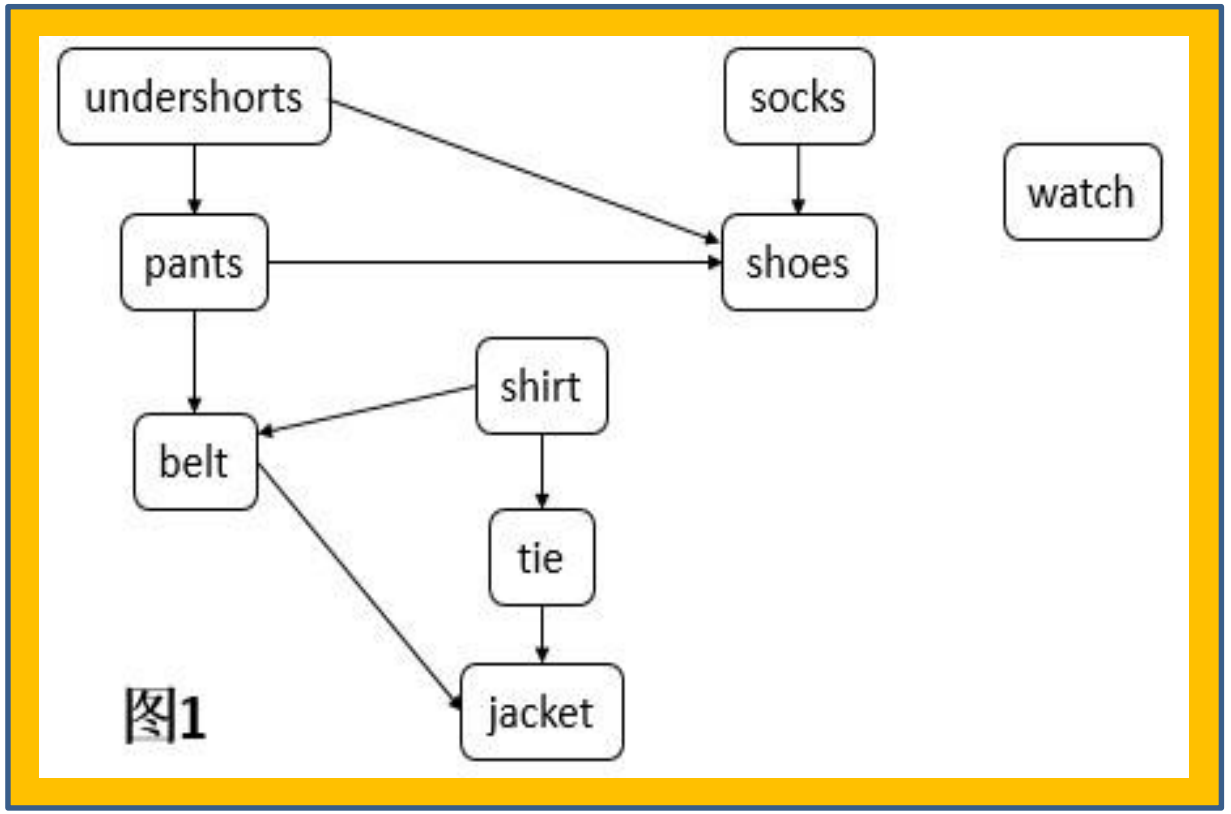
第2讲　图的拓扑排序算法

- https://vjudge.net/

问题1：拓扑排序
问题2：图的邻接表存储

引例1:



Bumstead教授早晨穿衣的过程：他必须先穿好某些衣服，才能再穿其他衣服（如先穿袜子后穿鞋），其他一些衣服则可以按任意次序穿戴（如袜子和裤子），在图1中，有向边<u，v>表示衣服u必须先于衣服v穿戴。

