


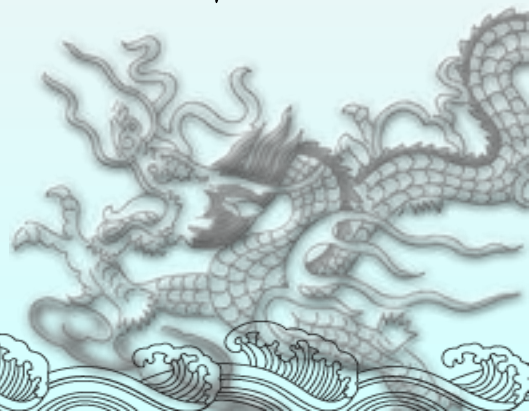
SAM入门——从根开始

吴作同 (E.Space)



符号声明

- ◆ S: 字符串
- ◆ |S|: S的长度
- ◆ S[i]: S的第i个字母, i从1开始编号
- ◆ S[l,r]: S从第l个字母到第r个字母组成的子串
- ◆ 字符串对应的节点: 用自动机从Root开始匹配完字符串后到达的节点
- ◆ (跳过构造思路请走这里) 



什么是SAM？

- ◆ SAM (Suffix Automaton Machine)，中文名后缀自动机，是一个以一个字符串的所有后缀为接受状态的有限状态自动机。



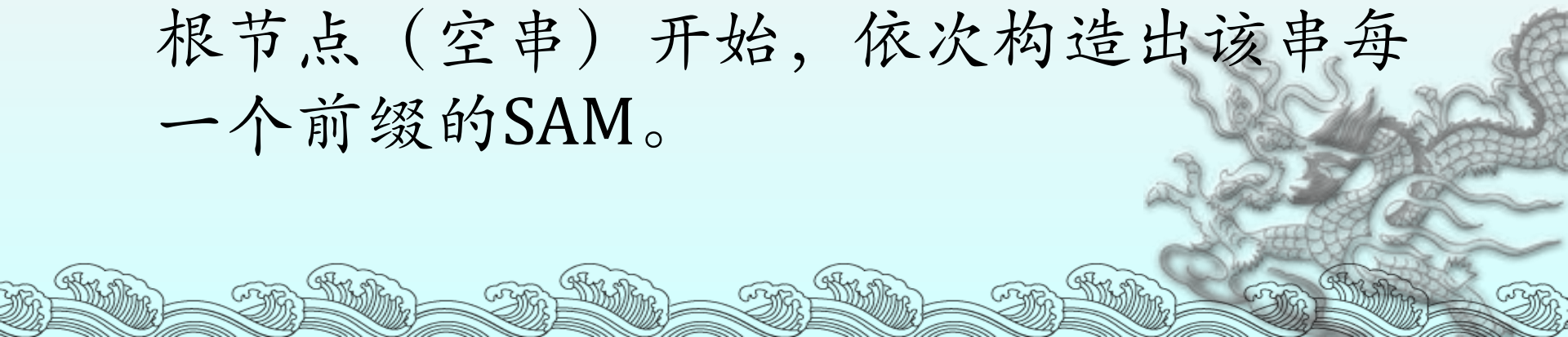
Substring Automaton Machine

- ◆ 因为子串是后缀的一个前缀，所以若把所有节点都视为接受状态，SAM就可以接受一个字符串的所有子串。
- ◆ 本文中讲的构造SAM和SAM的性质都是建立在此之上的。



构造SAM

- ◆ 我们想想如何构造一个这样的SAM。
- ◆ SAM接受了一个非空串的所有子串，肯定也接受了该串去掉最后一个字母后的字符串的所有子串。
- ◆ 空串的SAM显然就是一个根节点。
- ◆ 所以，对于一个字符串，我们可以从这个根节点（空串）开始，依次构造出该串每一个前缀的SAM。



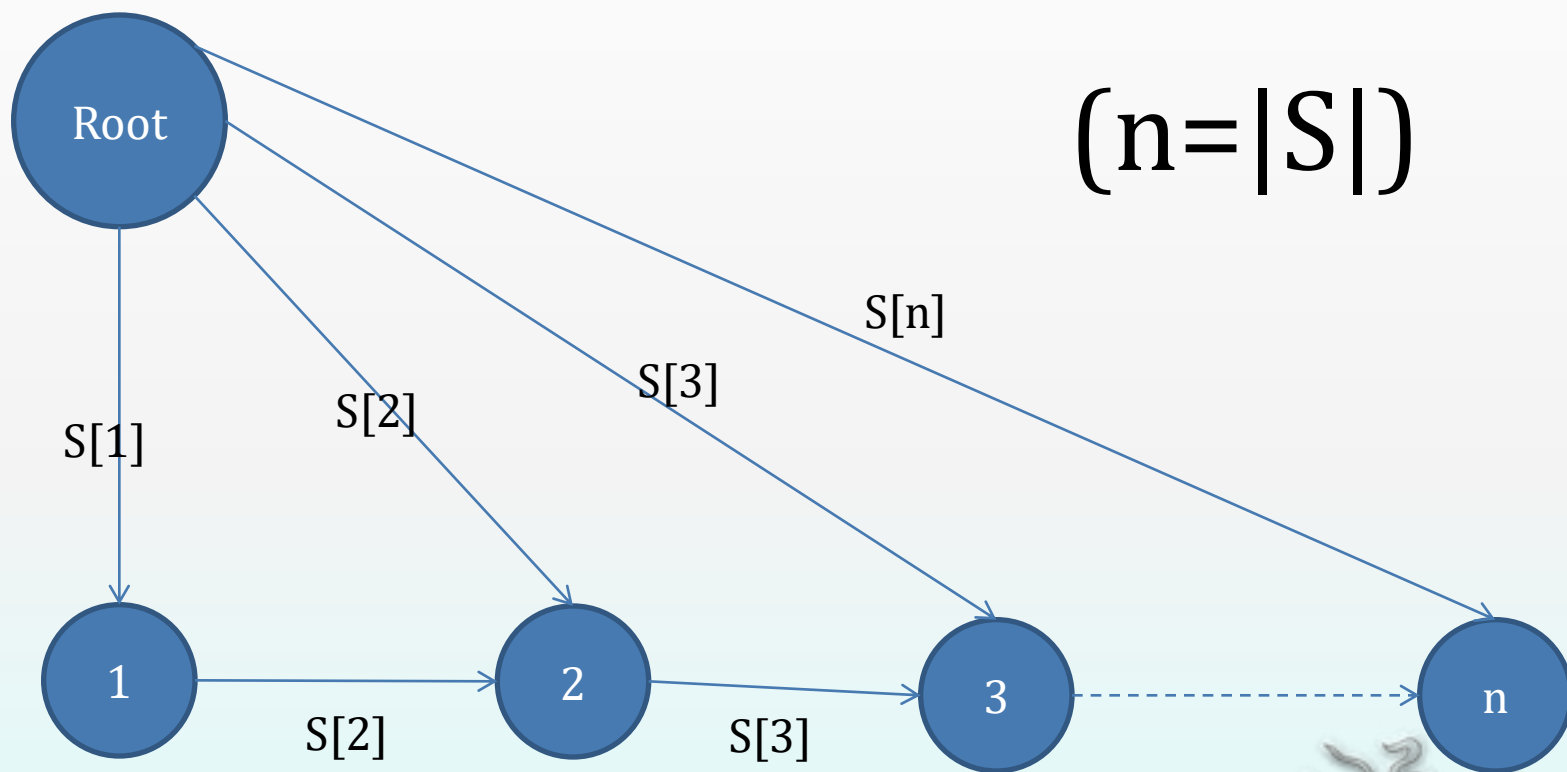
构造SAM

- ◆ 在一个长度为 l 的字符串后加入一个字符，那么该字符串就比原来多了 $(l+1)$ 个子串。
- ◆ 这些子串是由原串的 $(l+1)$ 个后缀（包括空串）之后加上新加入的字符得到。
- ◆ 所以我们很容易想到一种方法，就是用第 i 号节点来接受第 i 个前缀的所有后缀。
- ◆ 对于一个串 S ，我们得到了如下的结构：



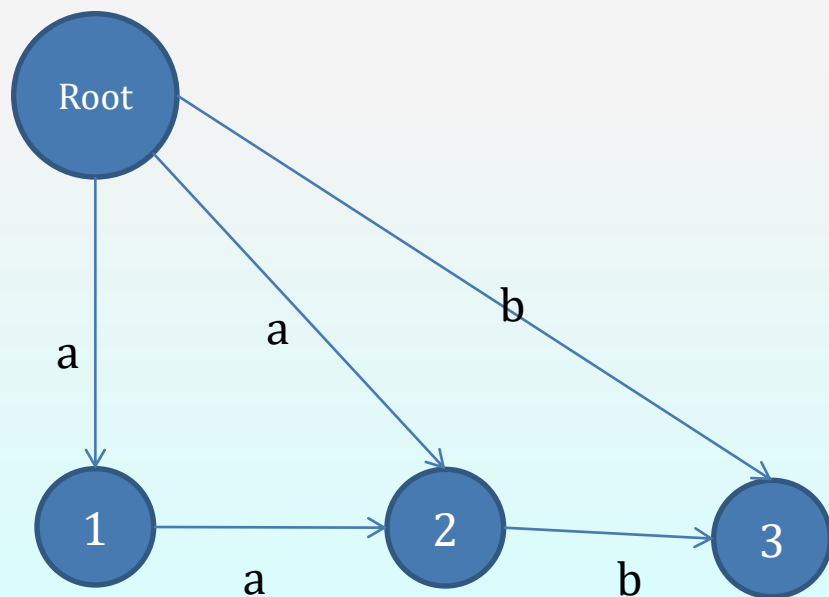
构造SAM

$(n=|S|)$



构造SAM

- 但是这个结构有个问题：从Root出发的同一字母的转移可能有多条。这样就不能良好地进行匹配等工作。

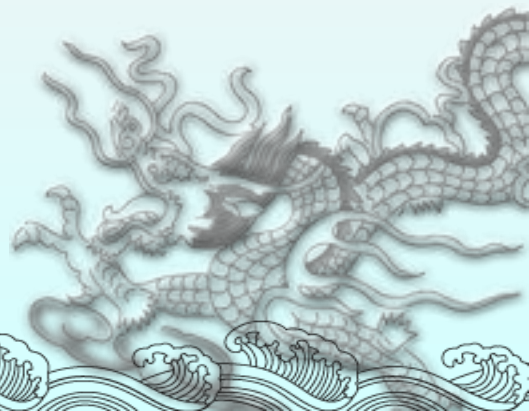


构造SAM

- ◆ 我们要进行一下改进。
- ◆ 我们原来是用第 i 号节点来接受第 i 个前缀的所有后缀，现在改成，对于第 i 个前缀，把它长度为 $0 \sim i$ 的后缀分成若干段不相交的区间，用若干个节点表示。为了方便找到所有后缀，我们这里定义一个叫做last的指针把表示相邻两个区间的节点连接起来。
- ◆ （在之后的图中用红色虚线箭头表示last指针）

