最小树形图

清华大学
马皓然

定义

- 最小生成树针对无向加权连通图，在有向带权图中，指定一个特定的顶点root，求出一棵以root为根的有向生成树T，使得T中所有边的总权值最小，该问题称为最小树形图。

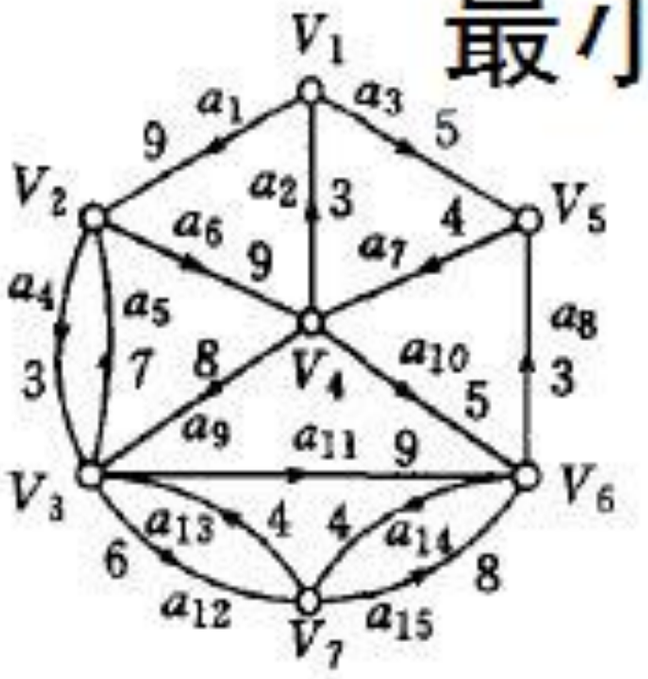
朱刘算法

- 朱刘算法是最小树形图的第一个算法，在1965年由朱永津和刘振宏提出，复杂度为O（VE）。

算法流程

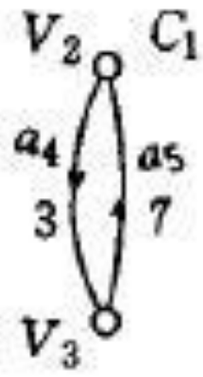
- 对固定根root进行一遍DFS判断是否存在最小树形图。
- 为除了root之外的每个顶点选定一条最小的入边（用pre[u]记录顶点u最小入边的起点）。
- 判断2中构造的入边集合是否存在有向环，如果不存在，那么这个集合就是该图的最小树形图。
- 如果3中判断出现有向环，则消环缩点。设（u, v, w）表示从u到v的权为w的边。假设3中判断出的有向环缩为新顶点new，若u位于环上，并设环中指向u的边权是ω[u]。那么对于每条从u出发的边（u, v, w），在新图中连接（new, v, w），对于每条进入u的边（in, u, w），在新图中建立边（in, new, w-ω[u]）。新图里最小树形图的权加上旧图中被收缩的环的权和，就是原图中最小树形图的权值。重复2，3，4。
- 3中判断无有向环，返回权值。