图2是一种穿衣过程

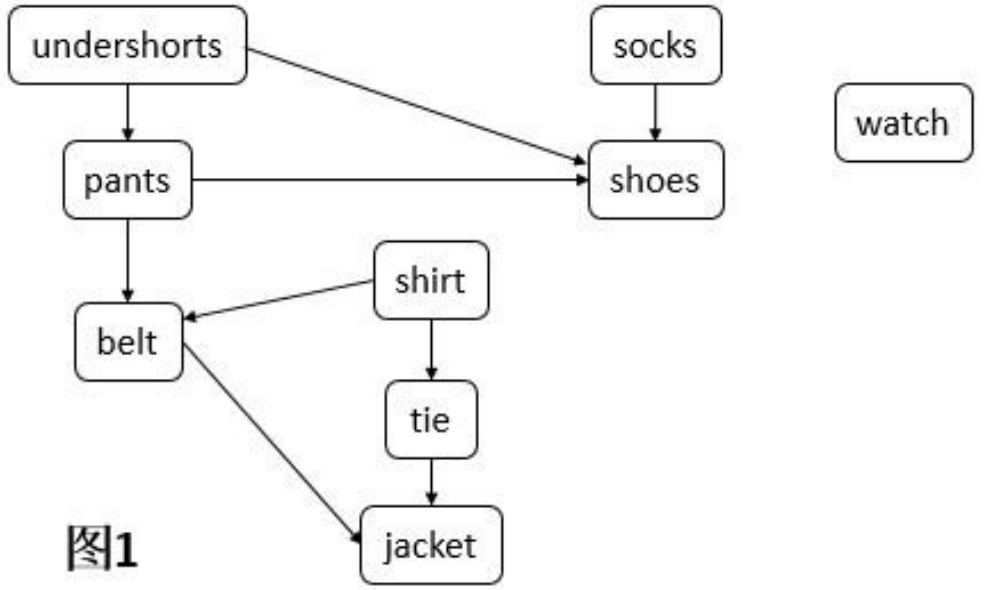


图1

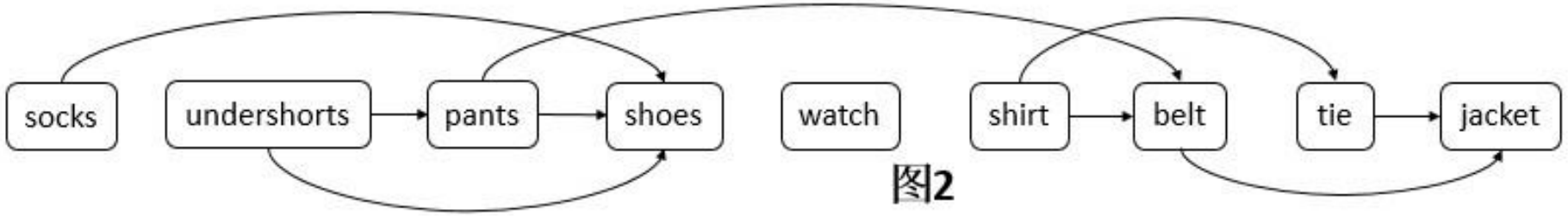
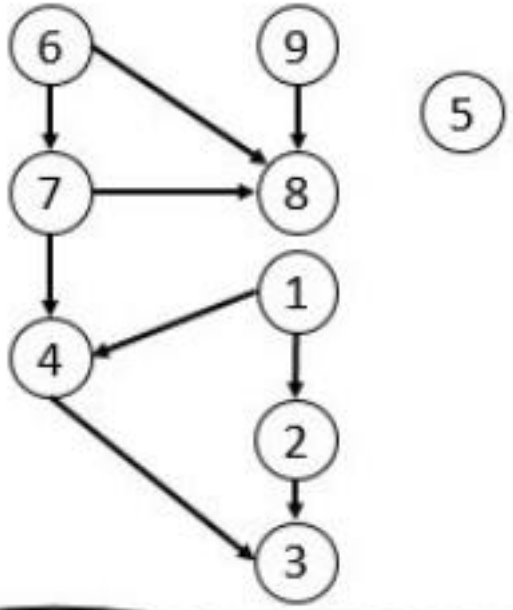
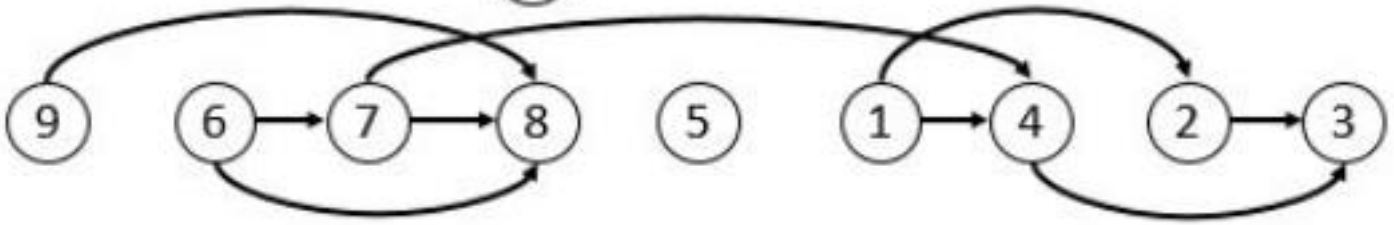


