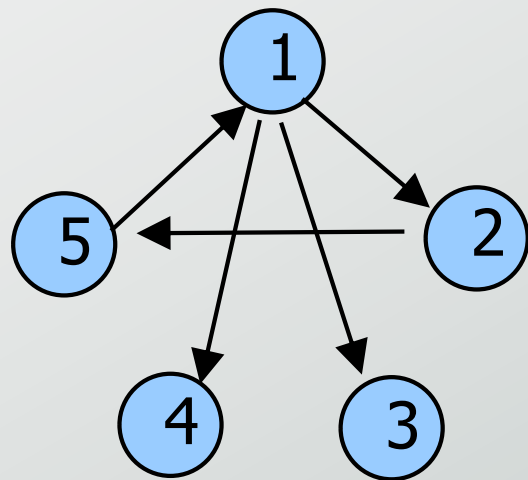


第五章 数的存储与组织

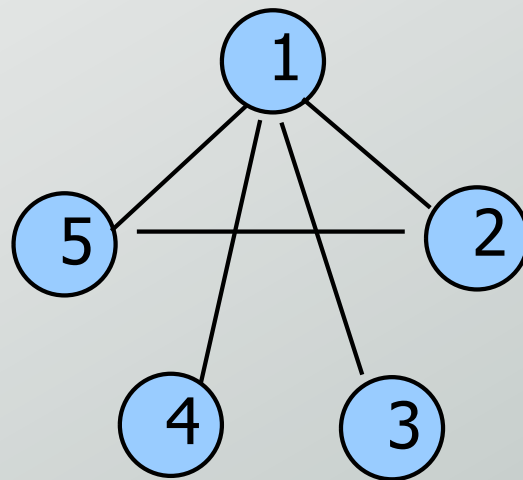
第 6 6 课 图的存储方式

《信息学奥赛一本通·编程启蒙 C++版》

图(Graph)是一种复杂的非线性结构。在人工智能、工程、数学、物理、化学、生物和计算机科学等领域中，图结构有着广泛的应用。



(a)有向图



(b)无向图

一、二维数组邻接矩阵存储

图的邻接矩阵存储方式是用一个二维数组（称邻接矩阵）存储图中的边或弧的信息。它适用于稠密图。

定义 `int G[101][101]`，其中 `G[i][j]` 的值，表示从点 `i` 到点 `j` 的边的权值，定义如下：

$$G[i][j] = \begin{cases} 1 \text{ 或权值} & \text{当 } v_i \text{ 与 } v_j \text{ 之间有边或弧时, 取值为 } 1 \text{ 或权值} \\ 0 \text{ 或 } \infty & \text{当 } v_i \text{ 与 } v_j \text{ 之间无边或弧时, 取值为 } 0 \text{ 或 } \infty \text{ (无穷大)} \end{cases}$$

二、邻接表存储结构

邻接表是一种将数组与链表相结合的存储方法，其具体实现为：将图中顶点用一个一维数组存储，每个顶点 V_i 的所有邻接点用一个单链表来存储，链表中存放与当前节点相邻的节点在数组中的下标。它适用于稀疏图。

用 STL 库的 vector(向量) 存邻接表相对简单。

```
1. vector<int> linker[MAXN];  
2. for(int i=1;i<=edge;++i){  
3.     int u,v;  
4.     scanf("%d%d",&u,&v);  
5.     linker[u].push_back(v); //存边  
6. }
```

【例 66.1】 图论入门

【题目描述】

假设用一个 $n \times n$ 的数组 a 来描述一个有向图的邻接矩阵：

- (1) 编写一个函数确定一个顶点的出度
- (2) 编写一个函数确定一个顶点的入度
- (3) 编写一个函数确定图中边的数目。

【输入格式】

第一行：节点总数 n ，指定节点 m ，节点从 **1** 开始编号。

下面 n 行：有向图的邻接矩阵。

$$1 \leq n, m, a[i][j] \leq 1000$$

【输出格式】

第一行输出三个数：节点编号 m ， m 的出度， m 的入度（之间用一个空格隔开）。

第二行输出：图中边的总数。

【样例输入】

```
5 3
0 4 2 2 3
2 0 1 5 10
2 0 0 4 0
0 3 7 0 7
6 2 0 0 0
```

【样例输出】

```
3 2 3
15
```

【代码实现】

```
1. #include <bits/stdc++.h>
2. using namespace std;
3. int G[1005][1005];
4. int main(){
5.     int n, m, ans = 0, rd = 0, cd = 0;
6.     cin >> n >> m;
7.     for (int i = 0; i < n; i++)
8.         for (int j = 0; j < n; j++) {
9.             cin >> G[i][j];
10.            if (G[i][j] != 0) {
11.                ans++;
12.                if (i == m - 1) cd++;
13.                if (j == m - 1) rd++;
14.            }
15.        }
16.    cout << m << " " << cd << " " << rd << endl << ans;
17.    return 0;
18.}
```