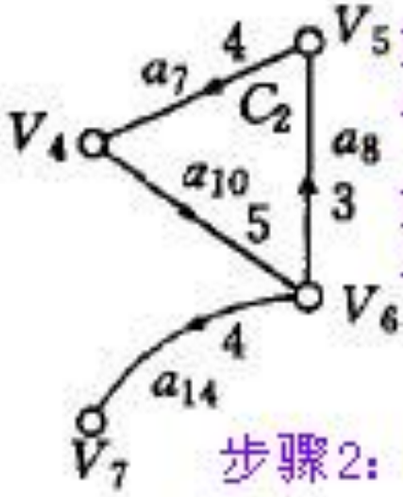
最小树型图构造流程



(a) D

步骤1：求最短弧集





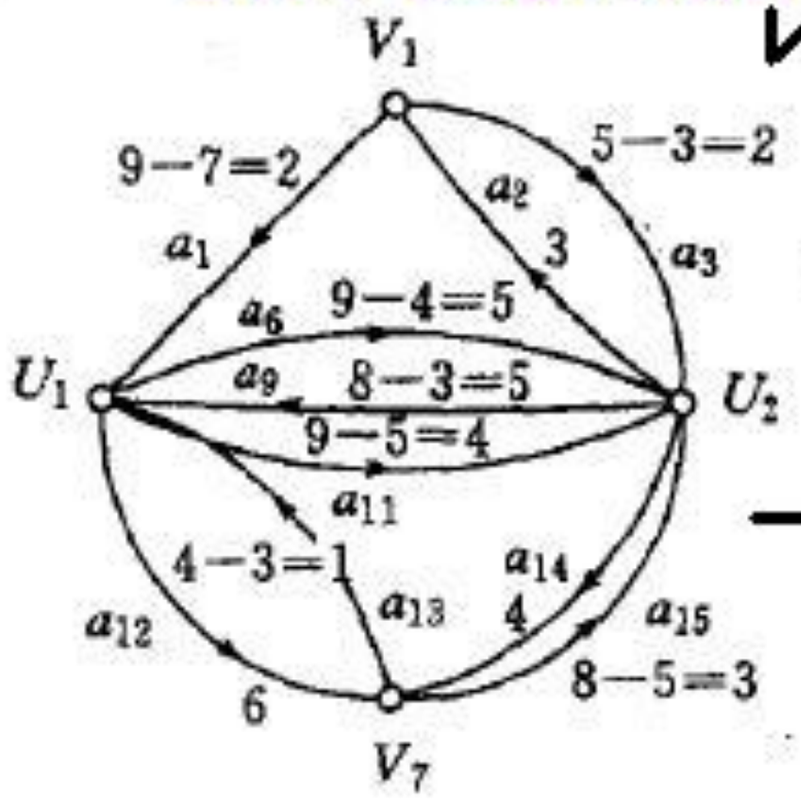
如果无圈又无收缩点，A0就是答案

如果有除了V1之外的孤立点，则无解

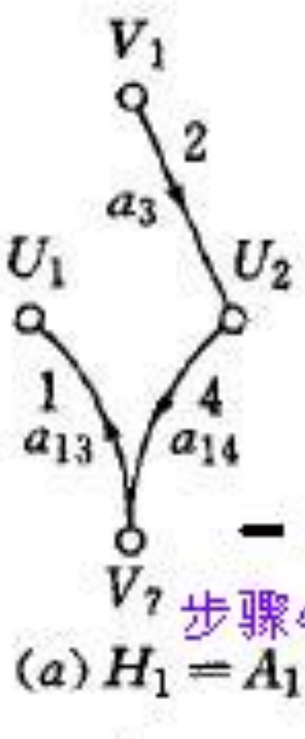
(b) A_0

步骤2：检查有向圈和收缩点

步骤3：收缩圈，构新图

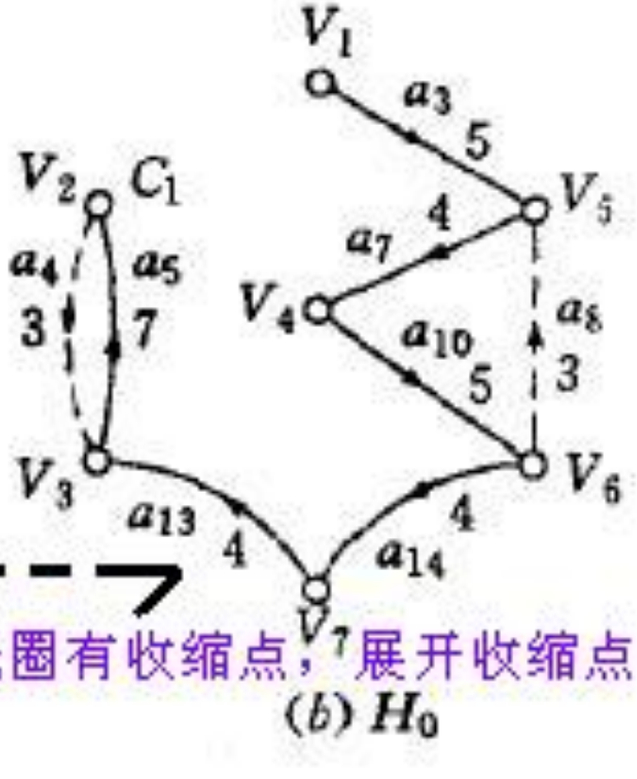


有圈继续收缩



重复步骤1，2

(a) $H_1 = A_1$



步骤4：无圈有收缩点，展开收缩点

(b) H_0

其他

- 其他：
  - 如果没有指定固定根，可以增加一个虚拟根，并且为其增加V条边指向每个顶点，权值都为原图所有边权值之和加一，转化为固定根求解。

  - 朱流算法只能求解最小总权值，无法求出路径（缩点后原顶点编号改变）。