图2

◆将上图顶点转化为数字



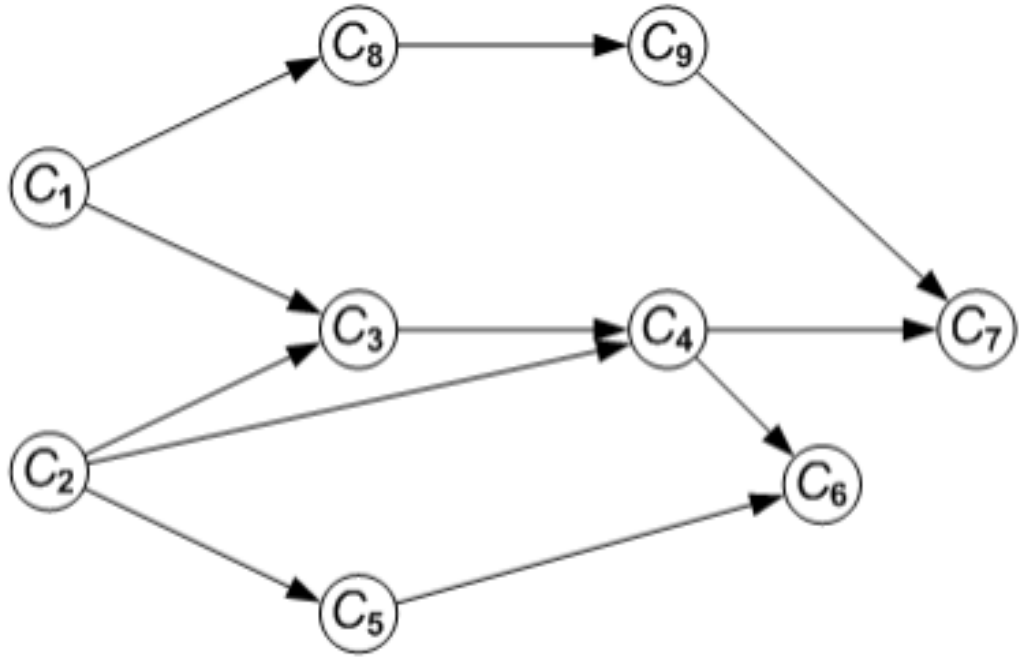
序列：



引例2：课程安排

课程代号	课程名称	先修课程
C_1	高等数学	
C_2	程序设计	
C_3	离散数学	C_1，C_2
C_4	数据结构	C_2，C_3
C_5	算法分析	C_2
C_6	编译技术	C_4，C_5
C_7	操作系统	C_4，C_9
C_8	普通物理	C_1
C_9	计算机原理	C_8

