};
vector<node> vec[maxn];
int n, m, k;
int getId(int u, int c) { return u*(k+1) + c; }
int dis[maxn];
bool st[maxn];

void dijkstra(int id) {
  memset(dis, 0x3f, sizeof(dis));
  priority_queue<node, vector<node>, greater<node>>q; q.push({id,0}); dis[id] = 0;
  while (!q.empty()) {
    node u = q.top(); q.pop();
    int u_id = u.to, u_dis = u.value;
    if (st[u_id]) continue;
    st[u_id] = true;

    for (node it : vec[u_id]) {
      if (u_dis + it.value < dis[it.to]) {
        dis[it.to] = u_dis + it.value; q.push({it.to, dis[it.to]});
      }
    }
  }
}

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

  dijkstra(getId(st,1));

  int res = inf;
  for (int i = 1; i <= k+1; ++i) res = min(res, dis[getId(ed,i)]);
  cout << res << endl;
  return 0;
}
```

扩散

二分 + 并查集: check 中 n^2 判断 mid 时间所有点能否连通到一块即可

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

using namespace std;

const int maxn = 50 + 5;
struct node {
  int x, y;
} w[maxn];

int f[maxn];
int fFind(int x) {
  if (f[x] != x) f[x] = fFind(f[x]);
  return f[x];
}

bool check(int mid, int n) {
  for (int i = 1; i <= n; ++i) f[i] = i;

  for (int i = 1; i <= n; ++i) {
    for (int j = i+1; j <= n; ++j) {
      int d = abs(w[i].x-w[j].x) + abs(w[i].y-w[j].y);
      if (d <= 2*mid) {
        int fi = fFind(i), fj = fFind(j);
        f[fi] = fj;
      }
    }
  }

  set<int> s;
  for (int i = 1; i <= n; ++i) s.insert(fFind(i));

  return (int)s.size()==1;
}

int main()
{
  int n; cin >> n;
  for (int i = 1; i <= n; ++i) cin >> w[i].x >> w[i].y;

  int l = 1, r = 1e9+10;
  while (l <= r) {
    int mid = (l + r) / 2;
    if (check(mid,n)) r = mid-1;
    else l = mid+1;
  }
  cout << l << endl;
  return 0;
}
```

走廊泼水节

kruskal, 中间维护并查集时额外维护每个集合的大小

当把 u 和 v 这两个点用边权 w 连接时, 额外贡献为 (sz[u]*sz[v]-1) * (w+1)

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

using namespace std;

const int maxn = 6000 + 5;
int f[maxn], sz[maxn];
int fFind(int x) {
  if (f[x] != x) f[x] = fFind(f[x]);
  return f[x];
}

struct node {
  int from, to, value;
  bool operator < (const node& p) const { return value < p.value; }
};

void solve() {
  int n; cin >> n;
  for (int i = 1; i <= n; ++i) f[i] = i, sz[i] = 1;
  vector<node> vec;
  for (int i = 1; i <= n-1; ++i) {
    int a, b, c; cin >> a >> b >> c; vec.push_back({a,b,c});
  }

  sort(vec.begin(), vec.end());
  int res = 0;
  for (node it : vec) {
    int x = it.from, y = it.to, value = it.value;
    int fx = fFind(x), fy = fFind(y);
    res += (sz[fx]*sz[fy]-1) * (value+1);
    f[fx] = fy, sz[fy] += sz[fx];
  }

  cout << res << endl;
}

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

[蓝桥杯 2020 省 AB3] 限高杆

分层图, 对于 d==0 的边, 不需要跨层; 对于 d==1 的边, 需要建跨层的边

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

using namespace std;

const int maxn = 4e4 + 5;
const int inf = 0x3f3f3f3f;
int get_id(int a, int b) { return a*3+b; }
struct eInfo {
  int to, value;
};
vector<eInfo> vec[maxn];

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

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

int main()
{
  int n, m; cin >> n >> m;
  while (m -- ) {
    int a, b, c, d; cin >> a >> b >> c >> d;
    for (int i = 1; i <= 3; ++i) {
      int now_a = get_id(a,i), now_b = get_id(b,i);
      int next_a = get_id(a, i+1), next_b = get_id(b, i+1);
      if (!d) vec[now_a].push_back({now_b,c}), vec[now_b].push_back({now_a,c});
      else if (i != 3) vec[now_a].push_back({next_b,c}),
vec[now_b].push_back({next_a,c});
    }
  }

  dijkstra(get_id(1,1));
```

```
    int last = dis[get_id(n,1)], now = min(dis[get_id(n,2)], dis[get_id(n,3)]);
    cout << last - now << endl;
    return 0;
}
```