小马加编信息学教案(三十九)

图和$Floyd$

- 一. 课程内容
- 二. 知识讲解
  - 1. 图的定义
  - 2. 图的存储
    - 2. 1 邻接矩阵
    - 2. 2 邻接表
    - 2. 3 邻接矩阵和邻接表的比较。
  - 3. $Floyd$
    - 3. 1 $Floyd$的介绍
    - 3. 2 $Floyd$的原理
- 三. 经典例题
- 四. 提高巩固

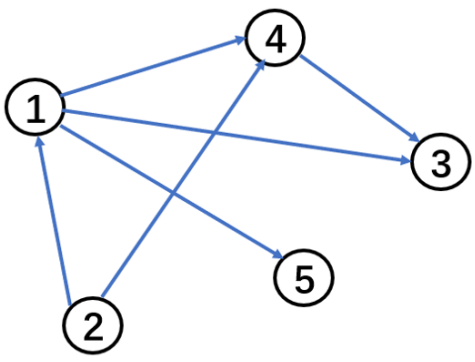
一. 课程内容

1. 图的定义
2. 图的存储
3. Floyd

二. 知识讲解

1. 图的定义

> 图由一些点和连接这些点的边组成，我们可以如下图给每个点一个编号

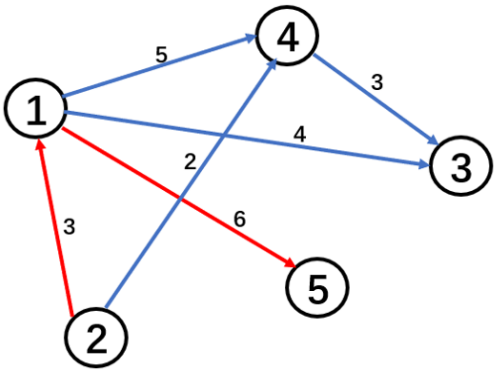


> 连接两个点u, v的边有两种。

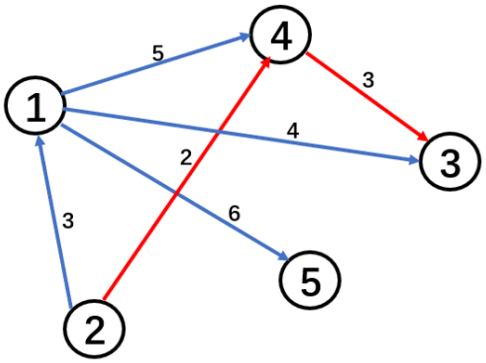
- 一种是有向边，只能从一个点走到另一个点，如上图1号点和4号点之间的边，根据箭头方向，只能从1号点走到4号点。
- 另一种是无向边，可以在两个点之间任意走动，即可以从u走到v也可以从v走到u。

> 由有向边组成的图可叫做**有向图**，由无向边组成的图叫**无向图**。

> 另外我们可以给每条边一个权值，代表边的长度，那么可以定义两个点经过某条路径的距离为走过所有边的长度和。如下图中的红色路径。



> 仔细观察可以发现，从2号点到3号点有多条路径可以选择，但是下图标识的长度为5的路径是所有路径中最短的，那么可以将这条路叫做2节点到3节点的**最短路径**，长度为5。



2. 图的存储

> 考虑怎么在程序成存储一幅有n个点m条边的图，常用的方法有两种：邻接矩阵和邻接表。

2. 1 邻接矩阵

> 我们可以用一个$n \times n$的二维数组$road$来存储这幅图的路径。对于由u指向v的有向边，将路径长度保存在$road[u][v]$处，若没有路径则令其等于-1。对于无向边，可以视为两条方向相反的有向边，将路径信息分别存储在$road[u][v]$和$road[v][u]$中。

2. 2 邻接表

> 邻接矩阵可以很方便的知道任意两点间的连边情况，但是需要n^2的存储空间，当n比较大时就无法用邻接矩阵存储。

> 邻接矩阵的缺点是有很大一部分空间是没有存储边的信息的，会浪费掉。那么换个思路，只对出现在图中的边开空间存储。即对于每个点，不用一个大小为n的数组存储与每个点是否有边相连，而是只记录连出边的信息。

> 更具体的说，就是用$vector$将边的信息保存下来，我们可以用两个$vecotr$，一个记录这条边连向那个点，一个记录这条边的长度。比如有一条u到v长度为len的有向边：

```
vector<int> poi[100], l[100];
//用poi[i]记录i连出去的边连向哪个点
//用l[i]记录i连出去的边的长度

poi[u].push_back(v);
l[u].push_back(len);
//加入一条从u连向v的长度为len的有向边
//若为无向边则再在v对应的poi和l中加入边即可
```