(a) 课程列表



(b) 学生选课工程图

引例3：士兵排队

- 有n个士兵(1≤n≤1000)，士兵的编号依次为1、2、3、……、n。指挥官要把这n个士兵从高到矮依次排成一行。但现在指挥官无法直接获得每个人的具体身高，只能获得“p1比p2高”这样的比较结果。
- 现在已知有m（m<=10000）个高矮关系，请按照从高到低输出一种合理的排队序列。
- 【输入】
- 第一行为一个正整数n。
- 第二行为一个正整数m。
- 以下m行，每行两个正整数x、y，表示x比y高。
- 【输出】
- 一行n个空格隔开的正整数，表示一种合理的排队序列（从高到矮）。

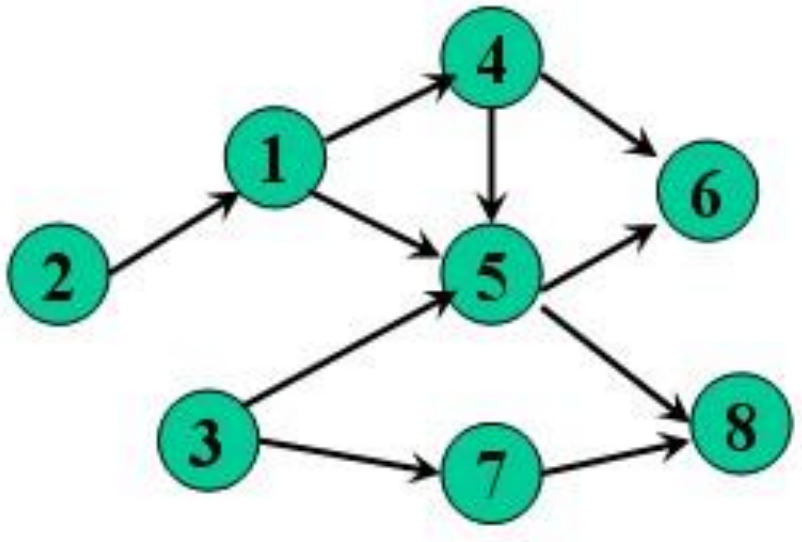
如：

```
8
10
2 1
1 4
1 5
4 5
4 6
3 5
3 7
5 6
5 8
7 8
```

如：

8
10
2 1
1 4
1 5
4 5
4 6
3 5
3 7
5 6
5 8
7 8

把士兵看成点，一个高矮关系看成一条有向边

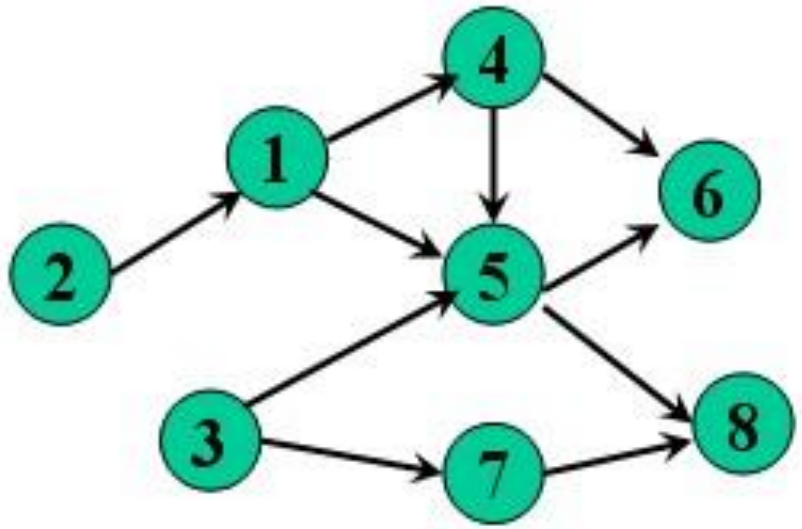


一种合法的排队顺序是：

2 3 1 4 7 5 6 8

◆拓扑排序的定义及算法：

- 对一个有向无环图G，将G中所有顶点排成一个线性序列，使得图中任意一对顶点u和v，若<u，v> ∈E，则在线性序列中u出现在v之前，这个线性序列称为图G的一个拓扑序列。
- 生成拓扑序列的过程叫做拓扑排序。