最小树形图

poj 3164 Command Network(最小树形图模板题)

平面图

平面图基本知识

- 什么是平面图?
- 顾名思义，就是在满足“任何两边除端点外无公共点”的前提下能在二维平面上画出的图。
- 平面图的边数M是O(N)级别的，具体来讲，M<=3N-6。
- 什么是平面图的对偶图?
- 把平面图的区域看做点，两个区域的公共边看做边，所构成的图。

平面图转对偶图的方法

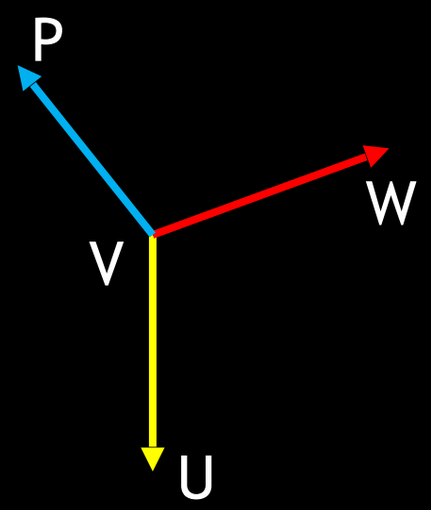
- 1.把原图中的每条线段看做双向向量。
- 2.选择一条未被标记的向量u→v。
- 3.找到与向量V→U逆(或顺)时针方向夹角最小的向量V→W。
- 4.如果向量V→W已经被标记，转到5，否则将它标记，重复3。
- 5.重复2~4直到所有向量都被标记。
- 6.通过记录每个区域的边，计算它的面积正负，可以判断它是有限大区域还是最外边的无限大区域。