> 之前学过，$vector$是动态开空间的，那么只需要$O(m)$的空间就能把这副图中边的信息记录下来。

2. 3 邻接矩阵和邻接表的比较。

功能	邻接矩阵	邻接表
所需空间	$O(n^2)$	$O(m)$
遍历所有边	$O(n^2)$	$O(m)$
访问两个点之间的边	$O(1)$	$O(m)$

- 对于所需空间上面已经提到
- 当要遍历一个点连出去的边时，对于邻接矩阵，需要将图中每个点访问一次，用$road$数组判断有没有连边。而邻接表直接访问$vector$即可。
- 当判断两个点之间有没有连边时，对于邻接矩阵，只需通过$road$数组判断。而邻接表要将整个$vector$遍历一次。

3. $Floyd$

3. 1 $Floyd$的介绍

> $Floyd$是一种求一幅图中两两点之间最短路的算法，通过动态规划的思想实现。其采用邻接矩阵对边进行存储。

3. 2 $Floyd$的原理

> 记$f[i][j]$表示从点i到点j的最短路（若为无向图则$f[i][j]=f[j][i]$），那么我们可以先根据图中已有的边对其赋初始值。若两个点之间没有边相连，则为∞。

```
if (i == j) {
        f[i][j] = 0;
        //自己到自己的距离是0
} else {
        if (road[i][j] == -1) {
                f[i][j] = Inf;
                //两点没有直接相连，距离为Inf，Inf为一个很大的常数。
        } else {
                f[i][j] = road[i][j];
                //两点有直接相连，对f数组赋初值
        }
}
```

> 现在只考虑了两个点直接相连的情况，怎么将经过中间节点的路径也考虑上？由于图中两个点之间存在很多路径，经过中间点的可能性会很多，没办法一次性考虑完或者枚举。那么换个思路，可以每次只考虑一个中间点是否在其他点对的最短路径上。

> 我们用动态规划的思想，分n轮来更新f数组，对于第k轮，只考虑将编号为1到k作为中间点的路径，那么对于第$k+1$轮对于任意点对i,j只有两种情况：

- i,j之间的最短路径不经过中间点$k+1$，$f[i][j]=f[i][j]$
- i,j之间的最短路经过中间点$k+1$，$f[i][j]=f[i][k+1]+f[k+1][j]$，即从$i$先到$k+1$，再从$k+1$到$j$。

> 综上所述，对于第k轮，我们可以得到通过一下式子更新f数组

```
f[i][j] = min(f[i][j], f[i][k] + f[k][j]);
```

> 经过n轮更新后的f数组就是考虑了所有点作为中间点的两两点间的最短路。

```
        for (int k = 1; k <= n; k ++) //枚举更新n轮
                for (int i = 1; i <= n; i ++)
                        for (int j = 1; j <= n; j ++) //枚举点对
                                f[i][j] = min(f[i][j], f[i][k] + f[k][j]);
        //若f[i][j]等于Inf则i,j之间不连通，否则f[i][j]为i,j之间的最短路。
```

> 由于使用邻接表存储，所以复杂度为$O(n^2)$，由于要枚举n轮，每轮都要枚举n^2个点对，所以时间复杂度是$O(n^3)$。

三. 经典例题

1. 寻找联通块

> 给一幅n个点，m双向条边的图，问有多少个节点和1号节点在同一个联通块（若两个点能通过若干条路径到达则称两个点在同一个联通块）。

> 输入格式:
> 第一行两个整数n, m。
> 接下来m行，每行两个整数u, v表示一条u连向v的双向边。
> $n, m \leq 10^5$

> 输出格式:
> 一行一个整数表示有多少个节点和1号节点在同一个联通块。

样例输入	样例输出
6 5 1 2 2 4 3 5 5 6 3 6	3

2. 最短距离（1）

> 给一副n个点，m条双向边的图，问每个节点到1号节点至少经过多少条边，保证每个节点都能到达1号节点。

> 输入格式:
> 第一行两个整数n, m。
> 接下来m行，每行两个整数u, v表示一条u连向v的双向边。
> $n, m \leq 10^4$

> 输出格式:
> 一行n个整数，第i个整数表示第i个点到1号节点的最短距离。

样例输入	样例输出
4 5 1 2 2 4 4 3 2 3 1 4	0 1 2 1

3. 最短距离（2）

> 给定一副n个点m条边的无向图，现在有t组询问，每组询问给定s_i和t_i，询问这两个点之间的最短路径。保证没有重边。

> 输入格式：
> 第一行三个整数n, m, t。
> 接下来m行，每行三个整数a_i, b_i, c_i表示一条从a_i连向b_i长度为c_i的无向边。
> 然后t行，每行两个整数s_i, t_i，表示一组询问。
> $n \leq 100$
> $m \leq 10000$