构造SAM

- ◆ 现在一个节点上记录了以下信息：
- ◆ min,max: 这个节点接受长度在 $[\text{min}, \text{max}]$ 内的后缀
- ◆ next[]指针: 记录加入字母后转移到的状态（类似AC自动机）
- ◆ last指针: 记录表示前一个区间的后缀的节点



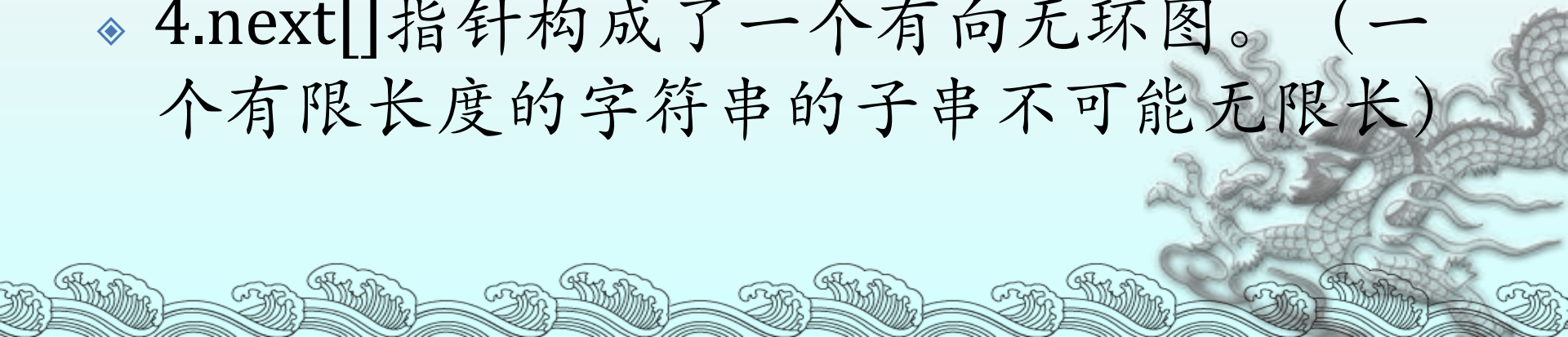
构造SAM

- ◆ 由定义我们可以得到以下性质：
- ◆ 1. last 指针指向的节点的 max 等于该节点的 $\text{min}-1$ 。（由区间划分得到）
- ◆ 2. Root 的 min 和 max 一定等于 0， Root 没有 last 指针。



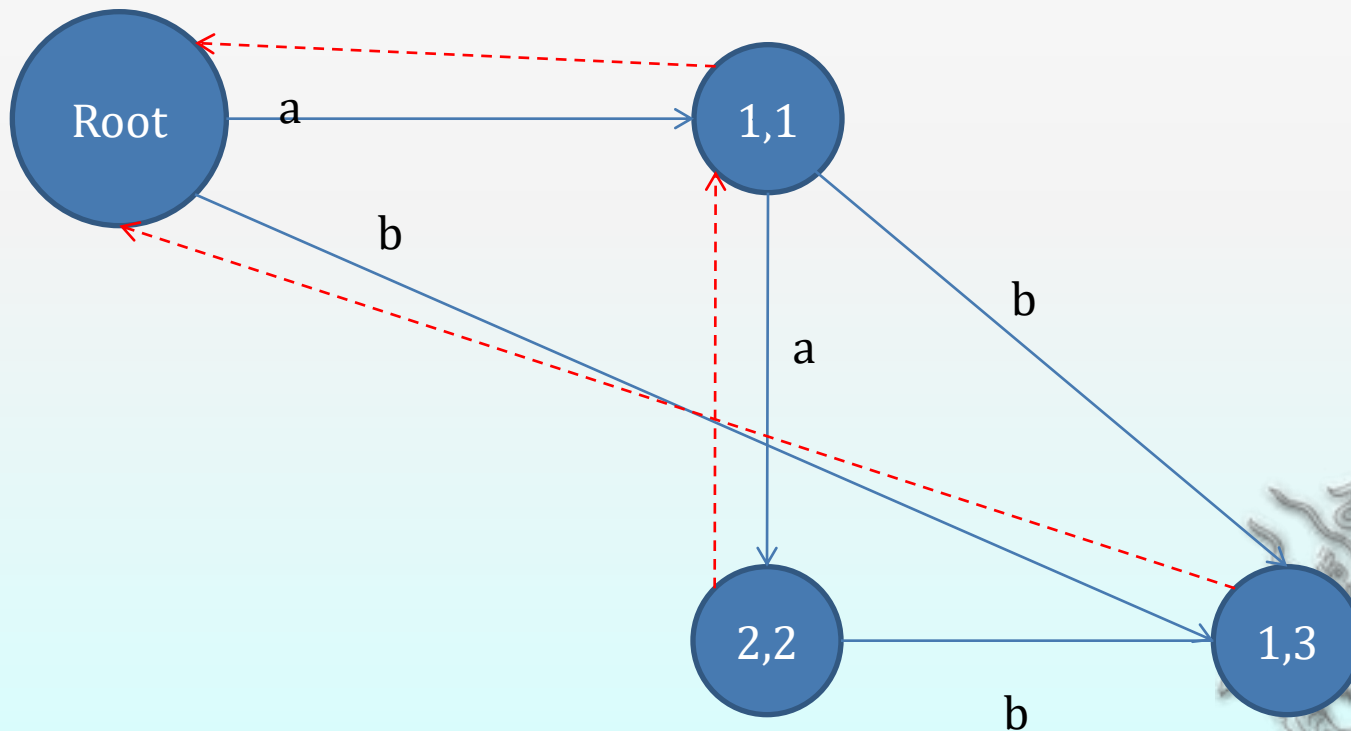
构造SAM

- ◆ 3.last指针构成了一棵以Root为根的树（由1,2得last指向的节点的max一定比该节点的max小，且最后指向根，不可能连出环）
- ◆ （所以我们定义一个点的父亲为last指向的节点，祖先为可通过last指针到达的节点的集合中的节点）
- ◆ 4.next[]指针构成了一个有向无环图。（一个有限长度的字符串的子串不可能无限长）



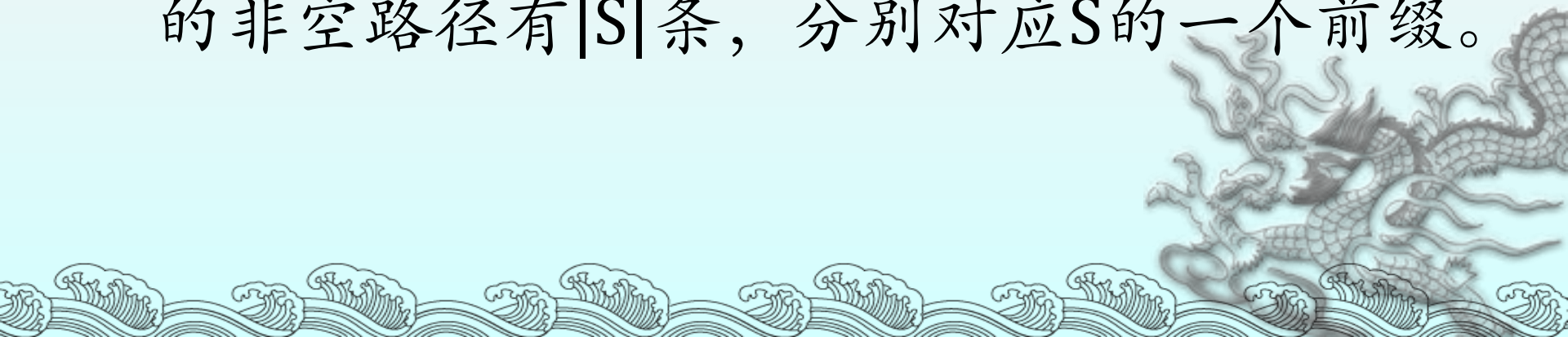
构造SAM

- 这里举一个 $S = \text{"aab"}$ 的例子：（节点上的数字表示该节点的min,max）



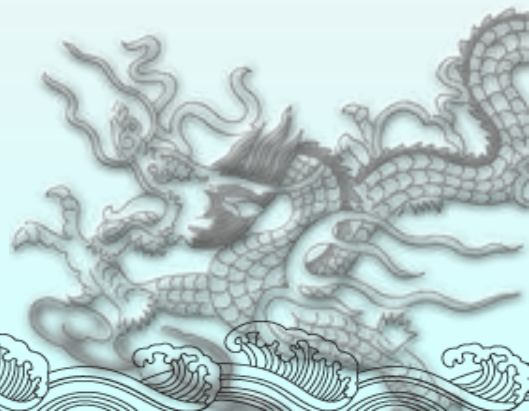
构造SAM

- ◆ 这里我把min和max的含义进行进一步的解释。
- ◆ 我们把SAM上的最长链称为“主链”。因为S最长的子串为本身，所以把主链上的字母按照链上的顺序连接，可以得到S。
- ◆ 对于S的SAM，从Root出发的完全在主链上的非空路径有 $|S|$ 条，分别对应S的一个前缀。

