Section 2.1　Flood Fill 种子染色法

译 by Lucky Crazy & Felicia Crazy

（Flood Fill 按原意应翻译成"水流式填充法"（如果我没译错），有些中文书籍上将它称作"种子染色法"，然而大部分的书籍（包括中文书籍）都直接引用其英文原名：Flood Fill。介于此，下文所有涉及到 Flood Fill 的都直接引用英文 ——译者）

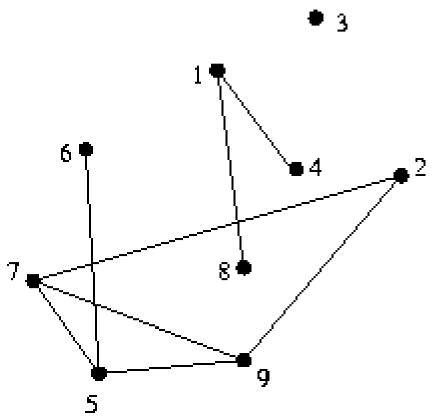
样例：　相连的农场

Farmer John 的农场被一次意外事故破坏了，有一些农场与其他的农场之间有道路相连，而有些道路却已被破坏。这使得 Farmer John 无法了解到从一个农场能否到达另一个农场。你的任务就是帮助 Farmer John 来了解哪些农场是连通的。

给出：

上题实际上就是要求寻找一张无向图的所有极大连通子图。

给出一张未知连通性的图，如下图：



可知，该图的极大连通子图是：{1,4,8}，　{2,5,6,7,9} 和 {3}。

算法：Flood Fill

Flood Fill 可以用深度优先搜索，广度优先搜索或广度优先扫描来实现。他的实现方式是寻找到一个未被标记的结点对它标记后扩展，将所有由它扩展出的结点标上与它相同的标号，然后再找另一个未被标号的 结点重复该过程。这样，标号相同的结点就属于同一个连通子图。

深搜：取一个结点，对其标记，然后标记它所有的邻结点。对它的每一个邻结点这么一直递归下去完成搜索。

广搜：与深搜不同的是，广搜把结点加入队列中。

广度扫描（不常见）：每个结点有两个值，一个用来记录它属于哪个连通子图（c），一个用来标记是否已经访问（v）。算法对每一个未访问而在某个连通子图当中的结点扫描，将其标记访问，然后把它的邻结点的（c）值改为当前结点的（c）值。

深搜最容易写，但它需要一个栈。搜索显式图没问题，而对于隐式图，栈可能就存不下了。

广搜稍微好一点，不过也存在问题。搜索大的图它的队列有可能存不下。深搜和广搜时间复杂度均为 O(N+M)。其中，N 为结点数，M 为边数。

广度扫描需要的额外空间很少，或者可以说根本不要额外空间，但是它很慢。时间复杂度是 O(N^2+M)。

（实际应用中，我们一般写的是 DFS，因为快。空间不是问题，DFS 可改用非递归的栈操作完成。但为了尊重原文，我们还是译出了广度扫描的全过程。——译者）

广度扫描的伪代码

代码中用了一个小技巧，因此无须额外空间。结点若未访问，将其归入连通子图（-2），就是代码里的 component -2。这样无须额外空间来记录结点是否访问，请读者用心体会。

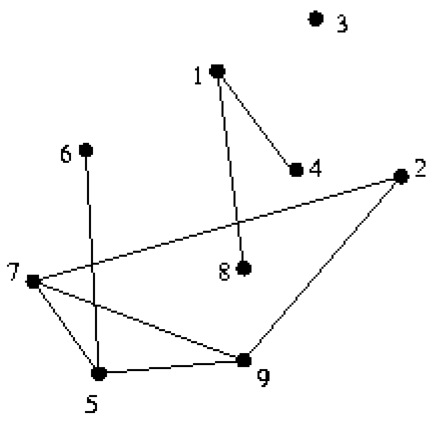
```
# component(i) denotes the
```

```
# component that node i is in
function flood_fill(new_component)
do
  num_visited = 0
  for all nodes i
  if component(i) = -2
    num_visited = num_visited + 1
    component(i) = new_component
    for all neighbors j of node i
      if component(j) = nil
        component(j) = -2
until num_visited = 0
function find_components
num_components = 0
for all nodes i
  component(node i) = nil
for all nodes i
  if component(node i) is nil
    num_components =
          num_components + 1
    component(i) = -2
    flood_fill(component
                num_components)
```

算法的时间复杂度是 $O(N^2)$，每个结点访问一次，每条边经过两次。

实例

考虑刚才的那张图：



开始时，所有的结点都没有访问。（下例中未访问被表示为 -2)