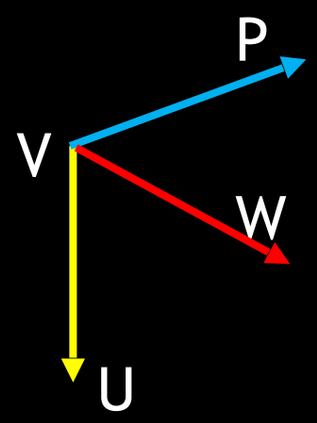
与向量V→U逆时针方向夹角最小的向量

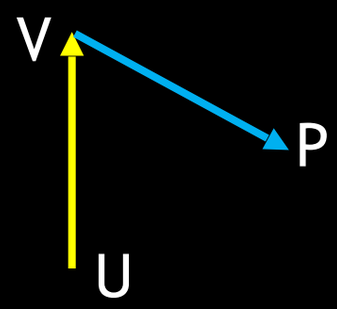
- 两种需要更新的情况：

- 当前夹角最小的向量V→P在V→U的顺时针方向（VU×VP叉积为负），当前向量V→W在V→u的逆时针方向（VU×VW叉积为正）。

- V→P和V→W在V→U同侧， V→W在V→P的顺时针方向（VP×VW叉积为负）。







例题：保护古迹

- http://www.lydsy.com:808/JudgeOnline/problem.php?id=2965

- 一个古迹被围起来，相当于它不能从它所在的区域到达最外边无限大的区域。

平面图最小割转最短路

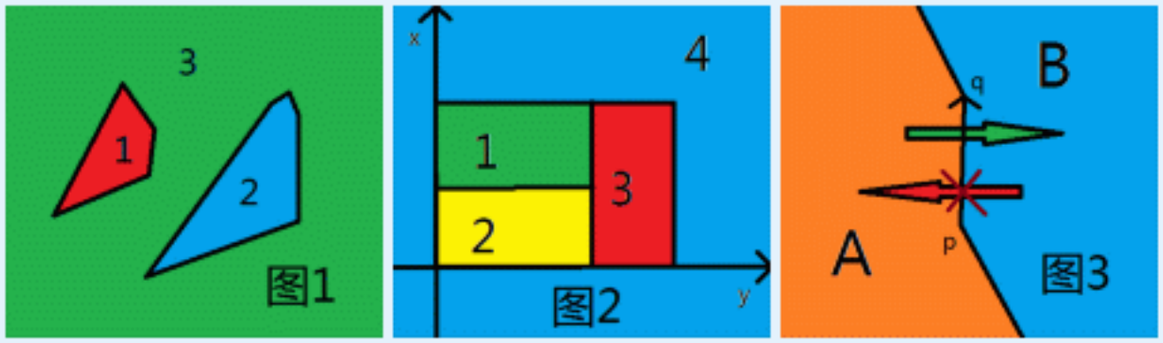
- 在原平面图中添加一条从起点S到终点T的边，会增加一个区域S’。
- 无限大的区域设为T’。
- 把加边后的平面图按照上面的方法转化为对偶图。
- 删去边S’-T’。
- 此时S-T的最小割大小等于S’到T’的最短路长度。
- 例题：Bzoj1001狼抓兔子、NOI2010海拔。

例题：跨平面

二维平面被划分成若干个简单多边形，每个简单多边形必与至少一个多边形共边，且多边形两两之间公共面积为0。所有多边形的并为一个简单多边形（即不存在图1所示情况）。

区域定义为：一个多边形，或者多边形的并关于平面的补集。

如图2，黑框表示三个简单多边形的边界，该平面共有4个区域。



.

简单多边形的每条边<p, q>有两个方向(p->q和q->p)，每个方向有一个激活费用V，表示激活这条边的该方向要花费代价V。V=0则该边的该方向不可被激活。如图3，激活了该方向后，就能无数次从向量<p, q>的逆时针方向走到顺时针方向，但不能从向量<p, q>的顺时针方向走到逆时针方向，即从区域A能走到区域B，但不能从区域B走到区域A。

Wayne希望你能帮他找到一个区域a，使得任取一个区域b，都能从a出发到达b。求在此区域a下的最小总激活费用。保证存在这个区域。

- 对于100%的数据，n≤3000，区域数不超过1000，点坐标绝对值不超过1W，每条边激活费用不超过100。

拓展思考：如何构造平面图的输入数据?

- 1. 三角形组合再删除法

- 每次在平面上随机一点，向两个已有点连边（要求连的边不能与任何已有边交叉，可以通过枚举或者多次随机找到这样的边）。

- 当点的规模足够大时，开始随机删除图中一些边（注意要保证最后所有点的度数均大于一并且所有点连通），这样三角形就逐渐变为了多边形。

- 优点：生成的平面图棱角分明，大小均匀。

- 缺点：不容易生成形状奇怪的图形，时间复杂度比较高（三次方）。

拓展思考：如何构造平面图的输入数据?