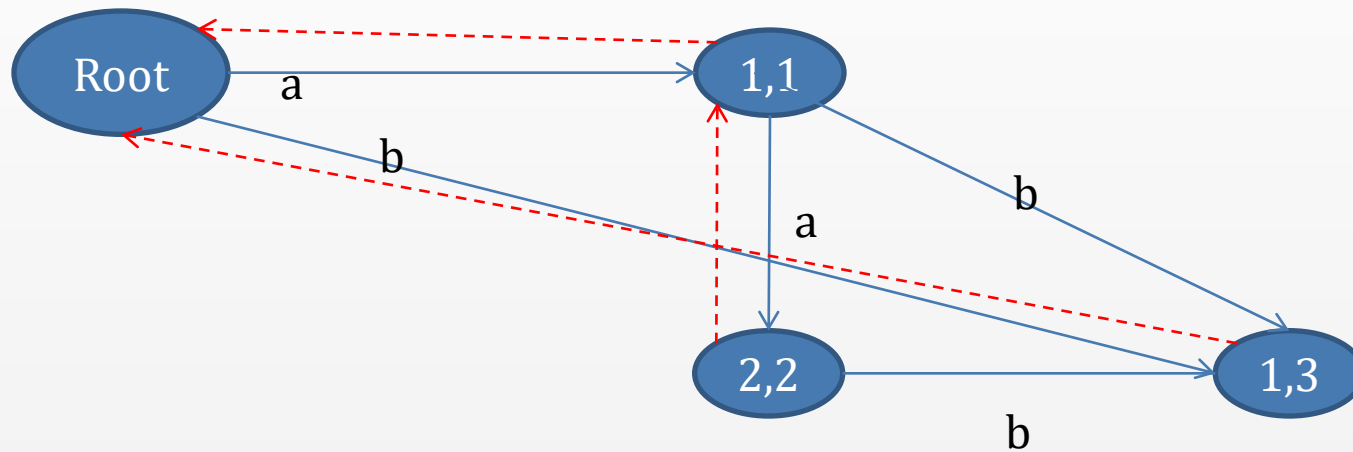


构造SAM

- ◆ 这些前缀的每个后缀，就对应着S的一个子串，一个长度为 l 的前缀，有 $(l+1)$ 个后缀，长度分别为 $0 \sim l$ 。
- ◆ 对于任意的 $i(0 \leq i \leq l)$ ，该前缀的长度为 i 的后缀在SAM上所对应的节点为该前缀所对应的节点的祖先中那个唯一满足 $\min \leq i \leq \max$ 的节点。




构造SAM



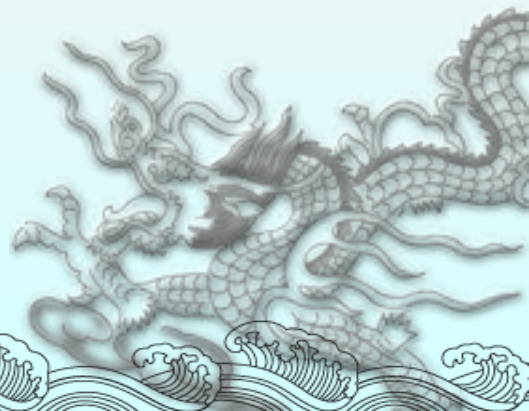
- ◆ 前缀aa对应的节点为(2,2)，它长度为2的后缀aa对应(2,2)，长度为1的后缀a对应祖先(1,1)，长度为0的后缀对应祖先Root。
- ◆ 前缀aab对应的节点为(1,3)，它长度为1,2,3的后缀b,ab,aab均对应(1,3)。

构造SAM

- ◇ 现在正式开始讲如何构造SAM。
- ◇ 我在一边讲的同时一边拿字符串 $S = \text{"abcbca"}$ 作例子。
- ◇ （不想看特例，直接看构造方法走这里）
- ◇ 我们还是从一个根节点开始。
- ◇ 首先我们有一个根节点Root, $S = \text{""}\text{"}$ 。



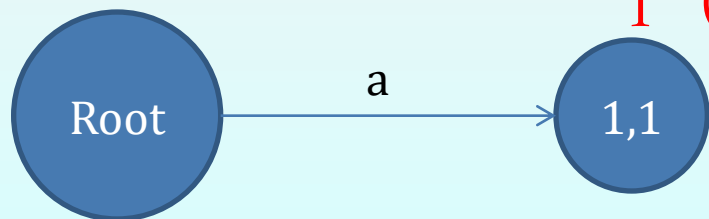
Root



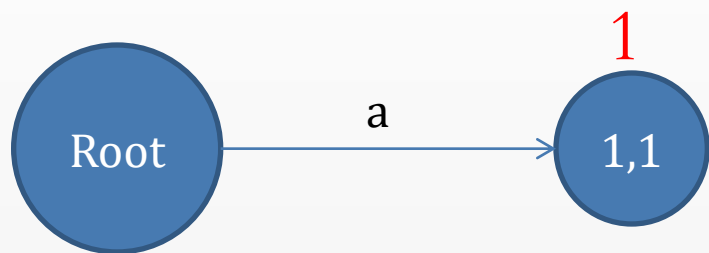
构造SAM

- ◆ 我们现在加入第一个字母a。
- ◆ 此时 $S=""$ 。
- ◆ S 的唯一后缀为空串，对应节点Root。
- ◆ 所以我们在Root后加入1号节点，连一条a的next边。在原来的 S 中Root对应长度为 $[0,0]$ 的后缀，所以1号节点对应后缀 $[1,1]$ 。

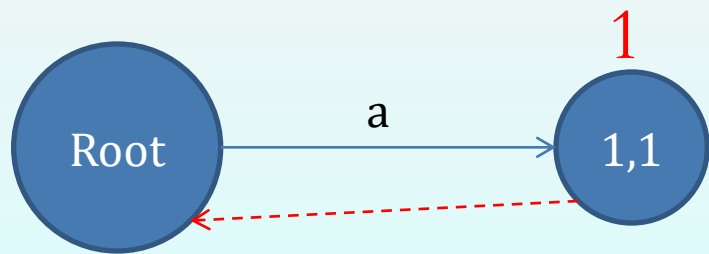
1（此后红色数字表示节点编号）



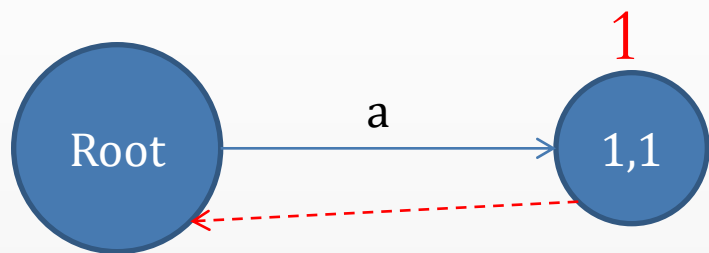
构造SAM



- ◆ 现在新的S还差长度为0的后缀，此时只要把1号点的last记为Root即可，由Root来接受这个后缀。

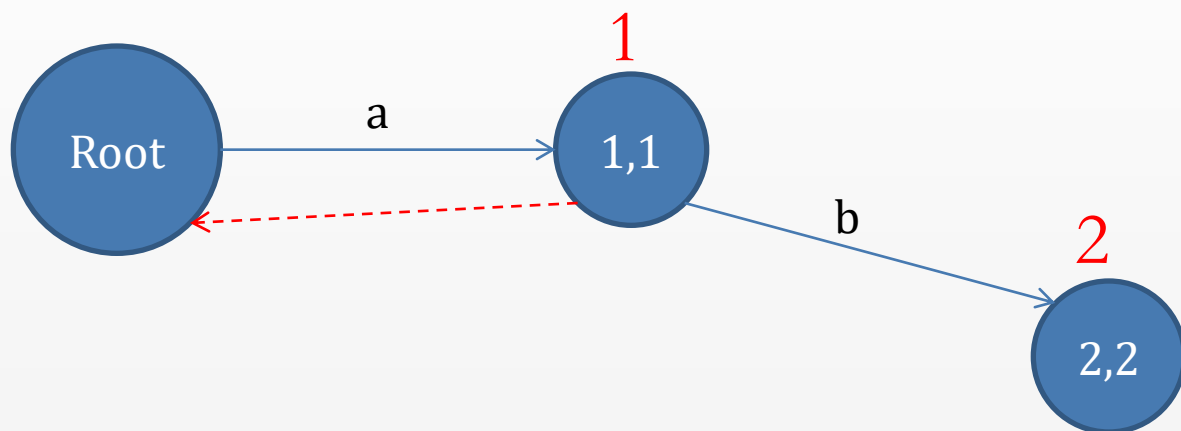


构造SAM



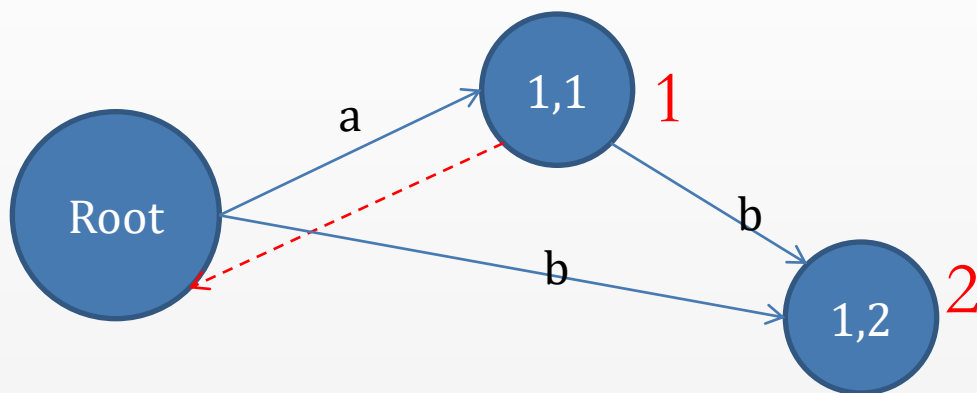
- ◇ 接下来加入第二个字母b。
- ◇ 现在 $S="a"$ 。
- ◇ 我们要加入3个后缀，我们从最长的开始。
- ◇ S 长度为1的后缀对应1号点，它还没有b的next边，所以我们把1号点用b的next边连向新建的2号点。此时2号点对应的后缀为 $[2,2]$ 。

