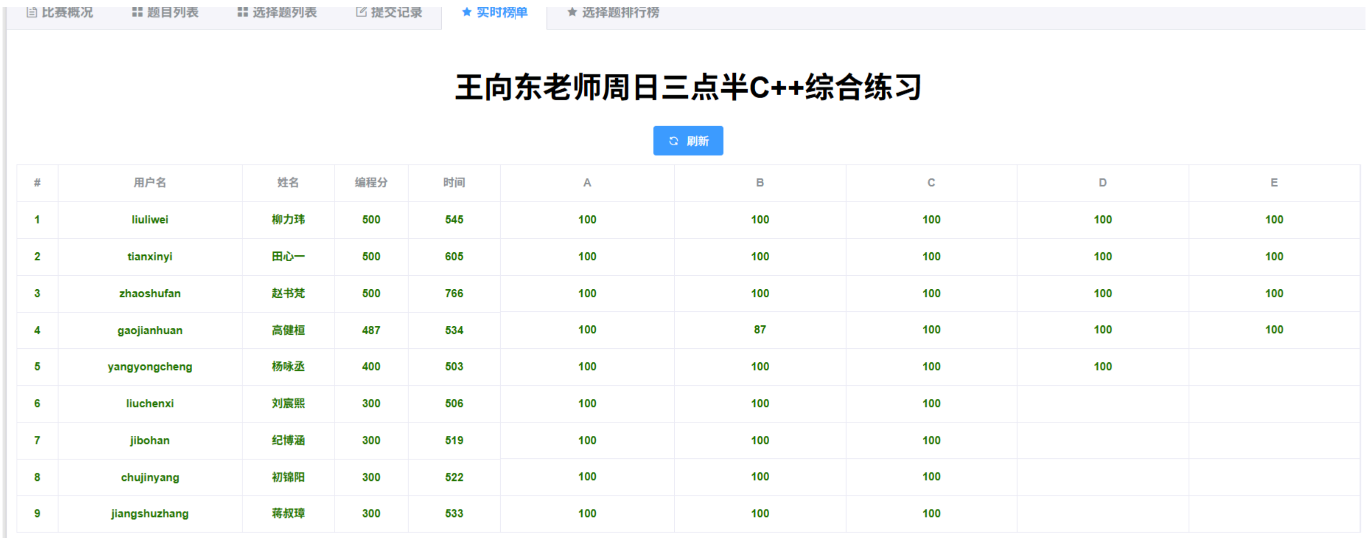
存图搜图

人员

田心一、纪博涵、蒋叔璋、初锦阳、刘宸熙、赵书梵、柳力玮、李瑞涵、杨咏丞 到课, 苑钊 线上

上周作业检查

上周作业链接: https://cppoj.kids123code.com/contest/2241



王向东老师周日三点半C++综合练习

#	用户名	姓名	编程分	时间	A	B	C	D	E
1	liuliwei	柳力玮	500	545	100	100	100	100	100
2	tianxinyi	田心一	500	605	100	100	100	100	100
3	zhaoshufan	赵书梵	500	766	100	100	100	100	100
4	gaojianhuan	高健桓	487	534	100	87	100	100	100
5	yangyongcheng	杨咏丞	400	503	100	100	100	100	
6	liuchenxi	刘宸熙	300	506	100	100	100		
7	jibohan	纪博涵	300	519	100	100	100		
8	chujinyang	初锦阳	300	522	100	100	100		
9	jiangshuzhang	蒋叔璋	300	533	100	100	100		

本周作业

https://cppoj.kids123code.com/contest/2332 (课上讲了 A ~ E 题, 课后作业是 F 题)

课堂表现

今天着重讲了建图搜图的内容, 邻接表是非常非常重要的内容, 同学们课下一定要好好复习总结一下。

课堂内容

[NOIP 2001 普及组] 求先序排列 (上周作业)

