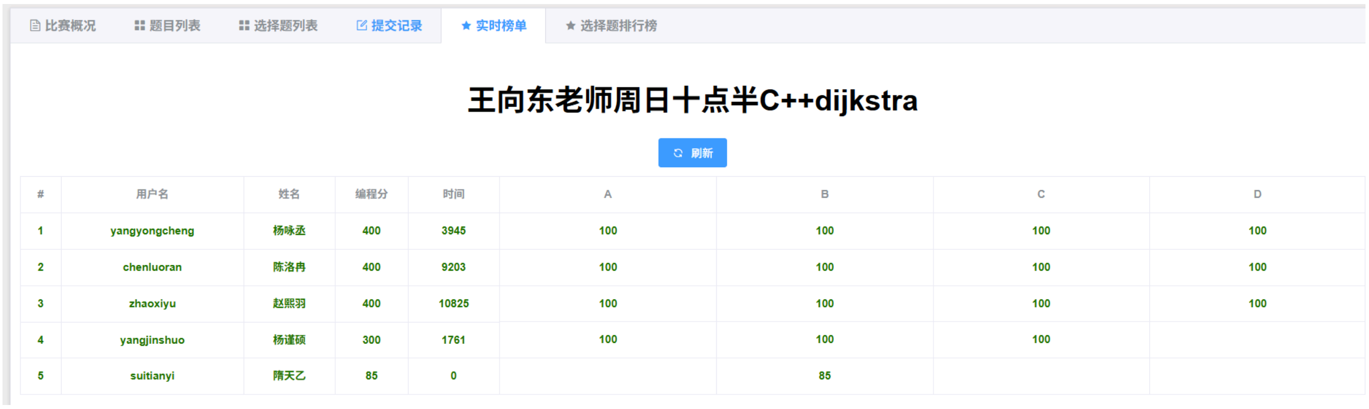
kruskal

人员

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

上周作业检查

上周作业链接: https://cppoj.kids123code.com/contest/1341



王向东老师周日十点半C++dijkstra

#	用户名	姓名	编程分	时间	A	B	C	D
1	yangyongcheng	杨咏丞	400	3945	100	100	100	100
2	chenluoran	陈洛冉	400	9203	100	100	100	100
3	zhaoxiyu	赵熙羽	400	10825	100	100	100	100
4	yangjinshuo	杨谨硕	300	1761	100	100	100	
5	suitianyi	隋天乙	85	0		85		

本周作业

https://cppoj.kids123code.com/contest/1478 (课上讲了 A B C E 题, 课后作业是 D 题必做, E 题选做)

课堂表现

今天讲了 kruskal 和 快速幂 这两个知识

在今天上课中, 反映出许多同学之前的并查集基础很不扎实, 同学们课下需要复习一下之前的并查集代码, 把并查集代码背熟

课堂内容

Shortest Path 3 (上周作业)

裸 dijkstra 模板题

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

using namespace std;

const int maxn = 2e5 + 5;
struct node {
  int dis, id;
  bool operator < (const node& p) const { return dis < p.dis; }
  bool operator > (const node& p) const { return dis > p.dis; }
};
vector<node> vec[maxn];
int w[maxn], f[maxn];
bool st[maxn];

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

  memset(f, 0x3f, sizeof(f));
  priority_queue<node, vector<node>, greater<node>> q;
  q.push({w[1],1}); f[1] = 0;
  while (!q.empty()) {
    node u = q.top(); q.pop();
    int dis = u.dis, id = u.id;
    if (st[id]) continue;

    st[id] = true;
    for (node it : vec[id]) {
      if (st[it.id]) continue;
      if (dis+it.dis+w[it.id] < f[it.id]) {
        f[it.id] = dis+it.dis+w[it.id]; q.push({f[it.id],it.id});
      }
    }
  }

  for (int i = 2; i <= n; ++i) cout << f[i] << " ";
  cout << endl;
  return 0;
}
```

【模板】最小生成树

kruskal 模板题

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

using namespace std;

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

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

void kruskal(vector<node>& vec, int n) {
  sort(vec.begin(), vec.end());
  for (int i = 1; i <= n; ++i) f[i] = i;

  int sum = 0, cnt = 0;
  for (node it : vec) {
    int u = it.u, v = it.v, value = it.value;
    int pu = fFind(u), pv = fFind(v);
    if (pu == pv) continue;
    f[pu] = pv; sum += value; ++cnt;
  }
  if (cnt != n-1) cout << "orz" << endl;
  else cout << sum << endl;
}

int main()
{
  int n, m; cin >> n >> m;
  vector<node> vec;
  while (m -- ) {
    int u, v, value; cin >> u >> v >> value;
    vec.push_back({u, v, value});
  }
  kruskal(vec, n);
  return 0;
}
```

[SCOI2005] 繁忙的都市

之前做过, 第一次做的方法是用 二分 + 并查集 做的, 第二个方法是直接构建最小生成树即可

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

using namespace std;

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

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

```
void solve(vector<node>& vec, int n) {
  sort(vec.begin(), vec.end());
  for (int i = 1; i <= n; ++i) f[i] = i;

  int cnt = 0;
  for (node it : vec) {
    int u = it.u, v = it.v, value = it.value;
    int pu = fFind(u), pv = fFind(v);
    if (pu == pv) continue;
    f[pu] = pv; ++cnt;
    if (cnt == n-1) { cout << n-1 << " " << value << endl; break; }
  }
}

int main()
{
  int n, m; cin >> n >> m;
  vector<node> vec;
  while (m -- ) {
    int u, v, value; cin >> u >> v >> value;
    vec.push_back({u, v, value});
  }
  solve(vec, n);
  return 0;
}
```

Choose Two and Eat One

可以先 O(n^2) 求出任意两点间的得分, 然后构建最大生成树即可

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

using namespace std;

const int maxn = 500 + 5;
struct node {
  int u, v, value;
  bool operator < (const node& p) const { return value < p.value; }
};
int w[maxn], f[maxn];

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

int qmod(int a, int k, int mod) {
  int res = 1;
  while (k) {
    if (k&1) res = res*a % mod;
    a = a*a % mod;
    k >>= 1;
  }
  return res;
}

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

  vector<node> edges;
  for (int i = 1; i <= n; ++i) {
    for (int j = 1; j <= n; ++j) {
      if (i != j) {
        int v = (qmod(w[i],w[j],mod) + qmod(w[j],w[i],mod)) % mod;
        edges.push_back({i, j, v});
      }
    }
  }

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

  int res = 0;
  for (node it : edges) {
    int u = it.u, v = it.v, value = it.value;
    int fu = fFind(u), fv = fFind(v);
    if (fu != fv) f[fu] = fv, res += value;
  }
  cout << res << endl;
  return 0;
}
```