构造SAM

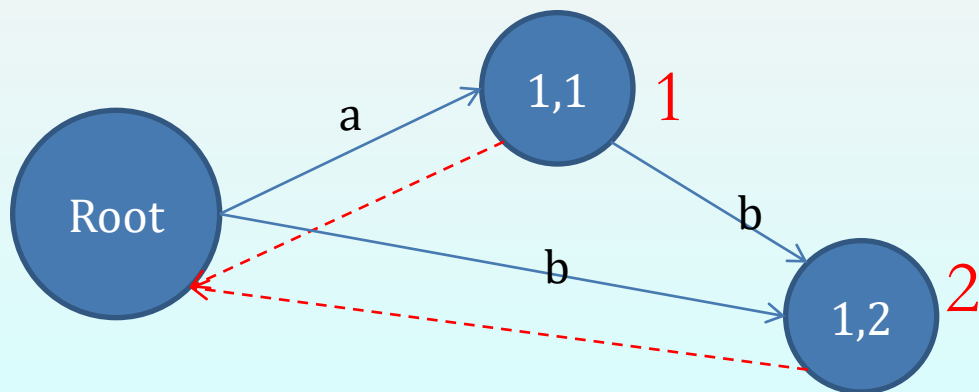


- ◇ 接下来加入长度为1的后缀。原来S中长度为0的后缀对应Root（1号点的last），此时Root并没有b的next边，所以我们把Root的next边连向2号点，现在2号点对应后缀[1,2]。

构造SAM

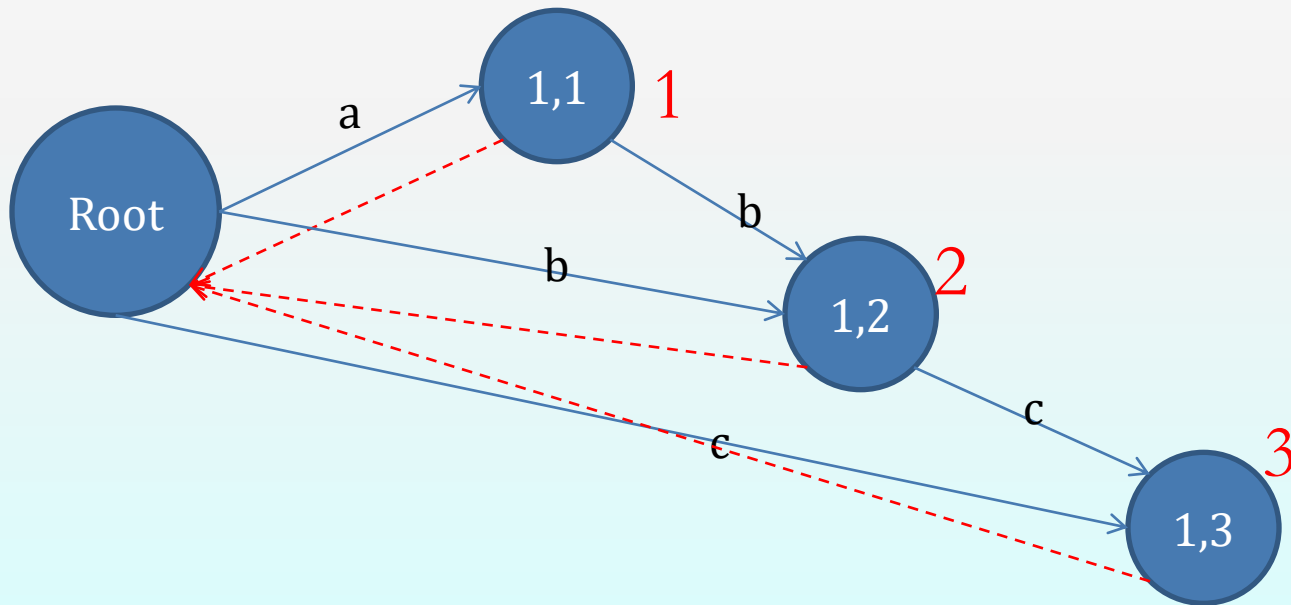


- ◆ 现在加入长度为0的后缀，只要把2号点的last设为Root即可。



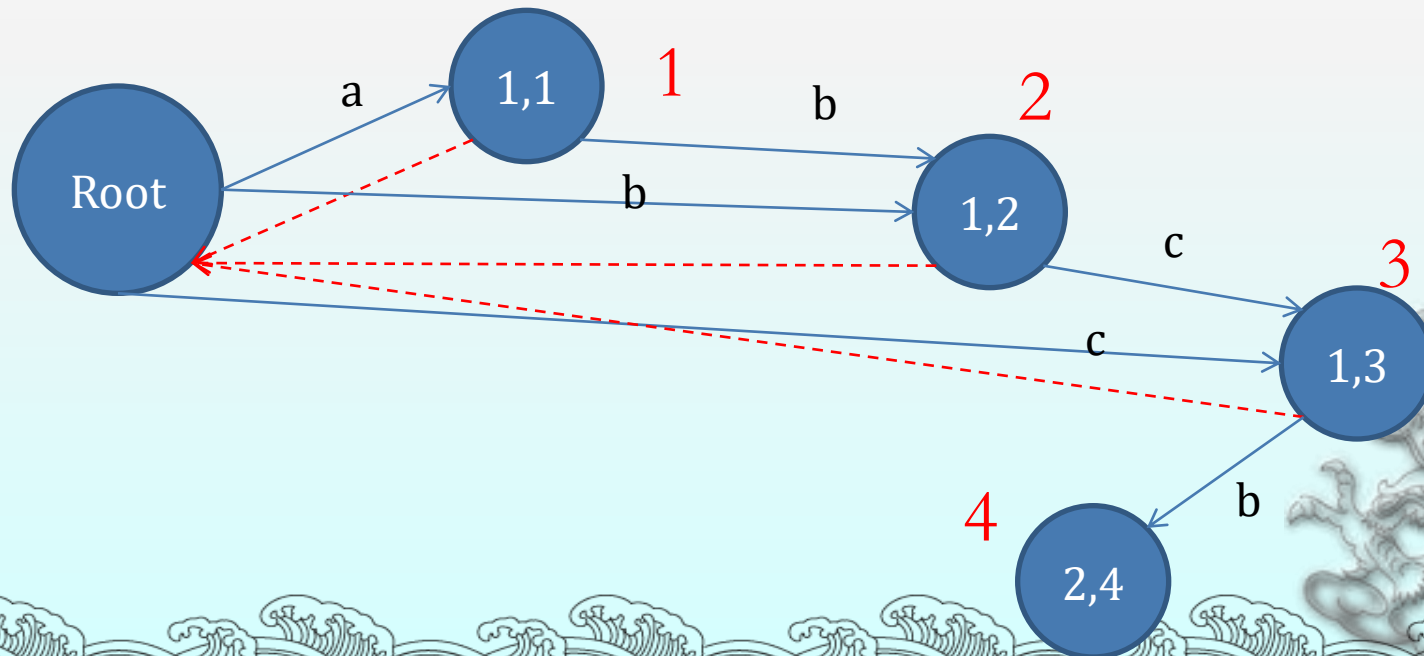
构造SAM

- ◇ 现在加入c， $S=\text{"ab"}$ 。
- ◇ 与加入b时同理，从主链的终点沿着last指针加边，由长到短加入后缀。



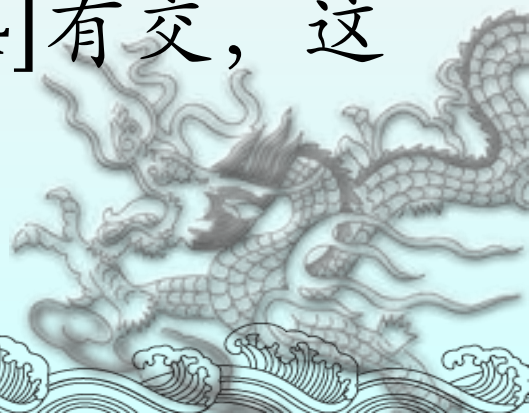
构造SAM

- ◆ 现在加入b, $S="abc"$ 。
- ◆ 把3号点连向4号点, 我们就成功加入了长度为2~4的3个后缀。
- ◆ 当要加入长度为1的后缀时, 我们发现Root已经有了一条b的next边。这条边指向2号点。



