蛟龙五班

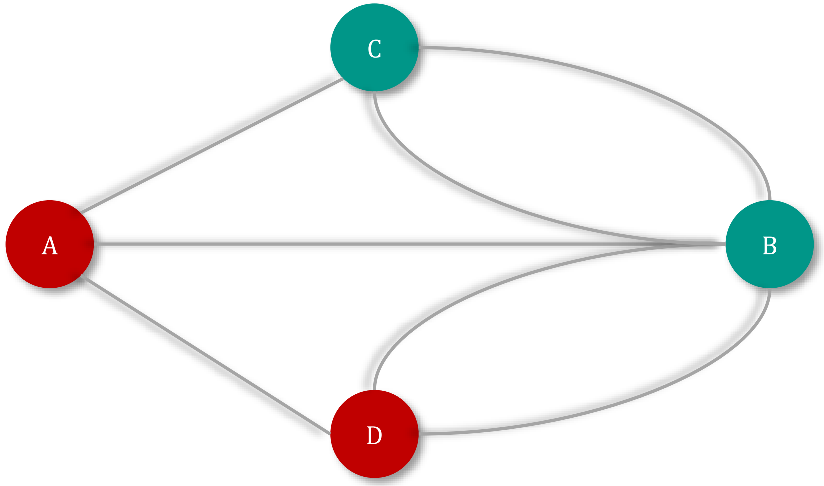
图

Mas

欧拉图

1736年,29岁的数学家欧拉来到普鲁士的古城哥尼斯堡。普瑞格尔河从市中心流过,河中心有两座小岛,岛和两岸之间建筑有七座古桥

欧拉发现当地居民有一项消遣活动,就是试图每座桥恰好走过一遍并回到原出发点,但从来没人成功过



若每座桥都恰好走过一次,对于每一个顶点,需要从某条边进入,同时从另一条边离开

进入和离开顶点的次数是相同的,每个顶点相连的边是成对出现的,即每个顶点的相连边的数量必须是偶数

上图中A、C、D四个顶点的相连边都是3,顶点B的相连边为5为奇数

因此无法从一个顶点出发,遍历每条边各一次。证明了这种走法是不可能存在的

现在看来,欧拉的证明过程非常简单,但他对七桥问题的抽象和论证思想,开创了一个新的学科：图论

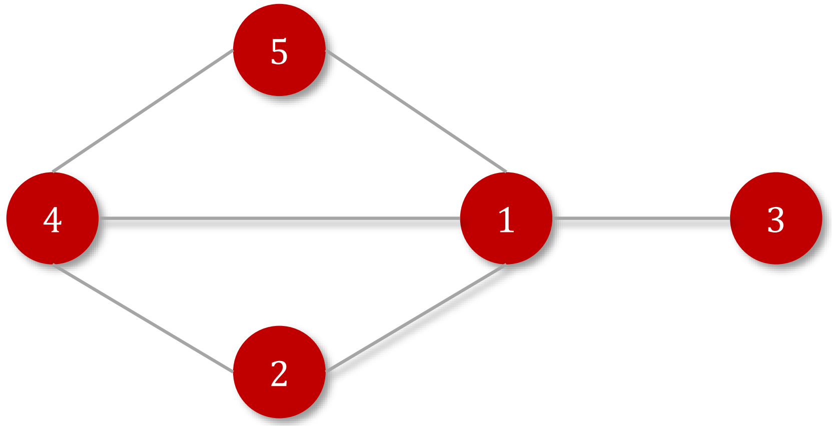
图

图 (graph$h$) 是一个二元组 $G = \big(V(G), E(G)\big)$