一种合法的拓扑序列：

2 3 1 4 7 5 6 8

◆拓扑排序算法：

- 1.从有向图中选择一个没有前驱(，先驱任务入度为0)的顶点并且输出它。
- 2.从图中删去该顶点,并且删去从该顶点发出的全部有向边。
- 3.返回1，直到剩余的图中不存在没有前趋的顶点为止。
- 4.剩余图中还有顶点，说明该有向图中存在环，无拓扑序列。
- 如果一个图有拓扑序列，拓扑序列可能不是唯一的。

◆实现方法1：迭代（时间o(n^2)）

1. for(int i=1;i<=n;i++)//查找n次
    1. 找出入度为0的一个点u
    2. 加入拓扑序列top[i]=u
    3. 找出u出发能达到的所有节点v
       v的入度减1
2. 输出拓扑序列 top

◆实现方法2：队列实现（时间o(n+m）)

1. 入度为0的点进入队列（可能多个）；
2. while 队列不空
    1. 队首元素u出队列；
    2. u并进入拓扑序列；
    3. 找出u出发到达的所有顶点v
        1. v的入度减1
        2. 如果v的入度为0，v进入队列。
3. 如果拓扑序列中的元素个数小于n个，有环存在；

说明：

- 只有入度为0（前驱任务已经完成）的点才能入队
- 和图的bfs的区别有哪些？

典型例题

1.任务排序(Ordering Task Uva10305)

样例输入：

```
5 4
1 2
2 3
1 3
1 5
0 0
```

样例输出：

```
1 4 2 5 3
```

vector清零

- 千万别忘了
- for(int i=0;i<=109;i++)g[i].clear();

2.奖金(一本通：第四章图论算法1352)

【题目描述】

由于无敌的A君在2008年世界英俊帅气男总决选中胜出，总经理B君心情好，决定给每位员工发奖金。公司决定以每个人本年在公司的贡献为标准来计算他们得到奖金的多少。

于是B君下令召开m方会谈。每位参加会谈的代表提出了自己的意见：“我认为员工a的奖金应该比b高！”。

B君决定要找出一种奖金方案，满足各位代表的意见，且同时使得总奖金数最少。每位员工奖金最少为100元。

【输入】

第一行两个整数n,m，表示员工总数和代表数；

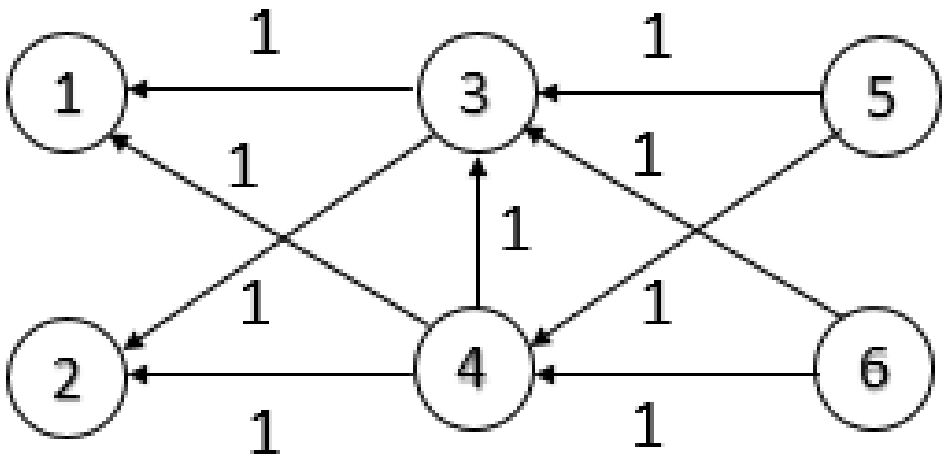
以下m行，每行2个整数a,b，表示某个代表认为第a号员工奖金应该比第b号员工高。