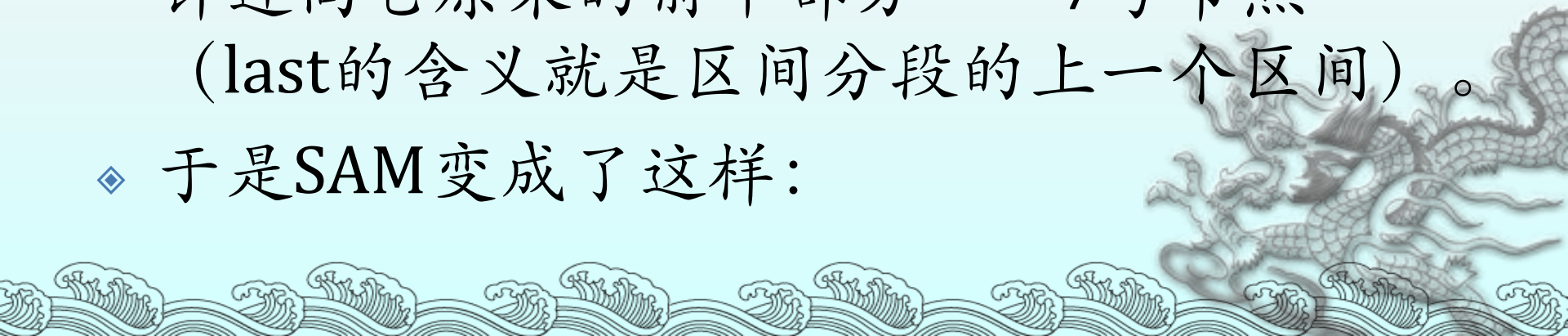
构造SAM

- ◆ 我们能不能用2号点来接受新的S的长度为1的后缀呢？
- ◆ 如果我们用了，那么4号点的last指针指向2号点。但是2号点对应的区间是 $[1,2]$ ，说明2号点除了表示一个长度为1的串，还表示为一个长度为2的串，我们需要的只有前者。
- ◆ 而且区间 $[1,2]$ 与4号点的区间 $[2,4]$ 有交，这不符合我们对last的定义。

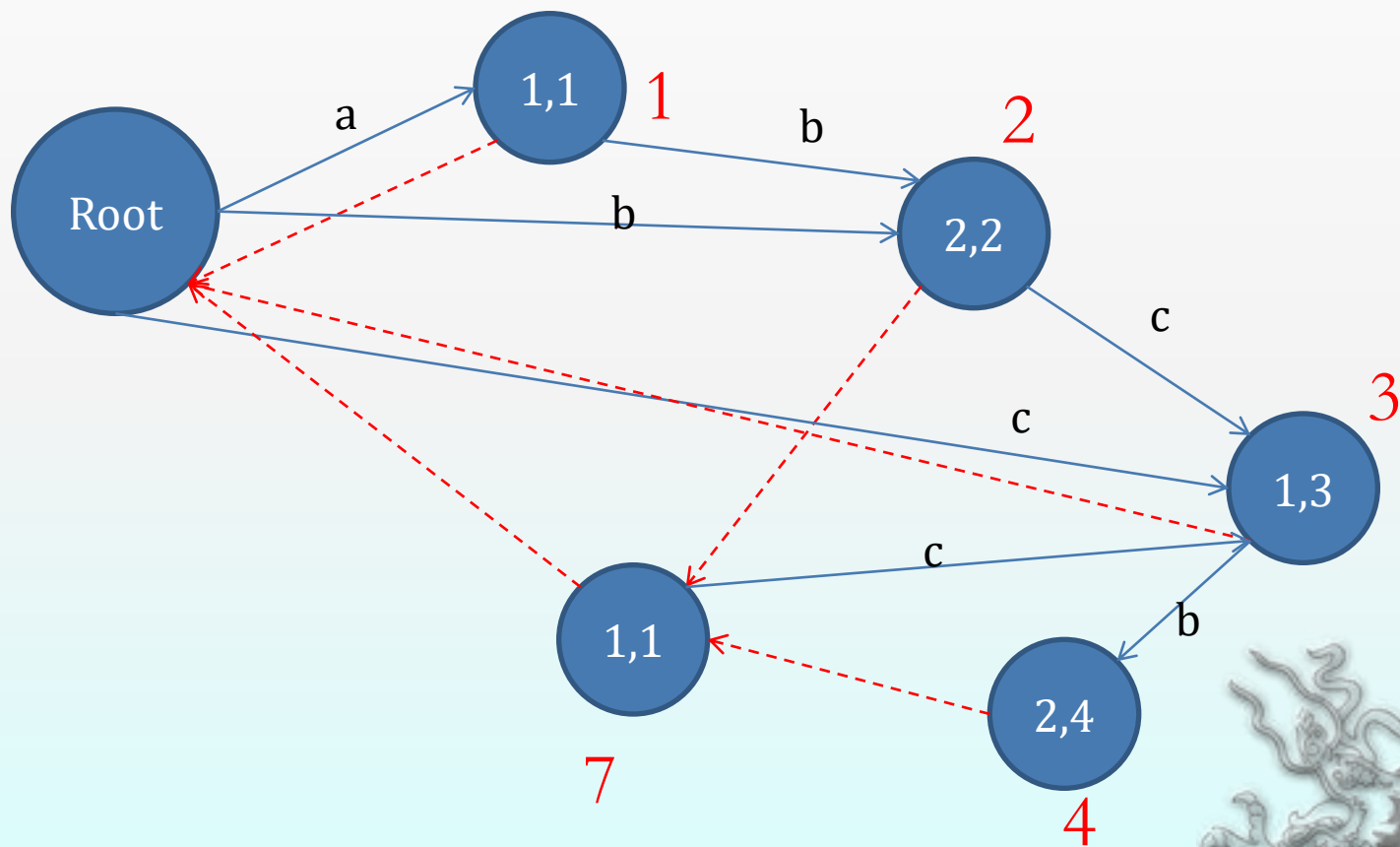


构造SAM

- ◆ 于是我们可以把2号点的区间拆开，新建一个7号点（至于为什么是7号之后再说），它的所有指针与2号点相同，表示区间 $[1,1]$ ，让2号点表示 $[2,2]$ 。这样我们就可以把4号点的last指针连向7号点。这时2号点的区间被改变，只有后半部分，于是把它的last指针连向它原来的前半部分——7号节点（last的含义就是区间分段的上一个区间）。
- ◆ 于是SAM变成了这样：



构造SAM



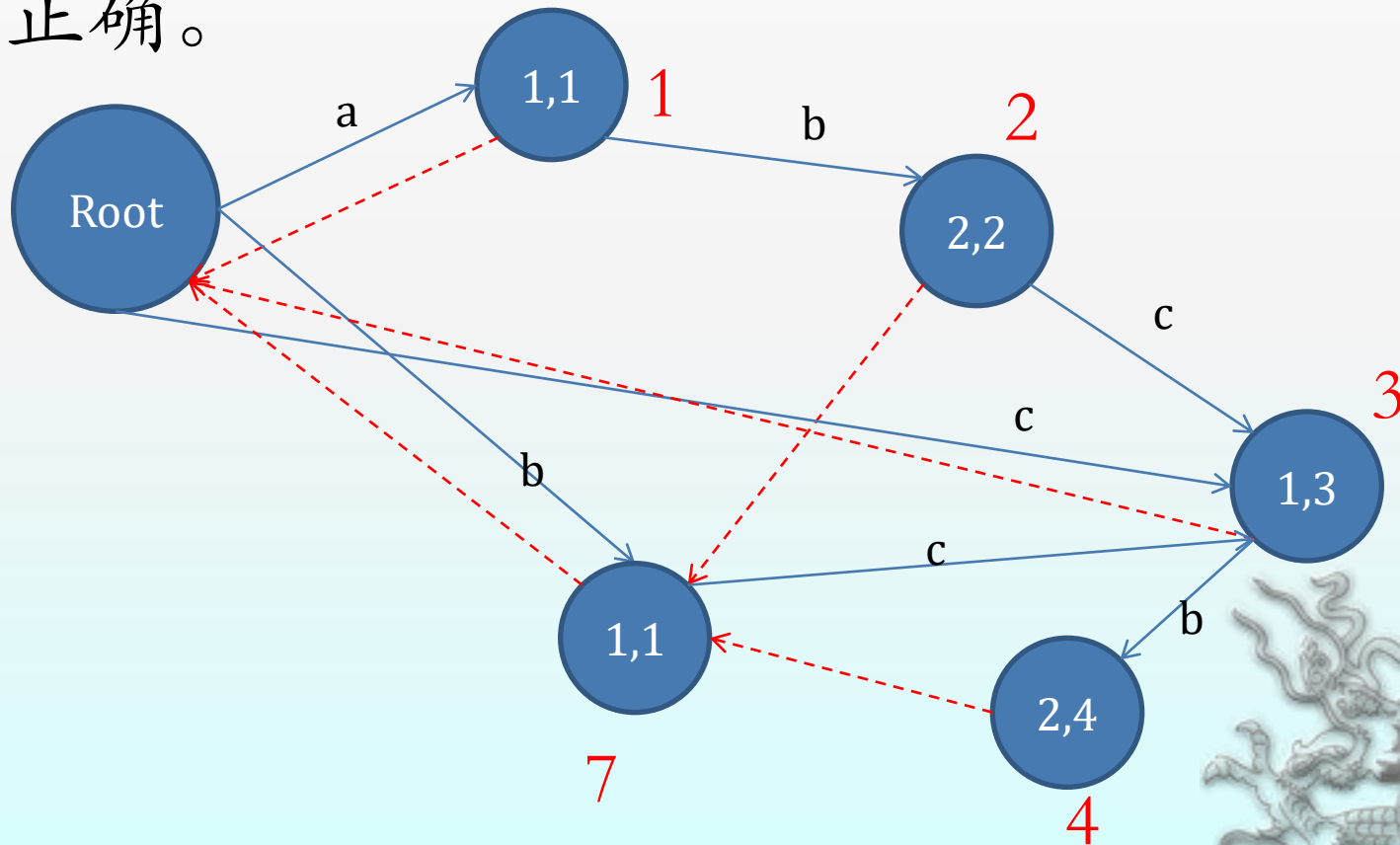
构造SAM

- ◆ 想想我们在新建2号点时做了些什么。
- ◆ 我们从1号点连了一条边给它，1号点的区间为 $[1,1]$ ，所有它有了 $[2,2]$ 的区间。
- ◆ 我们从Root连了一条边给它，Root的区间为 $[0,0]$ ，所以它有了 $[1,1]$ 的区间。
- ◆ 我们现在把2号点区间中1及1之前的部分给了7号点，所以我们要把Root和Root的所有祖先中连向2号点的b边全部连向7号点。