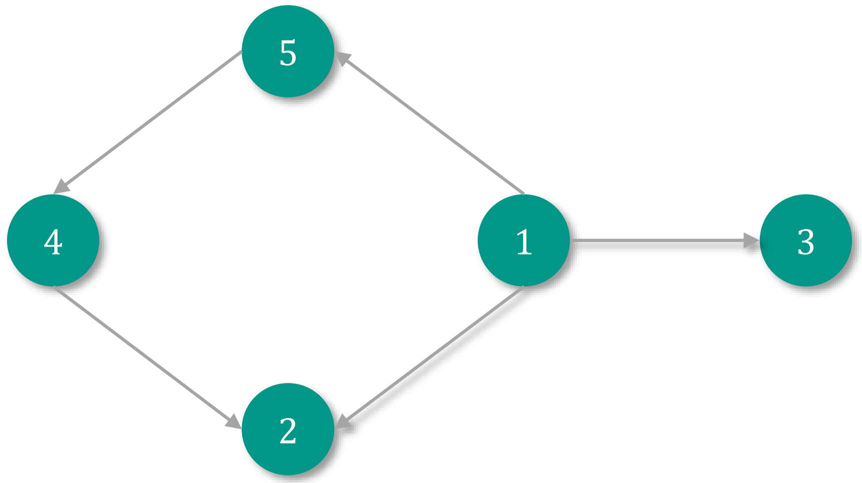
其中 $V(G)$ 是非空集,称为点集 (vertex set),对于V中的每个元素,称其为顶点(vertex)或节点(node),简称点

$E(G)$为$V(G)$各结点之间边的集合,称为边集 (edge set)

常用 $G = (V, E)$表示图



无向图:图的边没有方向



有向图:图的边有方向,只能按箭头方向从一点到另一点

度

与一个顶点v关联的边的条数称作该顶点的**度** (degree),记作 $d(v)$

图论基本定理

对于任何无向图 $G = (V,E)$,有

$$\sum_{v \in V} d(v) = 2\,|E|$$

推论：在任意图中,度数为奇数的点必然有偶数个

有向图 $G = (V,E)$中

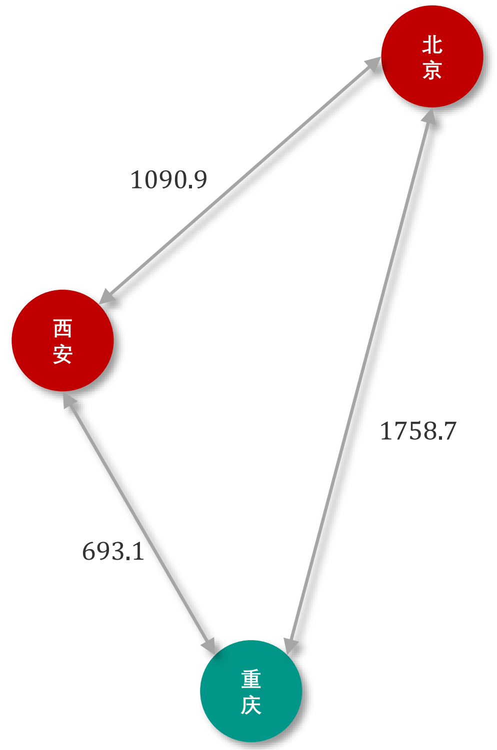
以一个顶点 v 为起点的边的条数称为该顶点的**出度** (out − degree),记作 $d^{+}(v)$