- 2. 分层构造法。
- 首先在平面上画若干个同心圆。
- 在每两个圆之间随机产生一些点，并在相邻点之间连边。
- 擦掉这些圆，平面上就是一些同心多边形了。
- 在两层多边形之间，随机连一些边，注意判断不要交叉。
- 随机擦除一些同心多边形上的边（注意要保证最后所有点的度数均大于一）。
- 优点：时间复杂度比较低（平方级别），适合产生大规模数据，以及可以用来坑掉某些不规范程序的扭曲状多边形。
- 缺点：产生的图不够一般化，具有稀松分层特点。

拓展思考：如何构造平面图的输入数据?

- 3. 递归中心点法。
- 首先在平面上随机产生一个很大的简单多边形（可以在四个象限内各随机一些点，然后按极角排序，再连接相邻的点）。
- 在这个多边形内部随机新建一个点，随机向多边形的顶点连边，注意判断不要交叉。
- 找到这个点把多边形划分成的每个区域，递归进入各个区域按照上述办法生成。

- 优点：生成的图一般性最好，时间复杂度低（平方级别），适合产生大规模数据。
- 缺点：编程复杂度较高。

AC自动机

简介

- 全名叫做Aho-Corasick automation
- 该算法在1975年产生于贝尔实验室，是著名的多模匹配算法之一。
- 一个常见的例子就是给出n个单词，再给出一段包含m个字符的文章，让你找出有多少个单词在文章里出现过。

方法

- 方法：
  - 构造一棵Trie，作为AC自动机的搜索数据结构。
  - 构造fail指针，使当前字符失配时跳转到具有最长公共前后缀的字符继续匹配。如同 KMP算法一样， AC自动机在匹配时如果当前字符匹配失败，那么利用fail指针进行跳转。由此可知如果跳转，跳转后的串的前缀，必为跳转前的模式串的后缀并且跳转的新位置的深度（匹配字符个数）一定小于跳之前的节点。所以我们可以利用 bfs在 Trie上面进行 fail指针的求解。
  - 扫描主串进行匹配。
- 例子：在包含五个单词：say she shr he her 的AC自动机中查找一个字符串包含了哪几个单词

构建Trie

- 首先我们需要建立一棵Trie。但是这棵Trie不是普通的Trie，而是带有一些特殊的性质。

- 首先会有3个重要的指针，分别为p, p->fail, temp。
  - 指针p，指向当前匹配的字符。若p指向root，表示当前匹配的字符序列为空。（root是Trie入口，没有实际含义）。
  - 指针p->fail，p的失败指针，指向与字符p相同的结点，若没有，则指向root。
  - 指针temp，测试指针（自己命名的，容易理解！~），在建立fail指针时有寻找与p字符匹配的结点的作用，在扫描时作用最大，也最不好理解。