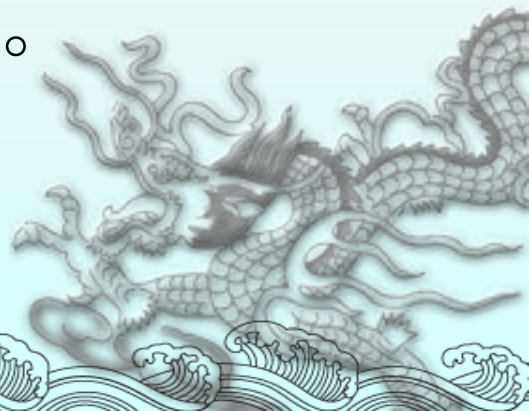
构造SAM

- 到此可以检查一下所有的min,max是否依然正确。



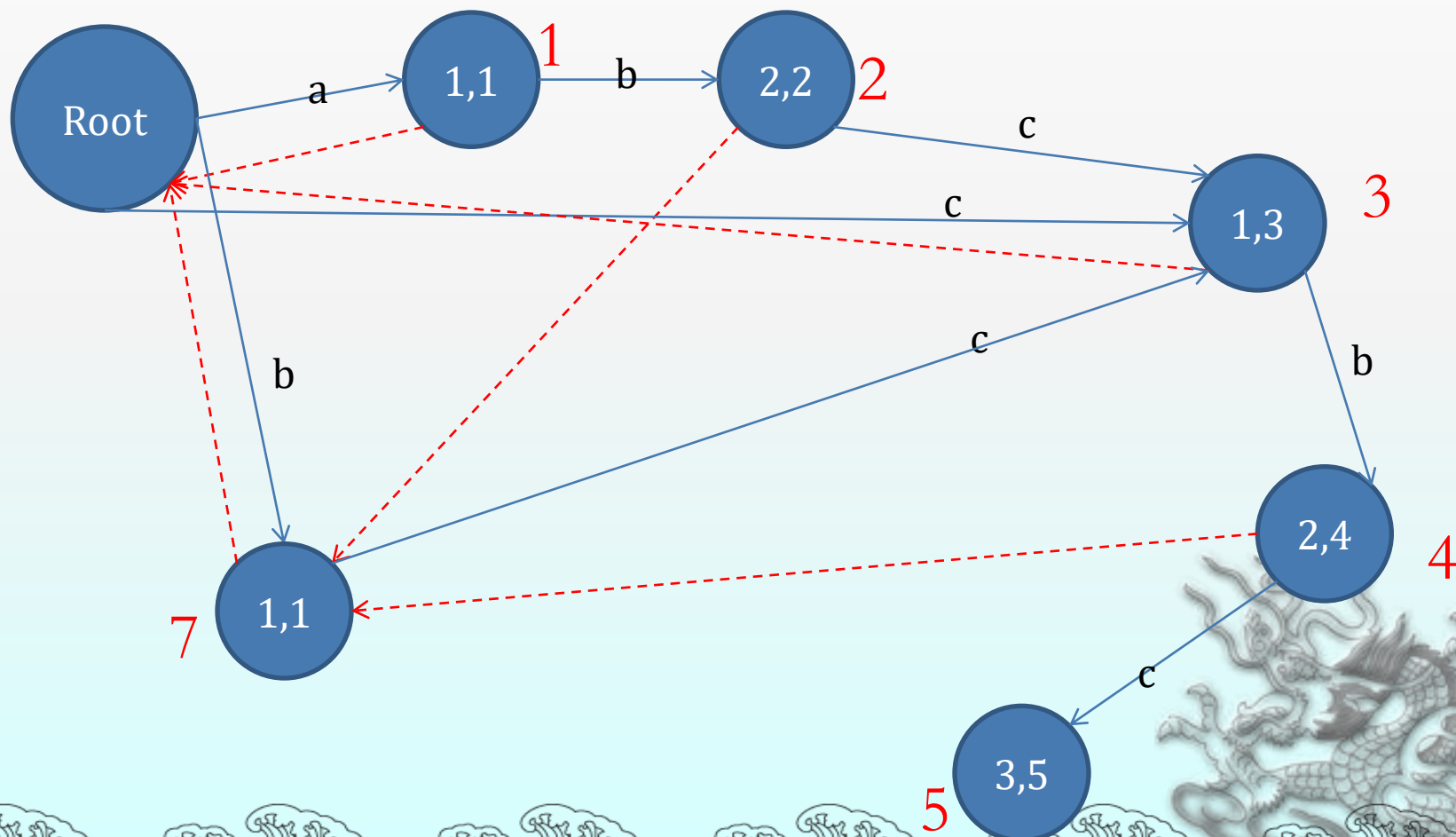
构造SAM

- ◆ 现在加入c, $S = \text{"abcb"}$ 。
- ◆ 我们新建5号点, 按照原先的规则加边。
- ◆ 当加到7号点时, 已经加了长度为 $[3, 5]$ 的后缀。我们发现7号点已经有c的next边, 它指向3。但3的区间是 $[1, 3]$, 我们只能把5号点的last指针指向 $[1, 2]$, 所以我们还得把3号点的区间拆开, 拆成 $[1, 2]$ 和 $[3, 3]$ 。



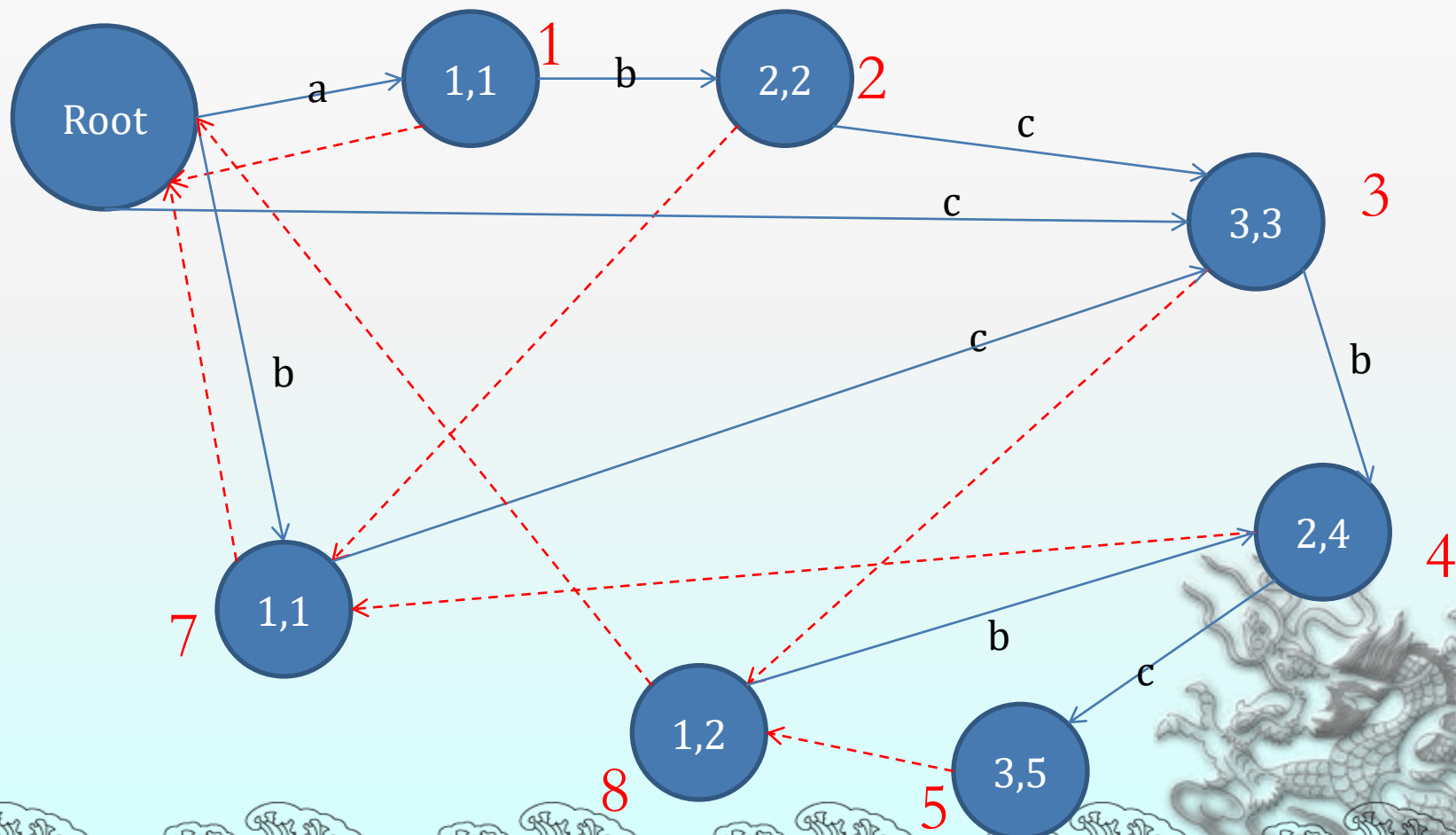
构造SAM

◇ 新建5号点



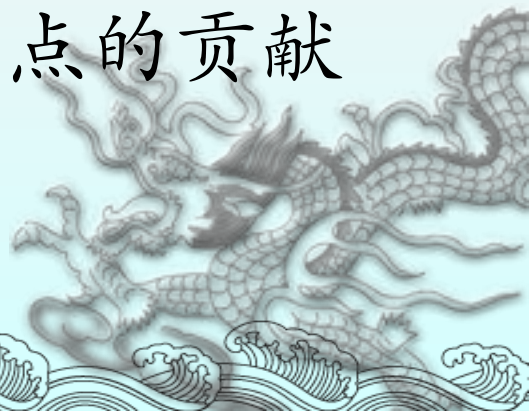
构造SAM

◇ 拆开3号点的区间（新建8号点复制3号点）



构造SAM

- ◆ 我们正在加入长度为2的后缀。由此我们现在知道两点：
- ◆ 1. 长度为3~5的后缀我们已经加好了。
- ◆ 2. 我们现在遇到的这个无法连出边（因为已连出一条c边）的节点（本例中是7号点）的max是1（=2-1）
- ◆ 由第2条可知7号点给被拆分的节点的贡献的区间右端点是2。