以一个顶点 v 为终点的边的条数称为该节点的**入度** (in − degree),记作 $d^{-}(v)$

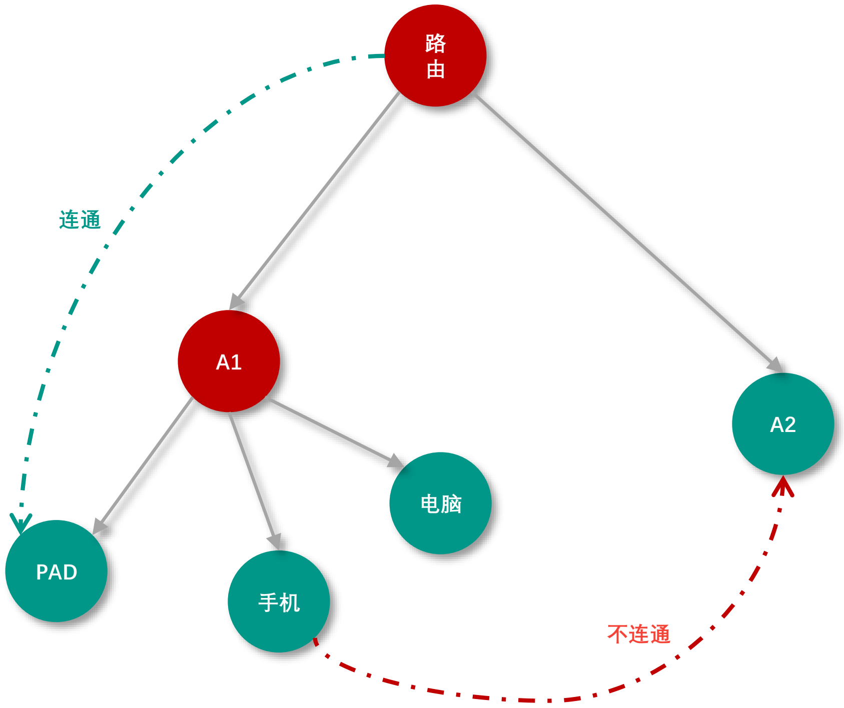
显然

$$d^{+}(v) + d^{-}(v) = d(v)$$

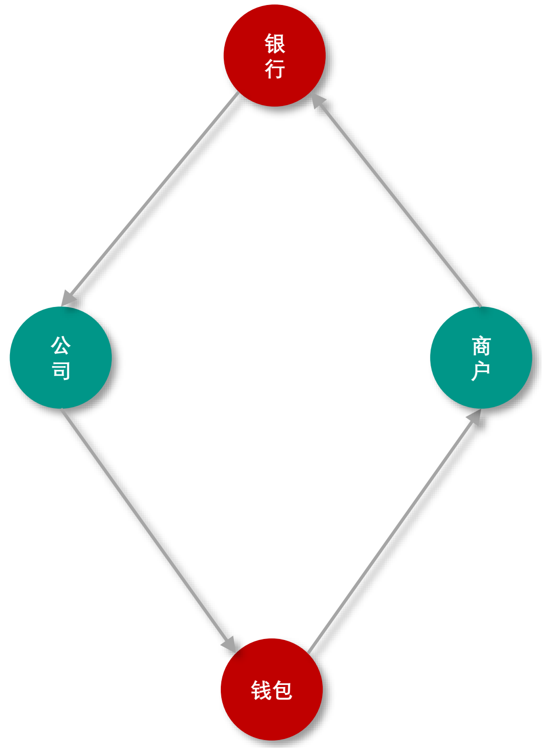
边权、连通、环



边的权值,可以形象理解为边的长度

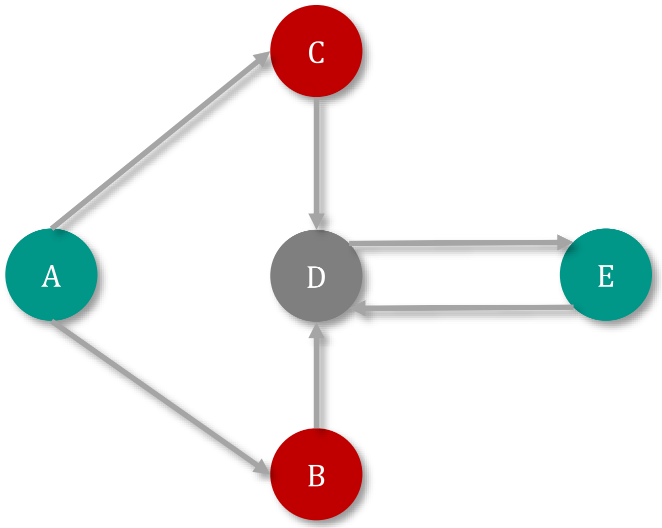


图中结点U,V之间存在一条从U通过若边到达V的通路,则称U,V连通



起点终点相同的路径,称为回路或环

图的存储-邻接矩阵



	A	B	C	D	E
A	-	1	1	0	0
B	0	-	0	1	0
C	0	0	-	1	0
D	0	0	0	-	1
E	0	0	0	1	-

邻接矩阵是一种简单的存储方式,用一个 $n \times n$ 的二维数组 g 存储边的信息

其中$g[u][v]$ 表示点 u 到点 v 之间的边的情况

如果边没有权值,则用 0/1 表示(0表示没有边,1表示有边)

