Floyd 算法研究

sdsy wahning

2023.8

多源最短路

- n 次 dijkstra 算法 O(n*n*logn)
- Floyd 算法 O(n*n*n)

理解Floyd算法的本质

- 本质是动态规划

- 定义一个数组 f[k][i][j]，表示只允许经过结点 1 到 k（这个非常重要）的前提下，结点 i 到结点 j 的最短路长度。
- 很显然， f[n][i][j] 就是结点 i 到结点 j 的最短路长度。

- 初始化：
- f[0][i][j]： 若存在 i 到 j 的边则初始化为边权 w(i,j)，否则初始化为正无穷 (i!=j)，以及 0 (i==j)
- f[k][i][j] = min(f[k-1][i][j], f[k-1][i][k]+f[k-1][k][j])

- 时间、空间是 O(N^3) 。
- 其实数组的第一维是没有用的，于是可以直接改成 f[i][j] = min(f[i][j], f[i][k]+f[k][j])，因为拿 k 做过渡点更新的结果，不可能再次用来更新其它以 k 作为过渡点的点对，所以完全可以把第一维去掉。

#1070

- 数据保证了 $t[0] \le t[1] \le \ldots\ldots \le t[N-1]$
- Floyd 的本质！

- 自行完成代码并AC!

总结反思

- 通过此题，深刻理解Floyd算法的本质。

#1069

- 无向连通图 N 个节点，M 条边的无向图，边有边权，点有点权，现给出 Q 个询问，每个询问查询两个节点之间的最小花费。这里最小花费的定义是两个节点之间的路径与这条路径中经过的节点点权的最大值之和。

- 多源最短路问题
- $1 \leq N \leq 250$
- 显然 floyd

- 最短路径涉及到路径中最大的点权，怎么办？

- 我们重新回想一遍floyd算法的原理：
- i 到 j 有两种可能：直接到 和 借助中转点 k，二者取 min

- 我们再往下细细的想一下：k 代表的是中转点，枚举顺序可以是任意的吗？
- 显然可以是任意的！
- 突破口就在这：我们可以肆意的修改 k 的枚举顺序！

- 我们对点按权值从小到大重新编号，这是的中转点 k 代表的就是重新编号的点，k 越大，点权越大
- 我们只需要从小到大枚举中转点 k
- 由于 k 是从小到大枚举的，我们在找 i, j 之间的最短路的时候，我们的中转点如果只是枚举到了 k，这说明了我们当前的 i, j 之间的最短路中并没有点权超过 k点点权的点，最后的 k 即作为 i->j 的路径上除 i, j 外点权最大的点
- 然后就可以去跑flyod，该路径上点权的最大值即为 i, j, k 三个点中点权的最大值！

- 自行完成代码并AC！

Floyd算法体现的是一种思想

- 它并不是只可以求最短路问题。
- 就好比dijkstra也可以求最长路、最大边、最小边、最大边最小值、最小边最大值 等等。
- Floyd也可以求解类似的问题。

#1015

- 求起点s到终点t的所有路径中的最大边权的最小值

POJ2253 Frogger

- Description

- 一只叫Freddy的青蛙蹲坐在湖中的一块石头上。突然他发现一只叫Fiona的青蛙在湖中的另一块石头上。Freddy想要跟Fiona约会，但由于湖水太脏，他不想游泳过去而是跳过去找Fiona。
- 很不幸，Fiona所在的石头距离他有点远，甚至超出了他的跳跃能力。然而Freddy注意到湖中还有一些其他的石头。这些石头也许会将这个很长的跳跃距离化成若干个短的跳跃距离。
- 我们定义“青蛙距离”为Freddy跳到Fiona那里所需要的若干次跳跃中最长的那一次。现在给你Freddy，Fiona，以及湖中其他石头的坐标，让你求出最短的“青蛙距离”。

- Input

- 输入有可能是多组测试数据。每组数据的第一行有一个整数n(2<=n<=200)，表示湖中一共有多少块石头。接下来的n行，每一行有两个整数xi， yi(0 <= xi,yi <= 1000)，表示第i块石头的坐标。第1块石头的坐标是Freddy所在的位置，第二块石头的坐标是Fiona所在的位置，其他的石头上都没有青蛙。当输入n=0的时候，程序结束。

- Output

- 对于每一组测试数据，先输出一行"Scenario #x"，然后在下一行输出"Frog Distance = y"。其中x表示当前是第几组测试数据，y为该组数据的最小“青蛙距离”。每两组测试数据之间输出一个空行。

- Sample Input
- 2
- 0 0
- 3 4

- 3
- 17 4
- 19 4
- 18 5

- 0

- Sample Output
- Scenario #1
- Frog Distance = 5.000

- Scenario #2
- Frog Distance = 1.414

#1015

头脑风暴

- 你做过类似的题目吗?
- 你能想到哪些方法?

- 方法1：floyd 的变形

- 方法2： Dijkstra 或 Spfa 的变形

- 方法3：最小生成树 Kruscal 的变形

- 把以上所有方法的代码完成并AC！

#541

- 思考：
- 什么情况下，一位选手的名次可以确定？

- 比他强的人数 + 比他弱的人数 = n-1
- 问题转化为：
- 求比他强的人数 和 比他弱的人数
- 如何求?

- 方法1：dfs
- 建图
- 有向图

- dfs_win 比他弱的人数
- dfs_lose 比他强的人数

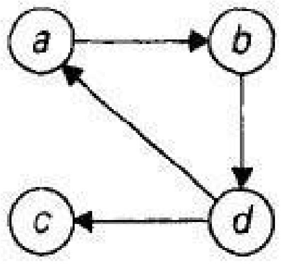
- 自行写出代码！

- 还有其他方法吗?