首先从结点 1 开始，结点 1 未访问，那么先处理结点 1，将它归入连通子图 1。

结点 1

连通子图 -2

标记完成后，对它进行第一步的扩展，由结点 4 和结点 8 与结点 1 连通，故它们被扩展出来。

结点 1 4 8

连通子图 1 -2 -2

之后，先处理结点 4，将它与结点 1 归入相同的连通子图。现在它没有可扩展的结点了（结点 1 已被扩展过）

结点 1 4 8

连通子图 1 1 -2

接着处理结点 8。结果与结点 4 一样。

结点 1 4 8

连通子图 1 1 1

现在，所有与结点 1 连通的结点都已扩展，标号为 1 的连通子图产生了。那么我们将跳出扩展步骤，寻找下一个连通子图，标号为 2。

与上一步相同的顺序，找到结点 2。

结点 1 2 4 8

连通子图 1 -2 1 1

扩展结点 2，结点 7 与结点 9 出现。

结点 1 2 4 7 8 9

连通子图 1 2 1 -2 1 -2

下一步，扩展结点 7，结点 5 出现。

然后是结点 9。

扩展结点 5。结点 6 出现。

结点 1 2 4 5 6 7 8 9

连通子图 1 2 1 2 -2 2 1 2

很遗憾，结点 6 没有可供扩展的结点。至此连通子图 2 产生。

结点 1 2 4 5 6 7 8 9

连通子图 1 2 1 2 2 2 1 2

之后寻找连通子图 3，至此，仅有结点 3 未被扩展。

结点 3 没有可供扩展的结点，这样，结点 3 就构成了仅有一个结点的连通子图 3。

结点 1 2 3 4 5 6 7 8 9

连通子图 1 2 3 1 2 2 2 1 2

结点 3 处理结束后，整个图的所有 9 个结点就都被归入相应的 3 个连通子图。Flood Fill 结束。

问题提示

这类问题一般很清晰，求解关于“连通”的问题会用到 Flood Fill。它也很经常用作某些算法的预处理。

扩展与延伸

有向图的连通性比较复杂。

同样的填充算法可以找出从一个结点能够到达的所有结点。每一层递归时，若一个结点未访问，就将其标记为已访问（表示他可以从源结点到达)，然后对它所有能到达且为访问的结点进行下一层递归。

若要求出可以到达某个结点的所有结点，你可以对后向弧做相同的操作。

例题

控制公司 [有删节，IOI 93]

已知一个带权有向图，权值在 0-100 之间。

如果满足下列条件，那么结点 A“拥有”结点 B：

A = B

从 A 到 B 有一条权值大于 50 的有向弧。

存在一系列结点 C 1 到 C k 满足 A 拥有 C 1 到 C k, 每个节点都有一条弧到 B，记作 x 1 ，x 2 ...x k，并且 x 1 + x 2 + ... + x k > 50。

找出所有的（A，B）对，满足 A 拥有 B。

分析：这题可以用上面提到的“给出一个源，在有向图中找出它能够到达的结点”算法的改进版解决。要计算 A 拥有的结点，要对每个结点计算其“控股百分比”. 把它们全部设为 0。现在，在递归的每一步中，将其标记为属于 A 并把它所有出弧的权加到“控股百分比”中。对于每个“控股百分比”超过 50 的结点，进行同样的操作（递归）。

街道赛跑 [IOI 95]

已知一个有向图，一个起点和一个终点。

找出所有的 p，使得从起点到终点的任何路径都必须经过 p。

分析：最简单的算法是枚举 p，然后把 p 删除，看看是否存在从起点到终点的通路。时间复杂度为 O(N (M + N))。题目的数据范围是 M

牛路 [1999 USACO 国家锦标赛, 有删节]

连通图的直径定义为图中任意两点间距离的最大值，两点间距离定义为最短路的长。

已知平面上一个点集，和这些点之间的连通关系，找出不在同一个连通子图中的两个点，使得连接这两个点后产生的新图有最小的直径。

分析：找出原图的所有连通子图，然后枚举不在同一个连通子图内的每个点对，将其连接，然后找出最小直径。

笨蛋建筑公司

Farmer John 计划建造一个新谷仓。不幸的是，建筑公司把他的建造计划和其他人的建造计划混淆了。Farmer John 要求谷仓只有一个房间，但是建筑公司为他建好的是有许多房间的谷仓。已知一个谷仓平面图，告诉 Farmer John 它一共有多少个房间。

分析：随便找一个格子，遍历所有与它连通的格子，得到一个房间。然后再找一个未访问的格子，做同样的工作，直到所有的格子均已访问。虽然题目给你的不是直接的图，但你也可以很容易地对其进行 Flood Fill。