如果边有权值,则用极大值表示没有边,普通数值表示边的权值

#1875 送温暖

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 。

下面 n 行：有向图的邻接矩阵，相邻两数之间以一个空格分隔。

输出格式

第一行包括三个数据：节点编号 m ， m 的出度， m 的入度（之间用一个空格隔开）。

第二行包括一个数据：图中边的总数。

数据范围

对于全部的数据 $1 \leq n, m, a_{ij} \leq 1000$

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
```

图的存储-邻接矩阵

优点：

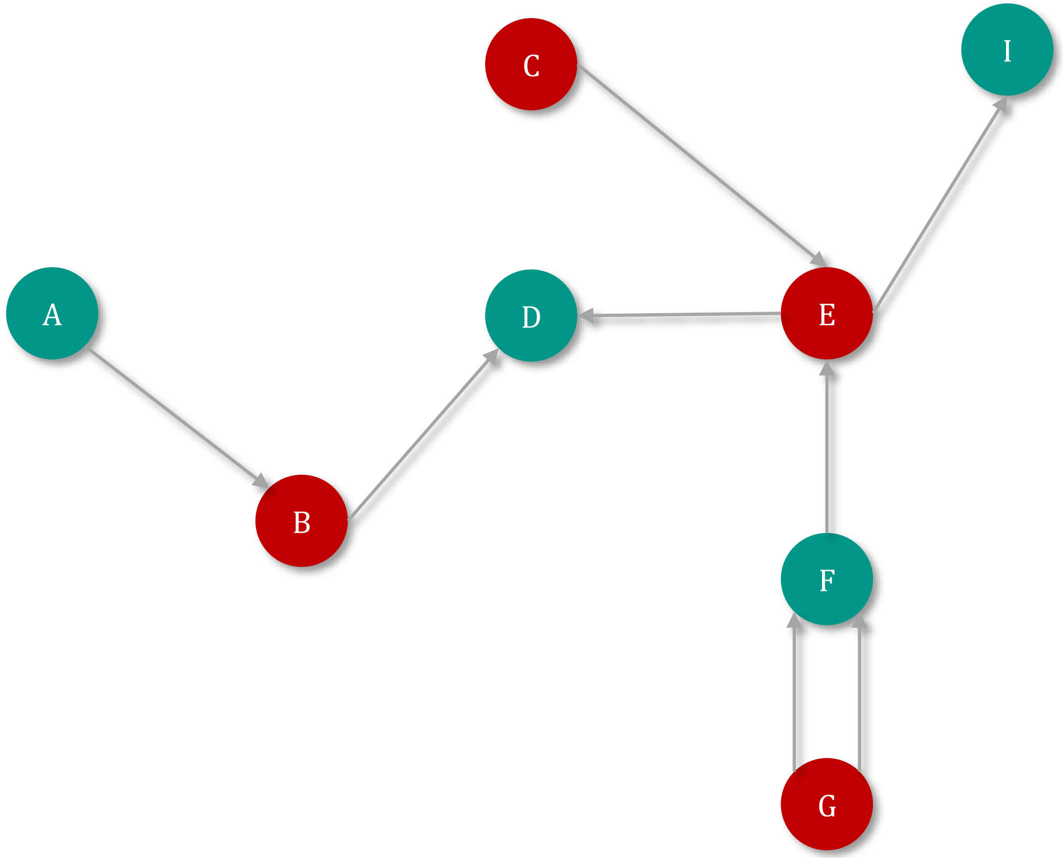
存储和遍历都简单便捷

缺点：

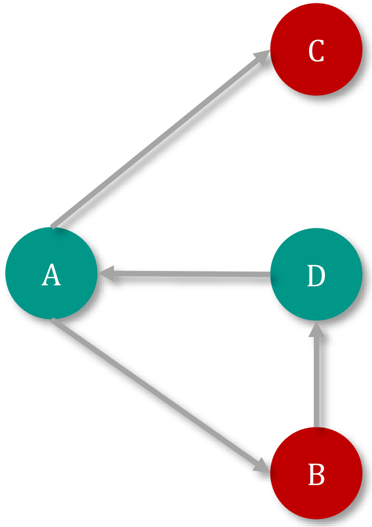
对于**稀疏图** 比较浪费存储空间，空间复杂度 $O(n^2)$