【输出】

若无法找到合法方案，则输出“-1”；否则输出一个数表示最少总奖金。

- 【样例输入】
- 6 9
- 1 3
- 1 4
- 2 3
- 2 4
- 3 4
- 3 5
- 3 6
- 4 5
- 4 6【样例输出】
- 609
- 【数据范围】
- 80%的数据满足n<=1000，m<=2000；
- 100%的数据满足n<=10000，m<=20000。

先确定钱最少，不比任何人高的那些人：
a比b高，建立有向边<b,a>



图的邻接表

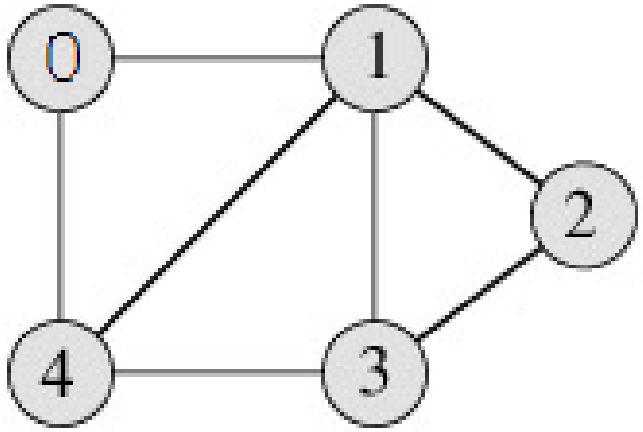
- 邻接矩阵的缺点：

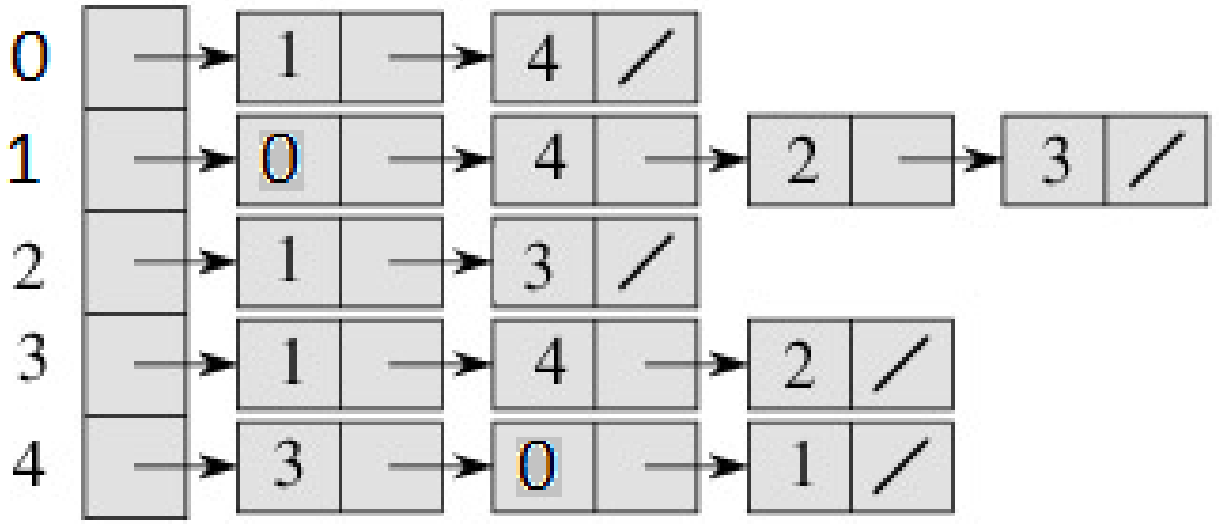
  G[10000][10000]