有向图的传递闭包

定义　一个 n 顶点有向图的传递闭包可以定义为一个 n 阶布尔矩阵 $T=\{t_{ij}\}$，如果从第 i 个顶点到第 j 个顶点之间存在一条有效的有向路径，矩阵第 i 行（$1\le i\le n$）第 j 列（$1\le j\le n$）的元素为 1；否则，t_{ij} 为 0。

作为例子，图 8.2 给出了一个有向图、该图的邻接矩阵及其传递闭包。



(a)

$$A = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

(b)

$$T = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

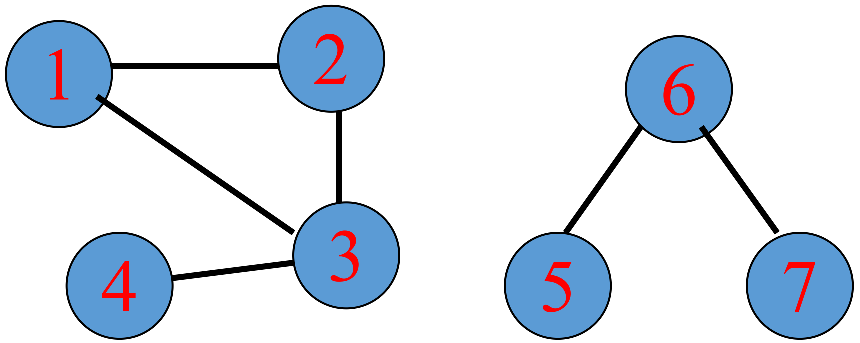
(c)

图 8.2　（a）有向图；（b）它的邻接矩阵；（c）它的传递闭包

- 简单地说，传递闭包的矩阵就是对邻接矩阵的修改。a能到b，b能到d，那么根据传递性，a也能到d，从而邻接矩阵中a到d原本为0的值就被改成1，表示a可达d。

无向图的传递闭包

- 给出一无向图，输出其传递闭包



输入：
7
1 2
2 3
1 3
3 4
5 6
6 7

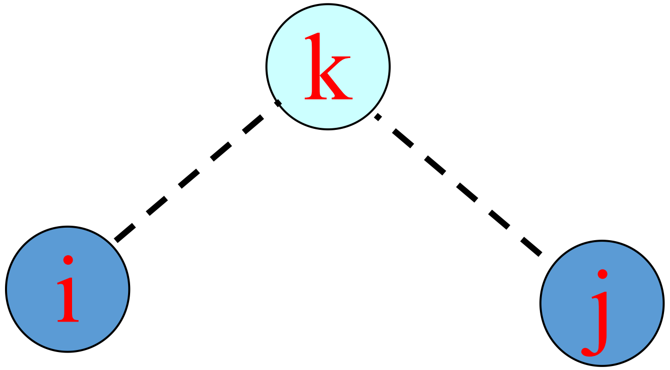
输出：
1 1 1 1 0 0 0
1 1 1 1 0 0 0
1 1 1 1 0 0 0
1 1 1 1 0 0 0
0 0 0 0 1 1 1
0 0 0 0 1 1 1
0 0 0 0 1 1 1

邻接矩阵

0 1 1 0 0 0 0
1 0 1 0 0 0 0
1 1 0 1 0 0 0
0 0 1 0 0 0 0
0 0 0 0 0 1 0
0 0 0 0 1 0 1
0 0 0 0 0 1 0

判断结点 i 和 j的连通性:

- 1、结点i和j如果原来有边则连通。
- 2、如果i和j之间没有边:
  - 如果存在另外的一个结点k，满足：i与k连通，k与j连通，则i与j连通。
  - 否则i和j不连通。



初始：can[i , j]: true ，i与j之间有边；false：无边。

则：can[i , j]=can[i , j] or (can[i , k] and can[k , j])

过程

- for k=1 to n
-     for i=1 to n
-         for j=1 to n
-             can[i,j]=can[i,j] or (can[i,k] and can[k,j]);

#541

- 方法2： floyd 求传递闭包
- 自行完成代码并AC！

#1072

Floyd求最小环

算法分析

- 暴力来枚举环
- 当删除其中一条边(i,j)后再跑一边从i到j的最短路，然后加上边(i,j)的值就是含有边(i,j)的最小环的值，这样最坏的时间复杂度可以达到O(n^4)，显然复杂度有点大。

Floyd 求最小环

- 有向图?
- 无向图?

- 首先对于有向图：
- 由于一条边 <U,V> 只能从 U 到 V，而不能逆行之，那么我们只需要更新出图中任意 2 个点之间的最短路径。
- 记F[U][V]表示U到V的最短距离
- G[U][V] 是原图中U到V的边长
- 对于每条边 <U,V>我们只要将 G[U,V]+F[V,U]来更新答案就行了。

- 无向图的最小环相对麻烦一些~

参考

- for(int k=1; k<=n; k++)
- {
- //新增部分:
- for(int i=1; i<k; i++)
- for(int j=i+1; j<k; j++)//for(int j=1;j<i;j++)
- mincircle = min(mincircle, F[i][j]+G[j][k]+G[k][i]);
- //通常的 floyd 部分:
- for(int i=1; i<=n; i++)
- for(int j=1; j<=n; j++)
- Fist[i][j] = min(F[i][j], Dist[i][k] + Disk[k][j]);
- }

#1072

- 自行完成代码并AC!