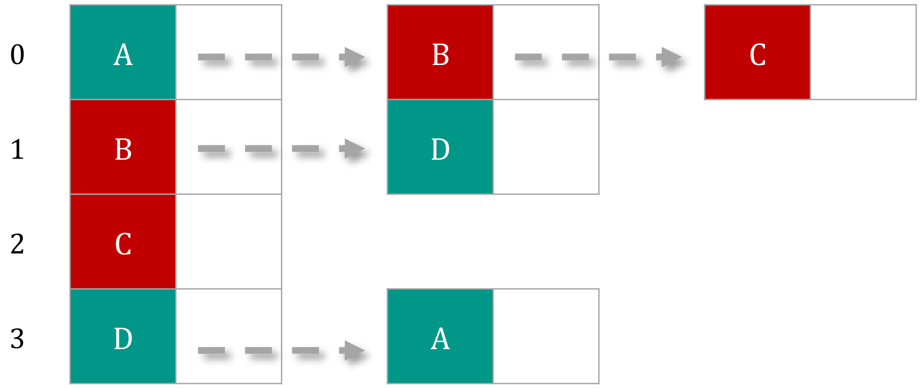
找点 u 出发的每一条边的复杂度较高，时间复杂度 $O(n)$

不好处理重边的情况



图的存储-邻接表





邻接表是一个二维容器

第一维描述点,第二维描述点所对应的边集

$g[u]$ 是一个容器里面存储着所有起点为u的边

#1027、又是图论送温暖

题目描述

小明同学跟随惊奇队长去往了宇宙的另一端建造新的家园。现在家园里已经修建好 n 个房子，标号从 $1 \sim n$ ，他们一共修建了 m 条边，每条边都是从一个房子通向另一个房子，互相不影响。

现在有一个任务，需要您计算每个房子所连接的边的个数。

输入格式

第一行输入 n, m

接下来 m 行，每行两个数 a, b

表示 a 有一条边连向 b

> *注意，这里可能存在重边*

保证 $1 \leq n, m \leq 100000$

输出格式

输出 n 行数

第 i 行输出房子 i 所连接的边的边数量

输入样例

