综合练习

人员

赵熙羽、司云心、牛同泽、于子珈、秦显森、陈洛冉 到课, 谢亚锴、周子一、杨咏丞、李子瀚 线上

上周作业检查

上周作业链接: https://cppoj.kids123code.com/contest/1478

王向东老师周日十点半C++kruskal

刷新

#	用户名	姓名	编程分	时间	A	B	C	D	E
1	yuzijia1	于子珈	416	2530	100	100	100	100	16
2	zhaoxiyu	赵熙羽	400	2797	100	100	100	100	
3	niutongze	牛同泽	400	2803	100	100	100	100	
4	chenluoran	陈洛冉	400	2925	100	100	100	100	0
5	zhouziyi	周子一	400	17759	100	100	100		100
6	siyunxin	司云心	332	5245	100	100	100		32
7	qinxiansen	秦显森	308	2687	100	100	100		8
8	yangjinshuo	杨谨硕	308	2761	100	100	100		8
9	lizihan	李子瀚	300	2647	100	100	100		
10	yangyongcheng	杨咏丞	300	4890	100	100	100		
11	xieyakai	谢亚锴	200	354	100	100			
12	suitianyi	隋天乙	100	298	100	0			

本周作业

https://cppoj.kids123code.com/contest/1587 (课上讲了 A ~ C 题, 课后作业是 D 题)

课堂表现

今天的 C 题思路不难, 是个 二分 + dijkstra 的题, 但是同学们课上整体做的不是很好, 课下要好好调一调自己的代码, 看看是哪里写错了。

课堂内容

营救 (上周作业)

最小生成树, 直到把 s 和 t 连到同一个并查集中为止

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

using namespace std;

const int maxn = 2e4 + 5;
struct node {
  int u, v, value;
```

```
} w[maxn];

bool cmp(node p, node q) {
  return p.value < q.value;
}

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

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

  for (int i = 1; i <= m; ++i) cin >> w[i].u >> w[i].v >> w[i].value;
  sort(w+1, w+m+1, cmp);

  for (int i = 1; i <= m; ++i) {
    int u = w[i].u, v = w[i].v, value = w[i].value;
    int fu = fFind(u), fv = fFind(v);
    if (fu != fv) f[fu] = fv;

    if (fFind(s) == fFind(t)) { cout << value << endl; break; }
  }
  return 0;
}
```

口袋的天空

按照最小生成树的方法来做, 直到剩 K 个联通块为止

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

using namespace std;

const int maxn = 1000 + 5;
struct node {
  int a, b, value;
  bool operator < (const node& p) const { return value < p.value; }
};
vector<node> edges;

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

```
int main()
{
  int n, m, K; cin >> n >> m >> K;
  while (m -- ) {
    int a, b, value; cin >> a >> b >> value; edges.push_back({a,b,value});
  }

  sort(edges.begin(), edges.end());
  for (int i = 1; i <= n; ++i) f[i] = i;

  if (n == K) { cout << 0 << endl; return 0; }

  int res = 0;
  for (node it : edges) {
    int a = it.a, b = it.b, value = it.value;
    int fa = fFind(a), fb = fFind(b);
    if (fa != fb) {
      n--; res += value; f[fa] = fb;
      if (n == K) { cout << res << endl; return 0; }
    }
  }

  cout << "No Answer" << endl;
  return 0;
}
```

[USACO08OCT] Watering Hole G

建一个 n+1 号点, 把 n+1 号点与其他 n 个点连接, 边权为 w[i] 的值

然后给这 n+1 个点跑最小生成树即可

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

using namespace std;

const int maxn = 300 + 5;
int w[maxn];
struct node {
  int a, b, value;
  bool operator < (const node& p) const { return value < p.value; }
};
vector<node> vec;

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

int main()
```

```
{
  int n; cin >> n;
  for (int i = 1; i <= n; ++i) {
    int x; cin >> x; vec.push_back({n+1,i,x});
  }

  for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= n; ++j) {
      int x; cin >> x; vec.push_back({i,j,x});
    }
  }

  sort(vec.begin(), vec.end());
  for (int i = 1; i <= n+1; ++i) f[i] = i;

  int res = 0;
  for (node it : vec) {
    int a = it.a, b = it.b, value = it.value;
    int fa = fFind(a), fb = fFind(b);
    if (fa != fb) f[fa] = fb, res += value;
  }
  cout << res << endl;
  return 0;
}
```

通往奥格瑞玛的道路

二分 + dijkstra

二分城市收费的最大值, 后续 check 时只能选收费更低的城市, 求起点到终点的最短路, 看最短路是否符合题目要求

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

using namespace std;

const int maxn = 1e4 + 5;
const int inf = 0x3f3f3f3f3f3f3f3f;
struct edge {
    int to, value;
};
vector<edge> 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], st[maxn];
```

```cpp
int w[maxn];

bool check(int mid, int n, int B) {
    if (w[1]>mid || w[n]>mid) return false;

    memset(dis, 0x3f, sizeof(dis)), memset(st, false, sizeof(st));
    priority_queue<node, vector<node>, greater<node>>q;
    dis[1] = 0, q.push({dis[1],1});

    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 (w[it.to] > mid) continue;
            if (d+it.value < dis[it.to]) {
                dis[it.to] = d+it.value, q.push({dis[it.to],it.to});
            }
        }
    }

    return dis[n]<=B;
}

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

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

    if (l > 1e9) cout << "AFK" << endl;
    else cout << l << endl;
    return 0;
}
```