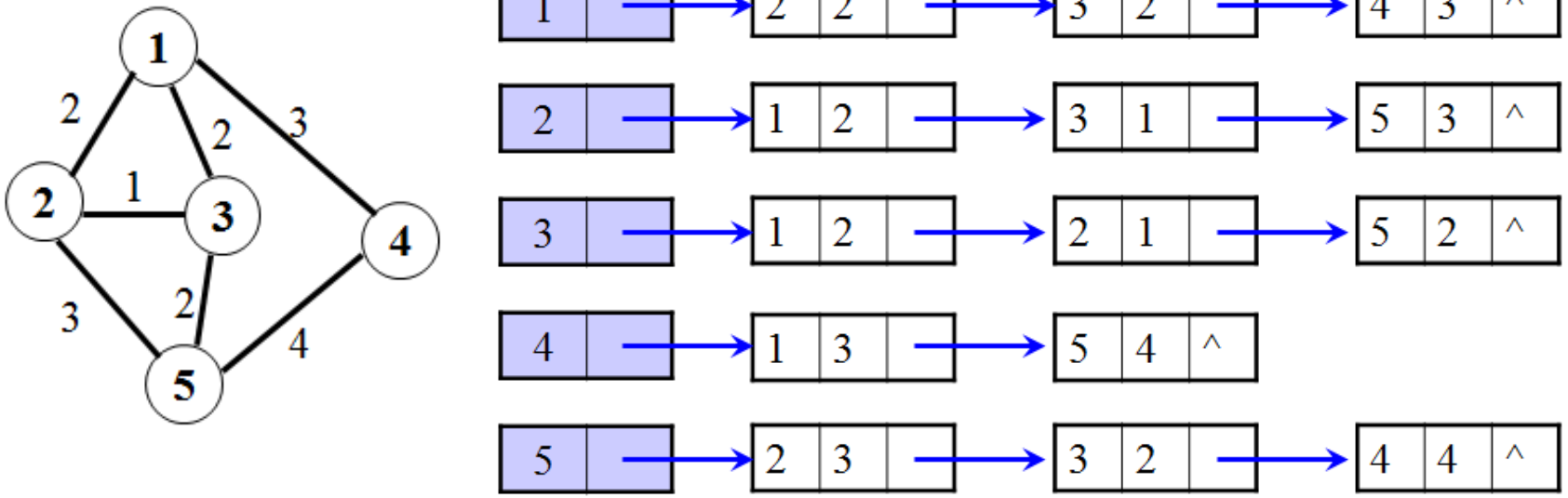
  空间大

  找邻接点慢

图的邻接表







实现

方法1：<vector>

方法2：数组模拟法

◆不定长数组<vector>

- 输入若干个数，按输入顺序输出

```
#include<iostream>
#include<vector>
using namespace std;
vector<int>a;
int n;
int main(){
    int x;
    while(cin>>x) a.push_back(x);
    for(int i=0;i<a.size();i++)
        cout<<a[i]<<" ";
    return 0;
}
```

- 和数组一样用
- 不用预先定义元素的个数：
- 后面插入元素：a.push_back(x);
- 元素的个数：a.size()
- a[i]

- For(int i=0;i<a.size();i++)cout<<a[i];

◆一维数组的每个元素也可以是不定长数组==二维数组

```
#include<cstdio>
#include<iostream>
#include<vector>
using namespace std;
const int maxn=100010;
vector<int>g[maxn];
int n,m;
int main(){
    cin>>n>>m;
    for(int i=0;i<m;i++){
        int u,v;
        cin>>u>>v;
        g[u].push_back(v);
    }
    for(int i=1;i<=n;i++)
        if(g[i].size()>0){
            for(int j=0;j<g[i].size();j++)
            cout<<g[i][j];
            cout<<endl;
        }
    return 0;
}
```