```
3 3
1 2
2 3
1 2
```

输出样例

```
2
3
1
```

链式前向星

维护 n 个单向链表

对于条边 (u,v), 将其插入第u个链表的尾部

对于 $1 \sim n$ 这 n 个链表,只维护它们的尾结点编号

初始时所有尾结点都是空的

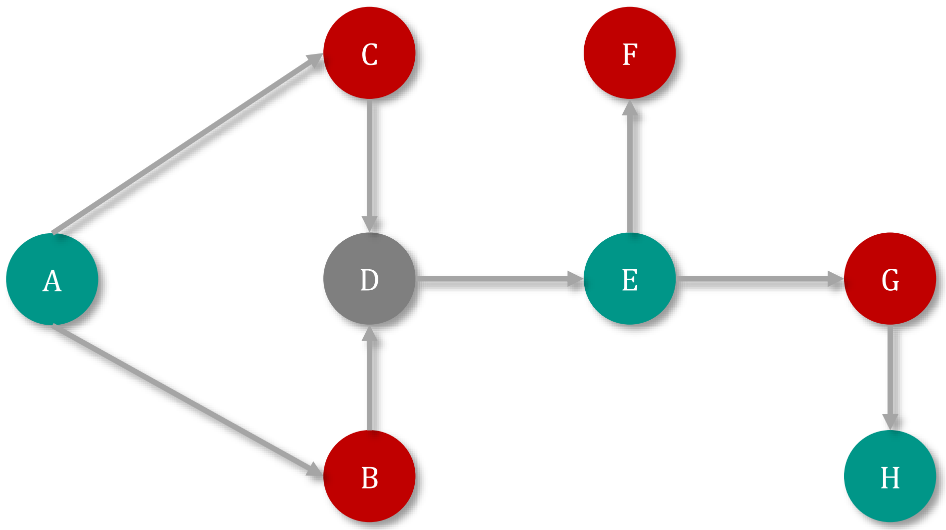
每当插入一条边 (u,v) 时,为其分配一个新的节点标号 cnt,把它的val置为 y

想象一下每个链表的结构:

当想遍历第 u 个链表时,从u的尾部$tail[x]$开始,不断跳nxt跳到上一个节点,直到为 0

```cpp
struct Edge
{
    int u, v, w, next;
} e[100001];
int n, m, k, head[100001], pos, u, v, w;
void addEdge(int u, int v, int w)
{
    e[++pos] = {u, v, w, head[u]};
    head[u] = pos;
}
```

图的遍历



深度优先遍历

选定一个方向每次都尝试向更深的节点走

从A出发进行深度优先遍历的结果为

$$ABDEGHFC$$

广度优先遍历

每次都尝试访问同一层的节点, 如果同一层都访问完了,再访问下一层

从A出发进行广度优先遍历的结果为

$$ABCDEFGH$$

#2117 图的深度优先遍历

Description

请定一个无向图，顶点编号从 $0 \sim n-1$，用深度优先搜索(DFS)，遍历并输出。遍历时，先遍历节点编号小的。

Input

输入第一行是两个整数 k, m （$0 < k \leq 100$，$0 < m \leq k \times k$），表示有 m 条边，k 个顶点。

下面的 m 行，每行是空格隔开的两个整数 u，v，表示一条连接 u，v 顶点的无向边。

Output

输出有 1 行，表示 DFS 的遍历结果。

输入样例

```
4 4
0 1
0 2
0 3
2 3
```

输出样例

```
0 1 2 3
```

```cpp
#include <bits/stdc++.h>
using namespace std;
int n, m, u, v, g[101][101];
bool vis[101];
void dfs(int u)
{
    vis[u] = true;
    cout << u << " ";
    for (int v = 1; v <= n; v++)
        if (g[u][v] && !vis[v])
            dfs(v);
}
int main()
{
    cin >> n >> m;
    for (int i = 0; i < m; i++)
    {
        cin >> u >> v;
        g[u][v] = 1;
        g[v][u] = 1;
    }
    dfs(0);
    return 0;
}
```

#2117 图的广度优先遍历

Description

请定一个无向图，顶点编号从 $0 \sim n-1$，用广度优先搜索(BFS)，遍历并输出。遍历时，先遍历节点编号小的。

Input

输入第一行是两个整数 k, m （$0 < k \leq 100$，$0 < m \leq k \times k$），表示有 m 条边，k 个顶点。

下面的 m 行，每行是空格隔开的两个整数 u，v，表示一条连接 u，v 顶点的无向边。

Output

输出有 1 行，表示 BFS 的遍历结果。

输入样例