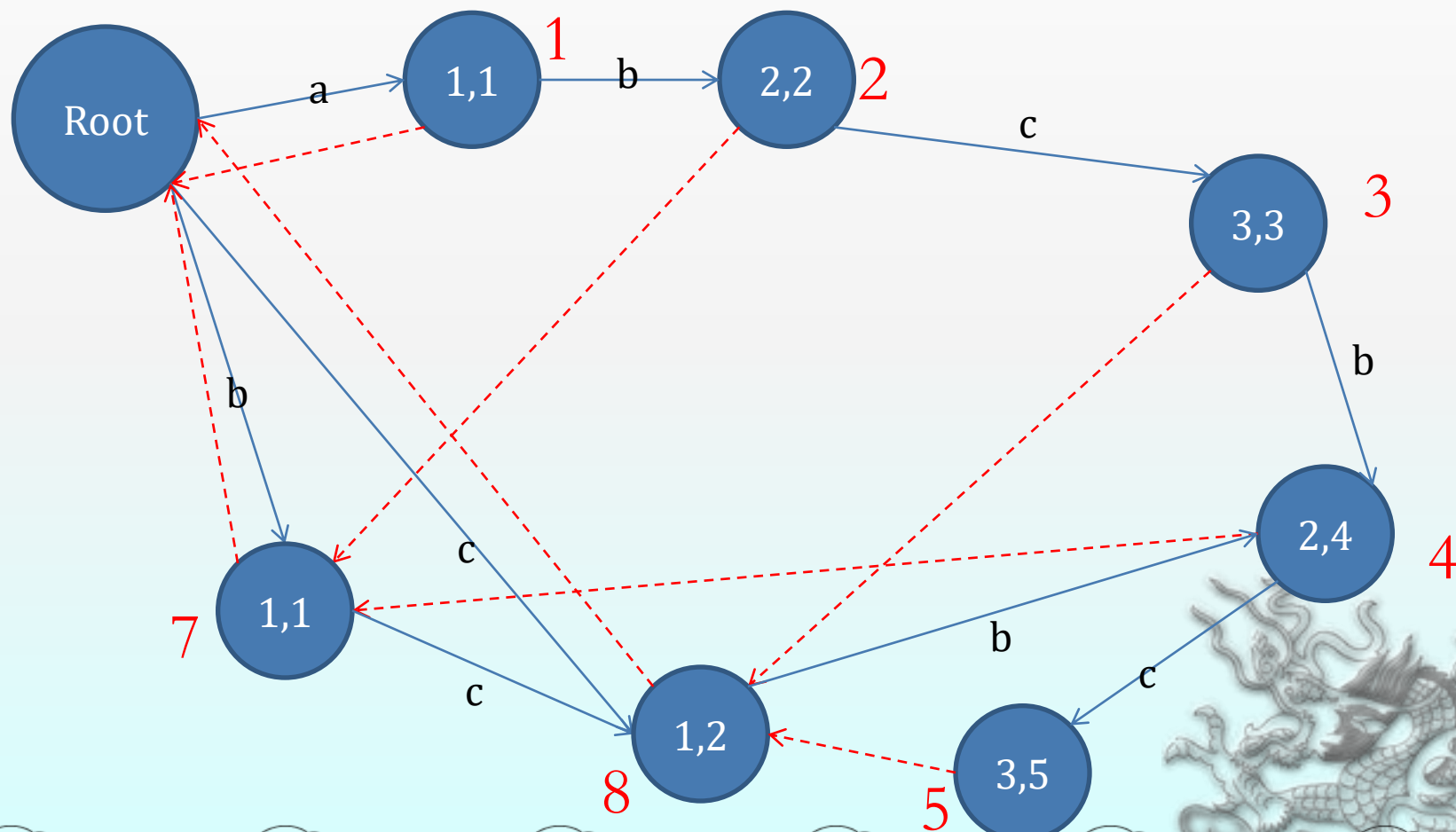
构造SAM

- ◆ 所以我们可以放心大胆地把该节点（7号点）及其祖先中所有连向3号点的c边连向8号点。
- ◆ （SAM的图见下页）



构造SAM

◆ 请再次检查min和max的正确性。

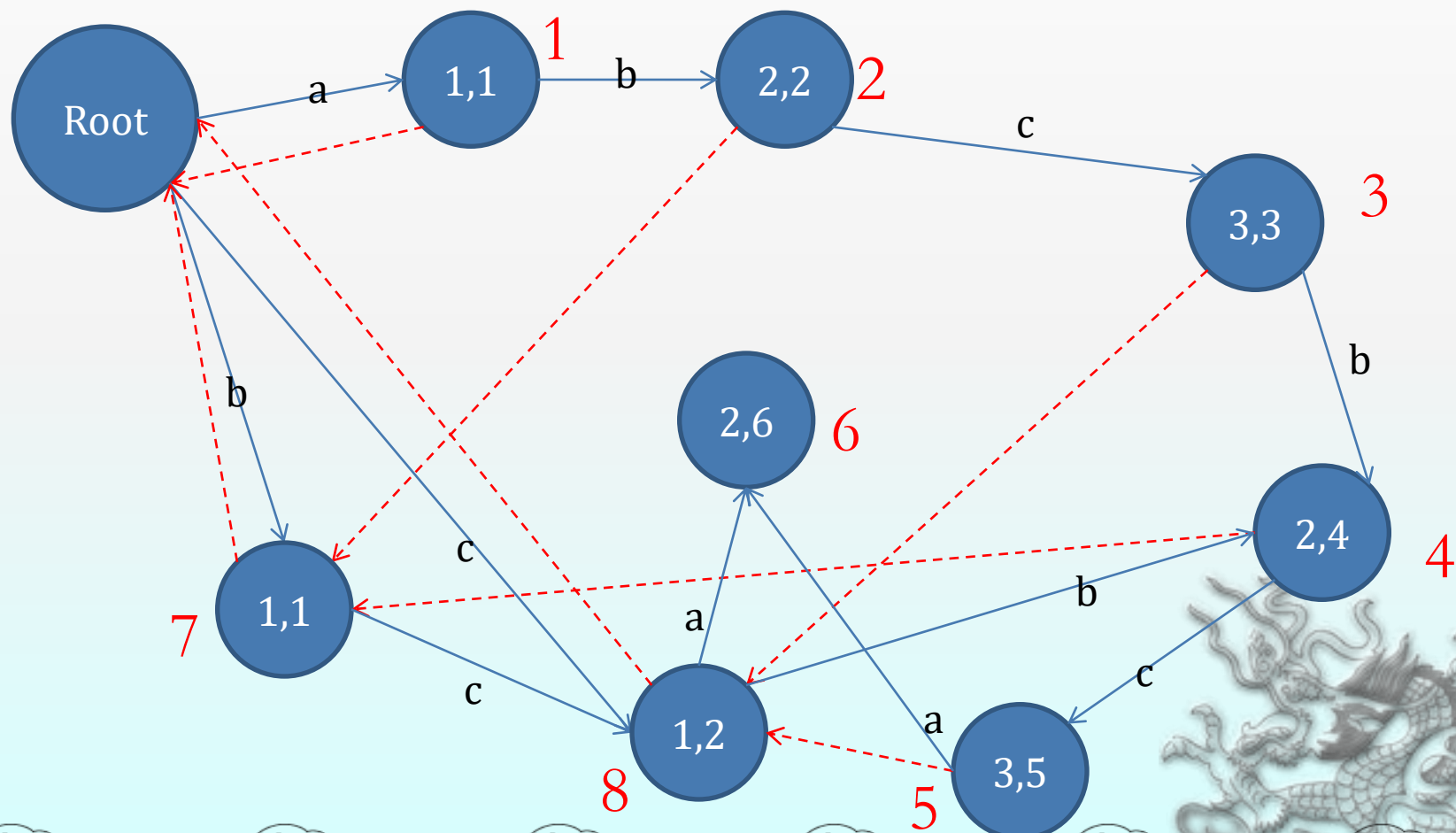


构造SAM

- ◆ 现在加入最后一个a。新建6号点。
- ◆ 沿着last从长到短加入后缀。
- ◆ 连a边。从5到6，从8到6。
- ◆ 现在到Root。发现Root已有一条a边，指向1。