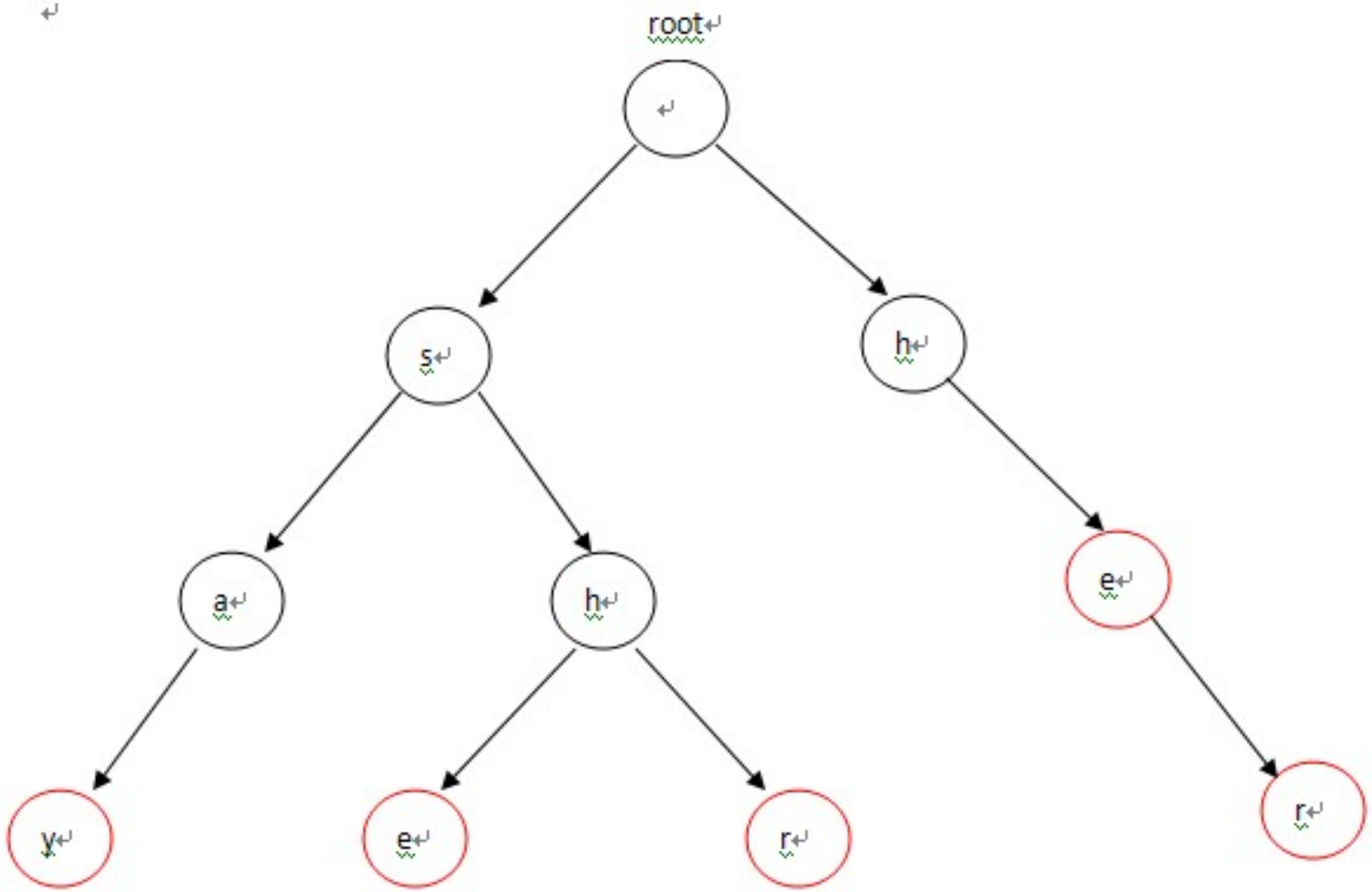
构建Trie

- 对于Trie树中的一个节点，对应一个序列s[1...m]。此时，p指向字符s[m]。若在下一个字符处失配，即p->next[s[m+1]] == NULL，则由失配指针跳到另一个节点(p->fail)处，该节点对应的序列为s[i...m]。若继续失配，则序列依次跳转直到序列为空或出现匹配。在此过程中，p的值一直在变化，但是p对应节点的字符没有发生变化。在此过程中，我们观察可知，最终求得得序列s则为最长公共后缀。另外，由于这个序列是从root开始到某一节点，则说明这个序列有可能是某些序列的前缀。

- 再次讨论p指针转移的意义。如果p指针在某一字符s[m+1]处失配（即p->next[s[m+1]] == NULL），则说明没有单词s[1…m+1]存在。此时，如果p的失配指针指向root，则说明当前序列的任意后缀不会是某个单词的前缀。如果p的失配指针不指向root，则说明序列s[i...m]是某一单词的前缀，于是跳转到p的失配指针，以s[i...m]为前缀继续匹配s[m+1]。

- 对于已经得到的序列s[1...m]，由于s[i...m]可能是某单词的后缀，s[1...j]可能是某单词的前缀，所以s[1...m]中可能会出现单词。此时，p指向已匹配的字符，不能动。于是，令temp = p，然后依次测试s[1...m], s[i...m]是否是单词。