```
4 4
0 1
0 2
0 3
2 3
```

输出样例

```
0 1 2 3
```

```cpp
#include <bits/stdc++.h>
using namespace std;
int t, n, m, u, v, g[101][101];
bool vis[101];
void bfs()
{
  queue<int> q;
  vis[0] = true;
  q.push(0);
  while (!q.empty())
  {
    u = q.front();
    q.pop();
    cout << u << " ";
    for (int v = 0; v < n; v++)
      if (g[u][v] && !vis[v])
        vis[v] = true, q.push(v);
  }
}
int main()
{
  cin >> n >> m;
  for (int i = 0; i < m; i++)
  {
    cin >> u >> v;
    g[u][v] = g[v][u] = 1;
  }
  bfs();
  return 0;
}
```

#20、图的联通块

问题描述

求无向图求这幅图中的联通块的个数

输入说明

第一行是两个整数 n, m，表示有 n 个点和 m 条边。

接下来 m 行，每行两个整数：u，v，分表表示这条边连接的两个点 u 和 v $(0 < u \leq v \leq n)$

输出说明

输出一个整数，表示联通块的个数

数据规模

对于 100% 的数据：$0 < n \leq 100000, 0 < m \leq 200000$

输入样例

```
4 3
1 2
1 3
2 3
```

输出样例

```
2
```

#1883、母牛野餐

题目描述

$k(1 \le k \le 100)$ 只奶牛分散在 $n(1 \le n \le 1000)$ 个牧场。

现在它们要集中起来进餐。

牧场之间有 $m(1 \le m \le 10000)$ 条有向路连接，而且不存在起点和终点相同的有向路。

它们进餐的地点(牧场)必须是所有奶牛都可到达的地方．那么，有多少这样的牧场呢?

输入格式

第一行共三个整数 k, n, m

接下来 k 行，每行一个整数表示一只奶牛所在的牧场编号

接下来 m 行，每行两个整数，表示一条有向路的起点和终点

输出格式

所有奶牛都可到达的牧场个数

样例解释

牧场 3， 4 是这样的牧场.

输入样例