> 输出格式：
> 对于每组询问，输出一个整数表示最短距离。

样例输入	样例输出
5 6 3 1 2 2 1 4 2 4 3 3 4 2 1 1 5 1 3 5 2 1 3 1 4 2 5	3 2 3

四. 提高巩固

1. 最好的地方

> 约翰拥有p个牧场.贝茜特别喜欢其中的f个.所有的牧场由c条双向路连接，第i路连接着a_i, b_i（双向），需要t_i单位时间来通过。作为一只总想优化自己生活方式的奶牛，贝茜喜欢自己某一天醒来，到达所有那f个她喜欢的 牧场的平均需时最小.那她前一天应该睡在哪个牧场呢？请帮助贝茜找到这个最佳牧场。

> 输入格式：
> 第一行三个整数p, f, c。
> 接下来f行，每行一个正整数表示贝茜喜欢的f个牧场。
> 接下来c行，每行三个正整数a_i, b_i, t_i表示一条路径。
> $n \leq 500$
> $c \leq 8000$
> $t \leq 900$

> 输出格式:
> 一个整数表示最佳牧场。

样例输入	样例输出
13 6 15 11 13 10 12 8 1 2 4 3 7 11 3 10 11 1 4 13 3 9 10 3 2 3 2 3 5 4 5 9 2 6 7 6 5 6 1 1 2 4 4 5 3 11 12 3 6 10 1 7 8 7	10

2. 牛栏

> Farmer John想让他的奶牛准备郡级跳跃比赛，贝茜和她的伙伴们正在练习跨栏。她们很累，所以她们想消耗最少的能量来跨栏。 显然，对于一头奶牛跳过几个矮栏是很容易的，但是高栏却很难。于是，奶牛们总是关心路径上最高的栏的高度。 奶牛的训练场中有n个站台，分别标记为$1..n$。所有站台之间有m条单向路径，第i条路经是从站台s_i开始，到站台e_i，其中最高的栏的高度为h_i。无论如何跑，奶牛们都要跨栏。 奶牛们有ti个训练任务要完成。第i个任务包含两个数字a_i和b_i，表示奶牛必须从站台a_i跑到站台b_i，可以路过别的站台。奶牛们想找一条路径从站台a_i到站台b_i，使路径上最高的栏的高度最小。你的任务就是写一个程序，计算出路径上最高的栏的高度的最小值。

> 输入格式:
> 第一行三个整数，n, m, ti
> 接下来m行，每行三个整数s_i, e_i, h_i表示一条单向路径，
> 接下来t行，每行两个整数a_i, b_i表示一个训练任务。
> $a_i, b_i, s_i, t_i \leq n \leq 300$
> $m \leq 25000$
> $t \leq 40000$
> $h_i \leq 10^6$

> 输出格式:
> 对于每个训练任务，输出一个整数表示路径上最高护栏的最小值。

样例输入	样例输出
5 6 3 1 2 12 3 2 8 1 3 5 2 5 3 3 4 4 2 4 8 3 4 1 2 5 1	4 8 -1

3. 奶牛大赛

> Farmer John的n头奶牛们最近参加了场程序设计竞赛。在赛场上，奶牛们按$1..n$依次编号。每头奶牛的编程能力不尽相同，并且没有哪两头奶牛的水平不相上下，也就是说，奶牛们的编程能力有明确的排名。 整个比赛被分成了若干轮，每一轮是两头指定编号的奶牛的对决。如果编号为a的奶牛的编程能力强于编号为b的奶牛($a \neq b$)，那么她们的对决中，编号为a的奶牛总是能胜出。 Famer John想知道奶牛们编程能力的具体排名，于是他找来了奶牛们所有m轮比赛的结果，希望你能根据这些信息，推断出尽可能多的奶牛的编程能力排名。比赛结果保证不会自相矛盾。

> 输入格式：
> 第一行两个整数n, m
> 接下来m行，每行两个整数a, b表示一场决斗，且第一头奶牛获胜。
> $n \leq 100$
> $m \leq 4500$

> 输出格式：
> 一行一个整数表示可以确定排名的奶牛的个数。

样例输入	样例输出
5 5 4 3 4 2 3 2 1 2 2 5	2