构建Trie

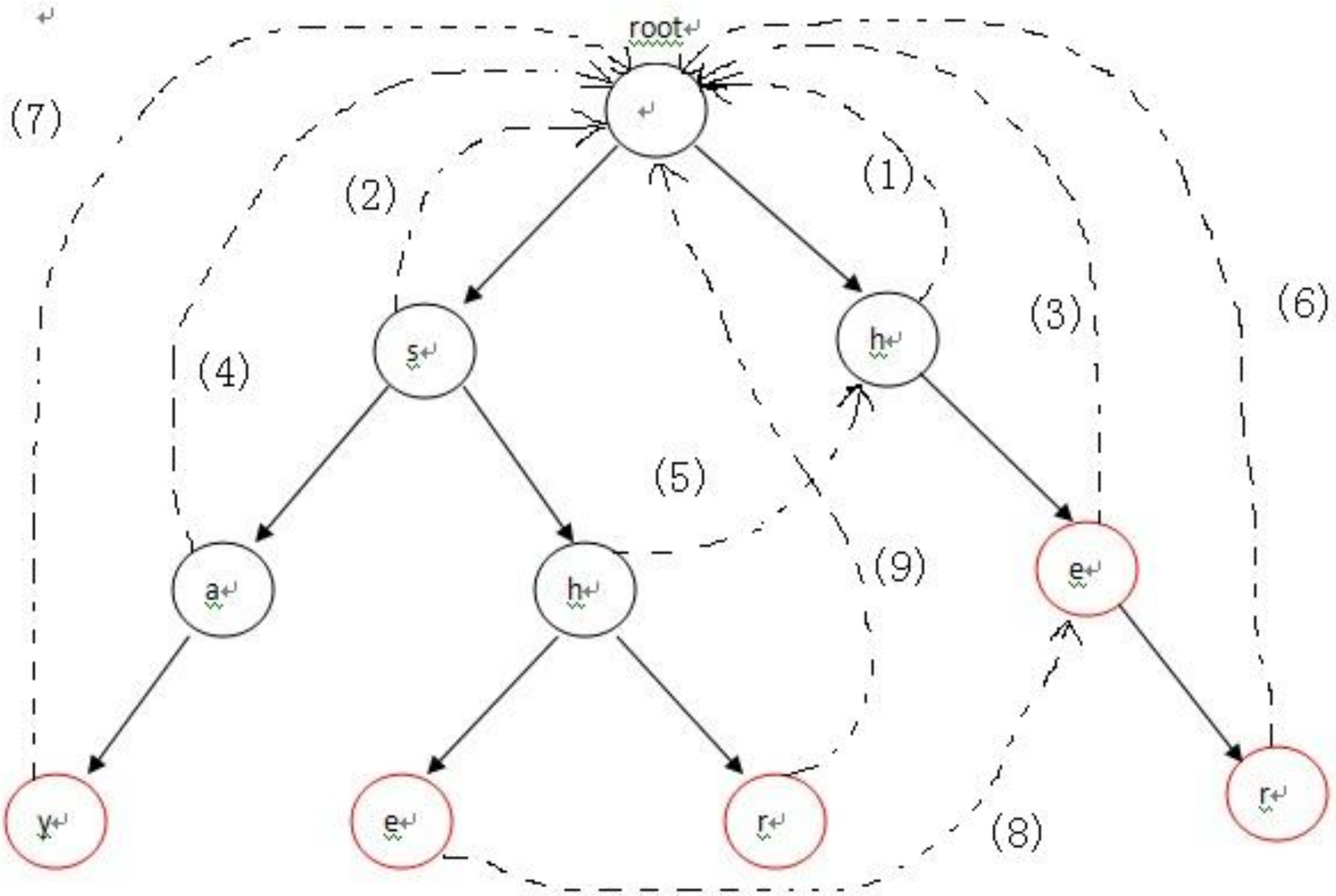


失败指针

- 用BFS来构造失败指针，与KMP算法相似的思想。

- 首先，root入队，第1次循环时处理与root相连的字符，也就是各个单词的第一个字符h和s，因为第一个字符不匹配需要重新匹配，所以第一个字符都指向root（root是Trie入口，没有实际含义）失败指针的指向对应下图中的(1)，(2)两条虚线；第2次进入循环后，从队列中先弹出h，接下来p指向h节点的fail指针指向的节点，也就是root；p=p->fail也就是p=NULL说明匹配序列为空，则把节点e的fail指针指向root表示没有匹配序列，对应图-2中的(3)，然后节点e进入队列；第3次循环时，弹出的第一个节点a的操作与上一步操作的节点e相同，把a的fail指针指向root，对应图-2中的(4)，并入队；第4次进入循环时，弹出节点h(图中左边那个)，这时操作略有不同。由于p->next[i]!=NULL(root有h这个儿子节点，图中右边那个)，这样便把左边那个h节点的失败指针指向右边那个root的儿子节点h，对应图-2中的(5)，然后h入队。以此类推：在循环结束后，所有的失败指针就是图-2中的这种形式。

失败指针



扫描

- 构造好Trie和失败指针后，我们就可以对主串进行扫描了。这个过程和KMP算法很类似，但是也有一定的区别，主要是因为AC自动机处理的是多串模式，需要防止遗漏某个单词，所以引入temp指针。

- 匹配过程分两种情况：(1)当前字符匹配，表示从当前节点沿着树边有一条路径可以到达目标字符，此时只需沿该路径走向下一个节点继续匹配即可，目标字符串指针移向下个字符继续匹配；(2)当前字符不匹配，则去当前节点失败指针所指向的字符继续匹配，匹配过程随着指针指向root结束。重复这2个过程中的任意一个，直到模式串走到结尾为止。