```
2 4 4
2
3
1 2
1 4
2 3
3 4
```

输出样例

```
2
```

设 cnt_u 为能到达点u的牛的数量

建图后,每头牛作为起点,进行 DFS/BFS 遍历

对于每次能到达的点 v 令 cnt_v 递增1

答案为 $cnt_v = k$ 的点个数

时间复杂度 $O(k(n+m))$

#2299、碎碎念

题目描述

以前有个孩子，他分分钟都在碎碎念。不过，他的念头之间是有因果关系的。
他会在本子里记录每一个念头，并用箭头画出这个念头的来源于之前的哪一个念头。
翻开这个本子，你一定会被互相穿梭的箭头给搅晕，现在他希望你用程序计算出这些念头中最长的一条因果链。
将念头从 1 到 n 编号，念头 i 来源于念头 $from_i$，保证 $from_i < i$， $from_i = 0$ 表示该念头没有来源念头，只是脑袋一抽，灵光一现。

输入

第一行一个正整数 n 表示念头的数量
接下来 n 行依次给出 $from_1, from_2, \ldots, from_n$

输出

共一行，一个正整数 L 表示最长的念头因果链中的念头数量

数据规模和约定

对于全部的数据 $1 \leq n \leq 1000$

观察到$from_i < i$

那么每条边的起点都已经确定

令d_i为以i结尾的因果链的长度

显然$d_{from_1} = 0$

只需要令

$$d_i = d_{from_i} + 1$$

答案为$\max(d_i)$

欧拉图

欧拉通路

通过图中所有边恰好一次且行遍所有顶点的通路称为欧拉通路

欧拉回路

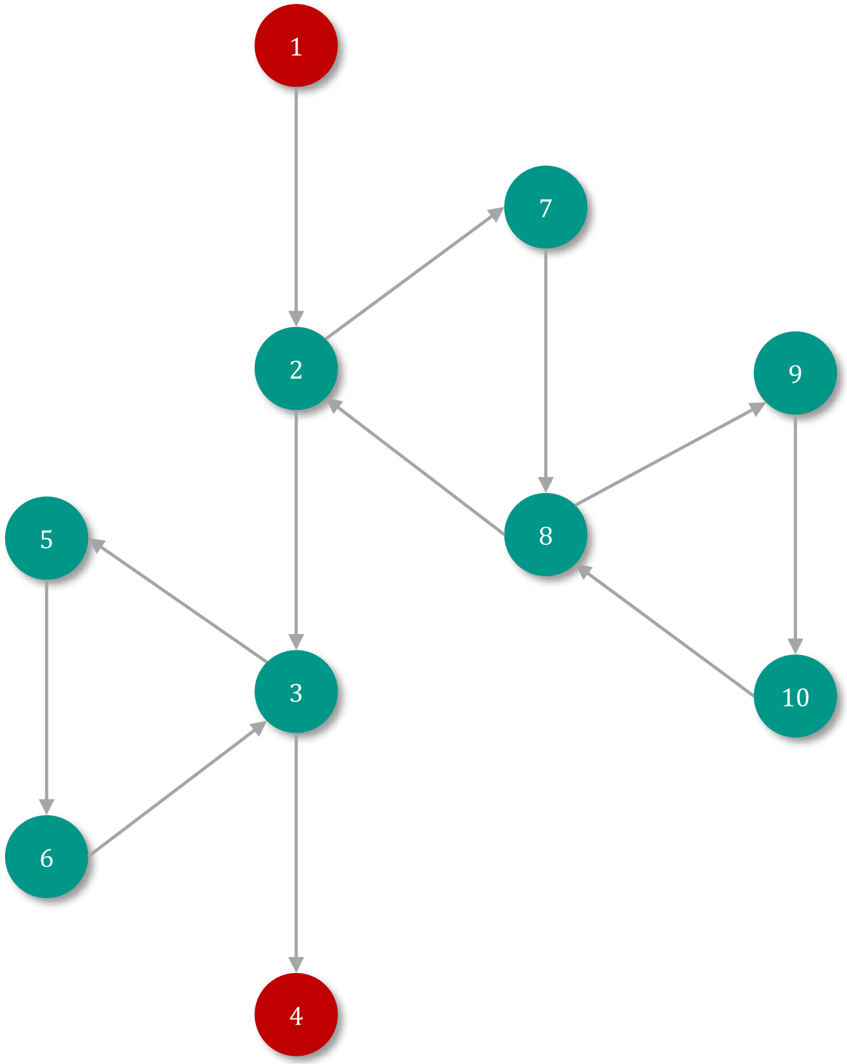
通过图中所有边恰好一次且行遍所有顶点的回路称为欧拉回路

欧拉图

具有欧拉回路的无向图或有向图称为欧拉图

半欧拉图

具有欧拉通路但不具有欧拉回路的无向图或有向图称为半欧拉图



欧拉图

对于无向图 G

是欧拉图当且仅当 G 是连通的且没有奇度顶点

对于无向图 G

是半欧拉图当且仅当 G 是连通的且恰有 2 个奇度顶点

对于有向图 G

是欧拉图当且仅当 G 的所有顶点属于同一个强连通分量且每个顶点的入度和出度相同

对于有向图 G , 是半欧拉图当且仅当

- 若将 G 中的有向边退化为无向边,那么 G 的所有顶点属于同一个连通分量
- 至多有一个顶点的出度与入度差为 1
- 至多有一个顶点的入度与出度差为 1
- 所有其他顶点的入度和出度相同

#3251、一笔画问题

题目描述

如果一个图存在一笔画，则一笔画的路径叫做欧拉路，如果最后又回到起点，那这个路径叫做欧拉回路
根据一笔画的两个定理,如果寻找欧拉回路,对任意一个点执行深度优先遍历找欧拉路,则对一个奇点执行 dfs ,时间复杂度为 $O(m+n)$
m 为边数, n 是点数

输入格式

第一行 n,m ，有 $n(1<n\le 100)$ 个点, m 条边,以下 m 行描述每条边连接的两点

输出

欧拉路或欧拉回路,输出一条路径即可

输入样例

```
5 5
1 2
2 3
3 4
4 5
5 1
```

输出样例

```
1 5 4 3 2 1
```

实验舱
青少年编程
走近科学 走进名校

谢谢观看