综合练习

人员

赵熙羽、杨瑾硕、谢亚锴、于子珈、刘闯速、牛同泽、孙靖轲、陈洛冉、秦显森、高健桓、于霄龙 到课

上周作业检查

上周作业链接: https://cppoj.kids123code.com/contest/846

2025-0921 周日10:30 (并查集)

刷新

#	用户名	姓名	编程分	时间	A	B	C	D	E	F
1	sunjingke	孙靖轲	600	11250	100	100	100	100	100	100
2	zhaoxiyu	赵熙羽	600	11394	100	100	100	100	100	100
3	yangjinshuo	杨谨硕	500	9444	100	100	100	100	100	
4	niutongze	牛同泽	430	6903	100	100	100	100	30	
5	yuzijia1	于子珈	400	4582	100	100	100	100		
6	xieyakai	谢亚锴	400	8337	100	100	100	100		
7	liuchuangsu	刘闯速	400	10852	100	100	100	100		
8	suitianyi	隋天乙	300	4248	100	100		100		

本周作业

https://cppoj.kids123code.com/contest/967 (课上讲了 A ~ C 题, 课后作业是 D 题)

课堂表现

今天的 A 题是一道裸的搜索题, 就是做两遍 搜索 找最远点, 同学们课上做 A 题整体都做的不太好, 课下要好好复习一下搜索的题目, 把之前的搜索题可以再做一做。

课堂内容

家谱 (上周作业)

并查集裸题, 多一步 要把字符串映射为整数, 把整数映射为字符串 的过程

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

using namespace std;

const int maxn = 5e4 + 5;
int f[maxn];

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

```
string str[maxn];

int main()
{
  int n = 0, id = 0;
  map<string, int> mp;
  map<int, string> mp2;
  while (true) {
    ++n;
    cin >> str[n];
    if (str[n] == "$") break;
    string s = str[n].substr(1);
    if (!mp.count(s)) {
      ++id; mp[s] = id; mp2[id] = s;
    }
  }

  for (int i = 1; i <= n-1; ++i) f[i] = i;

  int fa_id = 0;
  for (int i = 1; i <= n-1; ++i) {
    int id = mp[str[i].substr(1)];
    if (str[i][0] == '#') fa_id = id;
    else if (str[i][0] == '+') {
      int f1 = fFind(id), f2 = fFind(fa_id);
      f[f1] = f2;
    }
    else {
      int t = fFind(id);
      cout << mp2[id] << " " << mp2[t] << endl;
    }
  }
  return 0;
}
```

[蓝桥杯 2013 省 A] 大臣的旅费

两次搜索, 求树的带权直径

先从任意一点出发, 找到离他最远的点, 从这个点出发再搜一遍

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

using namespace std;

typedef long long LL;
const int maxn = 1e5 + 5;
struct node {
  int to, value;
};
```

```
vector<node> vec[maxn];
int dis[maxn];

void bfs(int x) {
  memset(dis, -1, sizeof(dis));

  queue<int> q; q.push(x); dis[x] = 0;
  while (!q.empty()) {
    int u = q.front(); q.pop();
    for (node it : vec[u]) {
      if (dis[it.to] != -1) continue;
      q.push(it.to), dis[it.to] = dis[u]+it.value;
    }
  }
}

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

    bfs(1);

    int id = 1;
    for (int i = 2; i <= n; ++i) {
    if (dis[i] > dis[id]) id = i;
    }

    bfs(id);

    int res = 0;
    for (int i = 1; i <= n; ++i) res = max(res, dis[i]);

    cout << res*10 + (LL)res*(res+1)/2 << endl;
  return 0;
}
```

[SCOI2005] 繁忙的都市

最少选边的数量是 n-1

边的最大值最小, 可以二分一个 mid, 把 <=mid 的边选中, 把这个边所连的两个点连通, 最后看是否所有点都连通了

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

using namespace std;
```

```cpp
const int maxn = 2e5 + 5;
int f[maxn];
struct node {
  int u, v, value;
  bool operator < (const node& p) const { return value < p.value; }
};
vector<node> vec;
int n, m;

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

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

  int cnt = n;
  for (node it : vec) {
    int u = it.u, v = it.v, value = it.value;
    if (value > mid) continue;

    int pu = fFind(u), pv = fFind(v);
    if (pu != pv) f[pu] = pv, --cnt;
  }

  return cnt==1;
}

int main()
{
  cin >> n >> m;
  while (m -- ) {
    int u, v, value; cin >> u >> v >> value;
    vec.push_back({u, v, value});
  }

  int l = 1, r = 10000;
  while (l <= r) {
    int mid = (l + r) / 2;
    if (check(mid)) r = mid-1;
    else l = mid+1;
  }
  cout << n-1 << " " << l << endl;
  return 0;
}
```

[CSP-J 2022] 上升点列

dp, f[i][j]: 考虑以第 i 个点结尾, 前面一共添加了 j 个点时, 能得到的序列的最大长度是多少

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

using namespace std;

const int maxn = 500 + 5;
struct node {
  int x, y;
  bool operator < (const node& p) const {
    if (x != p.x) return x < p.x;
    return y < p.y;
  }
} w[maxn];
int f[maxn][maxn];

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

  for (int i = 1; i <= n; ++i) {
    for (int j = 0; j <= m; ++j) {
      f[i][j] = j+1;
      for (int k = 1; k < i; ++k) {
        if (w[i].y < w[k].y) continue;

        int c = w[i].x-w[k].x + w[i].y-w[k].y - 1;
        if (j-c >= 0) f[i][j] = max(f[i][j], f[k][j-c]+c+1);
      }
    }
  }

  int res = 0;
  for (int i = 1; i <= n; ++i) res = max(res, f[i][m]);
  cout << res << endl;
  return 0;
}
```