- 对照上图，看一下模式匹配这个详细的流程，其中模式串为yasherhs。对于i=0,1。Trie中没有对应的路径，故不做任何操作；i=2,3,4时，指针p走到左下节点e。因为节点e的count信息为1，所以cnt+1，并且讲节点e的count值设置为-1，表示改单词已经出现过了，防止重复计数，最后temp指向e节点的失败指针所指向的节点继续查找，以此类推，最后temp指向root，退出while循环，这个过程中count增加了2。表示找到了2个单词she和he。当i=5时，程序进入第5行，p指向其失败指针的节点，也就是右边那个e节点，随后在第6行指向r节点，r节点的count值为1，从而count+1，循环直到temp指向root为止。最后i=6,7时，找不到任何匹配，匹配过程结束。

Trie图

- 注意到，在AC自动机匹配时，每一步可能有多次回溯（直到找到一个可以继续匹配的结点）。那么我们是否可以在之前就完成一些工作，使每步只需要转移一次呢?

- 我们引入Trie图的概念。Trie图是一个确定性有限状态自动机（DFA），比起AC自动机，增加了确定性的属性。即在任一位置，输入任一字符集中的字符，都可以转移到一个确定的结点。

- 实现方法很简单，就是把getFail()中的 if(!u) continue; 改成 if(!u) {ch[k][c]=ch[f[k]][c];continue;} ，然后就可以把Find()中 while(u && !ch[u][c]) u=f[u]; 删去。实质上就是通过添加有向边，使所有的转移统一化。这样做的直接好处就是在AC自动机上动规时转移比较方便。

POJ2778 DNA Sequence

- 题目给m个病毒串，问不包含病毒串的长度n的DNA片段有几个。

BZOJ 2434 阿狸的打字机

- NOI

- http://www.lydsy.com/JudgeOnline/problem.php?id=2434

Hdu 3247 Resource Archiver

- http://acm.hdu.edu.cn/showproblem.php?pid=3247
- 题目大意: 给定n个代码串，给定m个病毒串，让我们将n个代码串连起来，中间可以重叠，连接起来的代码串不能含病毒串，问最短的的连接字符串长度。n<=10,m<=1000,单代码串长度<=1000,病毒穿总长度<=5万