【例 66.2】 朋友关系

【题目描述】

同学们应该学会多交一些好朋友。朋友关系是相互的，A 是 B 的好朋友，则 B 也是 A 的好朋友。朋友关系是不传递的，A 是 B 的好朋友，B 是 C 的好朋友，但 A 和 C 不一定是好朋友。现在给出某小学部分同学之间的朋友关系，请编程统计朋友最多的人有多少个好朋友。

【输入格式】

输入从文件中读取，输入共 $m+1$ 行。

第 1 行是两个整数 n 和 m ，分别表示同学总人数和朋友关系对数。

第 2 行到第 $m+1$ 行，描述了 m 对朋友关系。每行两个用单个空格隔开的同学姓名，保证两个姓名不同。

每个人的姓名仅由小写字母组成，且 $1 \leq \text{姓名的长度} \leq 10$

【输出格式】

结果输出到文件中，输出共 1 行。

一个整数，表示朋友最多的人有多少个好朋友。

【样例输入】

4 3

lucy lily

jam lily

jam peter

【样例输出】

2

【代码实现】

```
1. #include <bits/stdc++.h>
2. using namespace std;
3. constexpr int N = 105;
4. int n, m;
5. bool exist[N][N];
6. int deg[N];
7. unordered_map<string, int> id; //
8. int tot;
9. signed main() {
10.     cin >> n >> m;
11.     for (int i = 1; i <= m; ++i) {
12.         string x, y;
13.         cin >> x >> y; // 需要将每一个串转换成一个固定的编号
14.         int u = id[x] ? id[x] : id[x] = ++tot,
15.         v = id[y] ? id[y] : id[y] = ++tot; // 值为0 要赋一编号
16.         if (u > v)
17.             swap(u, v);
18.         if (exist[u][v]) // 需要没出现这关系才能标记
19.             continue;
20.         exist[u][v] = true;
21.         ++deg[u], ++deg[v]; // 给两个节点都加上贡献
22.     }
23.     int res = 0;
24.     for (int i = 1; i <= n; ++i)
25.         if (res < deg[i])
26.             res = deg[i]; // 找到好朋友最多的人
27.     cout << res << endl;
28.     return 0;
29. }
```

练 66.1 黑马篮球队

【题目描述】

三人篮球起源于美国街头黑人孩子的三对三斗牛赛，是一项孩子们创造的竞技活动。目前有 n 个球员，其中有 m 对相互认识的关系。一个球员的知名度定义为 有多少人 和他相互认识。

现在教练需要从这 n 个球员中选出三个人成为一支黑马球队参加比赛，需要球队三人相互认识，但是教练又希望他们三个人的知名度总和加起来最低，到时给大家来一个“一鸣惊人”。

【输入格式】

第一行为两个整数 n, m ，都不大于 4000。

往后 m 行每一行输入两个不超过 n 的整数 $x, y (x < y)$ ，表示第 x, y 两人互相认识（数据中有可能会有重复的认识关系）。

对于 50% 的数据，满足 $n, m \leq 200$ 。

对于 100% 的数据，满足 $n, m \leq 40000$

【输出格式】

如果能找到三个相互认识的人，你需要输出所有选择中知名度总和的最低值 否则输出 **-1**。

第二行包括一个数据：图中边的总数。

【样例输入】

5 6

1 2

1 3

2 3

2 4

3 4

4 5

【样例输出】

8

【代码实现】

```
1. #include<bits/stdc++.h>
2. using namespace std;
3. int v[10000],n,m,ans=120000,x,y;
4. bool vis[4005][4005],f;
5. pair<int,int> a[4005];
6. int main(){
7.     cin>>n>>m;
8.     for(int i=1;i<=m;i++){
9.         cin>>x>>y;
10.        vis[x][y]=1;vis[y][x]=1;
11.        a[i].first=x;a[i].second=y;
12.    }
13.    for(int i=1;i<=n;i++)
14.        for(int j=1;j<=n;j++)
15.            if(vis[i][j]) v[i]++;
16.    for(int i=1;i<=m;i++){
17.        x=a[i].first;y=a[i].second;
18.        for(int j=1;j<=n;j++){
19.            if(vis[x][j] && vis[y][j]){
20.                f=1;
21.                ans=min(ans,v[x]+v[y]+v[j]);
22.            }
23.        }
24.    }
25.    if(!f) cout<<-1; else cout<<ans;
26.    return 0;
27.}
```

谢谢！