中序 + 后续 求前序

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

using namespace std;

const int maxn = 26 + 5;
char a[maxn], b[maxn];

int w_find(int l, int r, char x) {
  for (int i = l; i <= r; ++i) {
    if (a[i] == x) return i;
  }
  return 0;
}

void dfs(int l1, int r1, int l2, int r2) {
  if (l1 > r1) return;
  if (l1 == r1) { cout << a[l1]; return; }

  char x = b[r2];
  int pos = w_find(l1, r1, x);
  int l_len = pos - l1;
  cout << x;
  dfs(l1, pos-1, l2, l2+l_len-1);
  dfs(pos+1, r1, l2+l_len, r2-1);
}

int main()
{
  cin >> (a+1) >> (b+1);
  dfs(1, strlen(a+1), 1, strlen(b+1));
  cout << endl;
  return 0;
}
```

图的存储

邻接矩阵 + 邻接表 存图

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

using namespace std;

const int maxn = 1000 + 5;
int f[maxn][maxn];
vector<int> vec[maxn];

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

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            cout << f[i][j] << " ";
        }
        cout << endl;
    }
    for (int i = 1; i <= n; i++) {
        cout << vec[i].size() << " ";
        sort(vec[i].begin(), vec[i].end());
        for (int j : vec[i]) {
            cout << j << " ";
        }
        cout << endl;
    }
    return 0;
}
```

Adjacency List

邻接表存图

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

using namespace std;

const int maxn = 1000 + 5;
vector<int> vec[maxn];

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

    for (int i = 1; i <= n; i++) {
        cout << vec[i].size() << " ";
        sort(vec[i].begin(), vec[i].end());
        for (int j : vec[i]) {
            cout << j << " ";
        }
        cout << endl;
    }
    return 0;
}
```

图的存储与出边的排序

邻接表存图

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

using namespace std;
```

```
const int maxn = 5e5 + 5;
vector<int> vec[maxn];

void solve() {
    int n, m; cin >> n >> m;
    for (int i = 1; i <= n; i++) vec[i].clear();

    while (m -- ) {
        int u, v; cin >> u >> v;
        vec[u].push_back(v);
    }

    for (int i = 1; i <= n; i++) {
        sort(vec[i].begin(), vec[i].end());
        for (int j : vec[i]) cout << j << " ";
        cout << endl;
    }
}

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

图的遍历（简单版）

dfs 搜图, 以 每个点为起点, 进行 dfs 的搜索, 时间复杂度 O(n^2)

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

using namespace std;

const int maxn = 1000 + 5;
vector<int> vec[maxn];
int res;
bool vis[maxn];

void dfs(int u) {
    if (vis[u]) return;
    vis[u] = true;

    res = max(res, u);
    for (int i : vec[u]) dfs(i);
}

int main()
{
    int n, m; cin >> n >> m;
    while (m -- ) {
        int a, b; cin >> a >> b;
        vec[a].push_back(b);
    }

    for (int i = 1; i <= n; i++) {
        res = i;
        memset(vis, 0, sizeof(vis));
        dfs(i);
        cout << res << " ";
    }
    return 0;
}
```

Tour

跟上道题一样, n^2 看每个点能搜到多少个点

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

using namespace std;

const int maxn = 2000 + 5;
vector<int> vec[maxn];
bool vis[maxn];

void dfs(int u) {
    if (vis[u]) return;
    vis[u] = true;
    for (int i : vec[u]) dfs(i);
}

int main()
{
    int n, m; cin >> n >> m;
    while (m -- ) {
        int a, b; cin >> a >> b;
        vec[a].push_back(b);
    }

    int res = 0;
    for (int i = 1; i <= n; ++i) {
        memset(vis, 0, sizeof(vis));
        dfs(i);
        for (int j = 1; j <= n; ++j) {
            if (vis[j]) ++res;
        }
    }
    cout << res << endl;
    return 0;
}
```