构造SAM

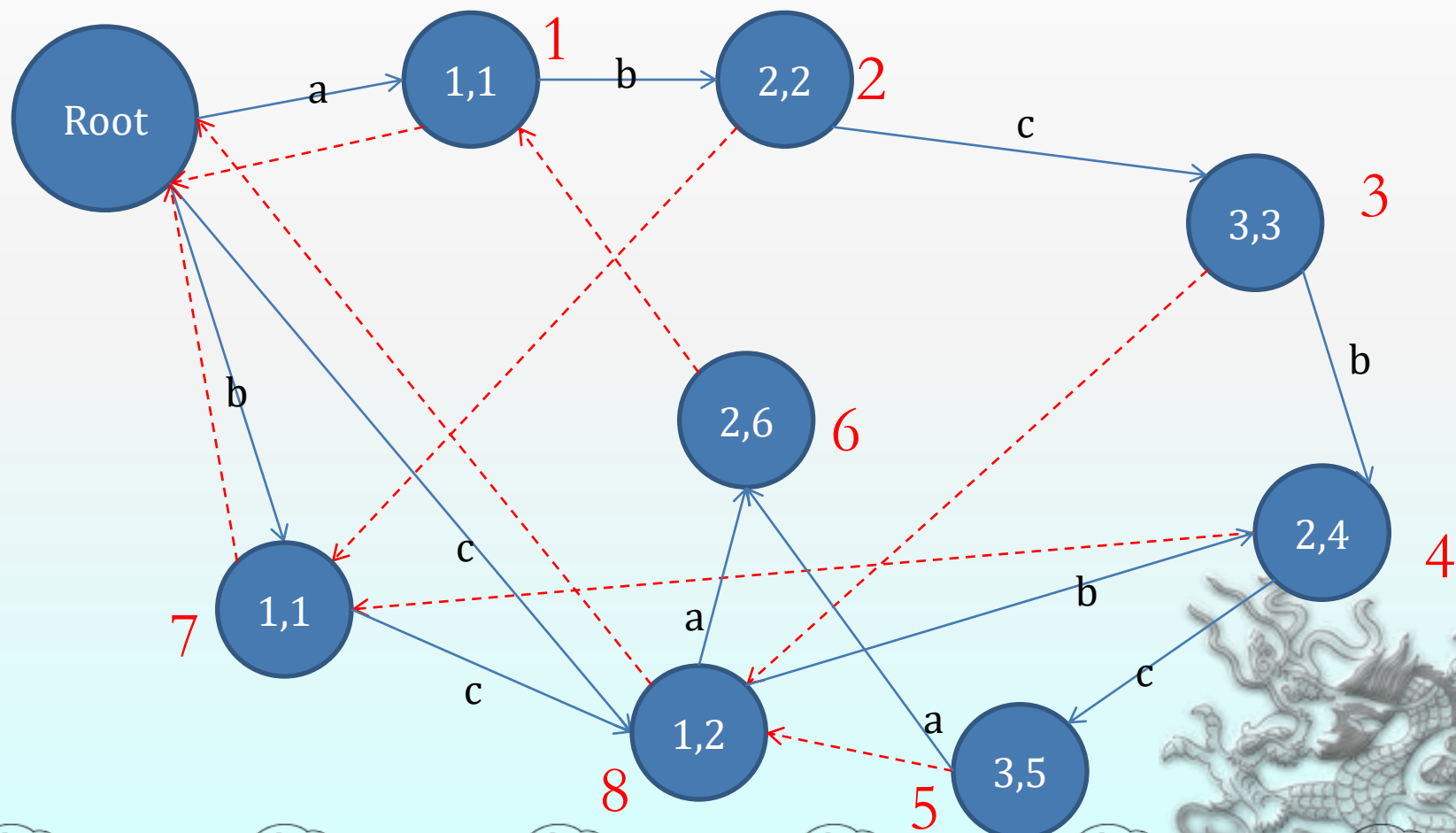


构造SAM

- ◆ 这时我们发现，这条a边指向的1号点对应的区间 $[1,1]$ 右端点正好是我们想要的1！
- ◆ 这时我们就可以不用拆分，可以很高兴地把6号点的last指针直接指向1号点了。
- ◆ 这样字符串 $S=\text{"abcbca"}$ 的SAM就完成了。
- ◆ 下页是完成后的SAM。
- ◆ 请务必再次检查min和max的正确性。



构造SAM



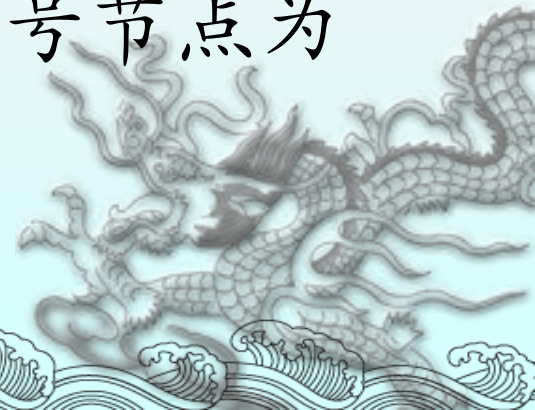
构造SAM

- ◆ 看过了一个具体的例子，下面就从理论方面介绍构造SAM的方法：



构造SAM

- ◆ 下面我们总结一下构造SAM的方法：
- ◆ 1. 从一个根节点Root开始，每次加一个字母，在前一个SAM的基础上构造下一个SAM。
- ◆ 2. 在加入第 i 个字母时，从第 $i-1$ 号节点开始，沿着last指针添加该字母的next边到 i 号点，直到已经加完Root的边或遇到一个已经存在该字母的next边的节点。（ $i-1$ 号节点为前一个SAM的主链的终点）



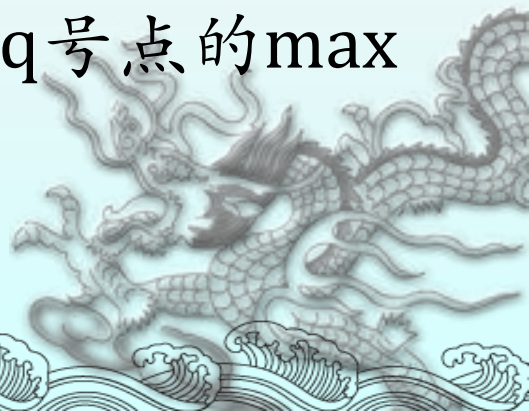
构造SAM

- ◆ 3.1.若已经加完Root的边，说明这是一个新的字母（否则Root就会有该字母的next边），此时只需要把i号点的last指向Root，让Root接受长度为0的后缀即可。
- ◆ 3.2若遇到了已连出边的节点，假设该节点编号为p，min值为m，max值为k，连出边指向的节点编号为q，那么现在的情况是这样：



构造SAM

- ◆ i. 已经加入了第 i 个前缀长度为 $(k+2) \sim i$ 的后缀。
- ◆ ii. 长度为 $(m+1) \sim (k+1)$ 的后缀均可由 q 号点接受得到。（根据last指针的性质，第 $(i-1)$ 个前缀的长度为 $m \sim k$ 的后缀的对应节点是 p ）
- ◆ iii. 长度为 $0 \sim m$ 的后缀均可由 q 号点的祖先接受得到。（因为根据定义， q 号点祖先接受的字符串一定是 q 号点接受的任一字符串的后缀）
- ◆ iv. 因为 p 号点向 q 号点有连边，所以 q 号点的 \max 至少为 $(k+1)$ 。



构造SAM

- ◆ 现在分为两种情况：
- ◆ 3.2.1.q号点的max等于 $(k+1)$ 。（这时候q号点对应的区间正好可以接上已加入的后缀长度区间 $[k+2, i]$ ，也就是q号点接受的字符串正好都是第i个前缀的后缀，）这时候只要把i号节点的last指向q号点即可。
- ◆ 3.2.2.q号点的max大于 $(k+1)$ 。我们只能让长度最多到 $(k+1)$ 的后缀作为i号点的父亲（即该父亲的max必须等于 $(k+1)$ ）。

构造SAM

- ◆ 所以我們必須拆開q號點對應的區間。
- ◆ 設q號點對應的區間為 $[Min, Max]$ 。
- ◆ 新建r號點（從 $(|S|+1)$ 或一個足夠大的數開始編號），使它對應 $[Min, k+1]$ 。
- ◆ 讓r號點擁有和q號點一樣的 $next[]$ 指針和 $last$ 指針（因為q號點的 $next[]$ 指向的節點的 $[Min+1, k+2]$ 現在要由r號點貢獻而不是q，r號點對應的區間的上一個區間是q號點的父親對應的區間）。

构造SAM


- ◆ 接着让q号点的last指针指向r号点（因为此时q号点对应的区间 $[k+2, \text{Max}]$ 的上一个区间是r号点对应的区间 $[\text{Min}, k+1]$ ）。
- ◆ 最后把p号点及其祖先指向q号点的该字符的next边全部指向r（由last指针性质得到，p号点这条next边对q号点区间的贡献是 $[m+1, k+1]$ ，祖先的next边对q号点区间的贡献是 $[\text{Min}, m]$ ，这样就把原来贡献给q号点的 $[\text{Min}, k+1]$ 区间全部转给了r）。

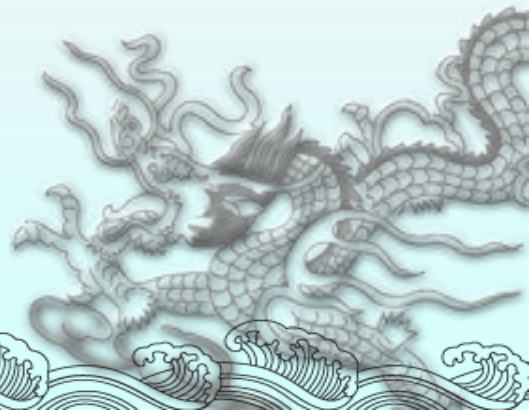
构造SAM

- ◆ 拆分完 q 之后，我们就可以把 i 的last指针指向我们拆出来的 r 号点了。



构造SAM


- ◆ 以下是缩略版：（跳过走这里）
- ◆ 1. 从一个根节点Root开始，每次加一个字母，在前一个SAM的基础上构造下一个SAM。
- ◆ 2. 在加入第 i 个字母时，从第 $i-1$ 号节点开始，沿着last指针添加该字母的next边到 i 号点，直到已经加完Root的边或遇到一个已经存在该字母的next边的节点。



构造SAM

- ◆ 3.1.若已经加完Root的边，把i号节点的last指向Root。
- ◆ 3.2.否则设已连出边为点p到点q。
 - ◆ 3.2.1.若q的max等于p的max+1，把i号点的last指向q。
 - ◆ 3.2.2.否则新建r号点，让r拥有和q一样的ne指针，把r的max设为p的max+1，接着让q的last指针指向r，最后把p及其祖先指向q的next边全部指向r。

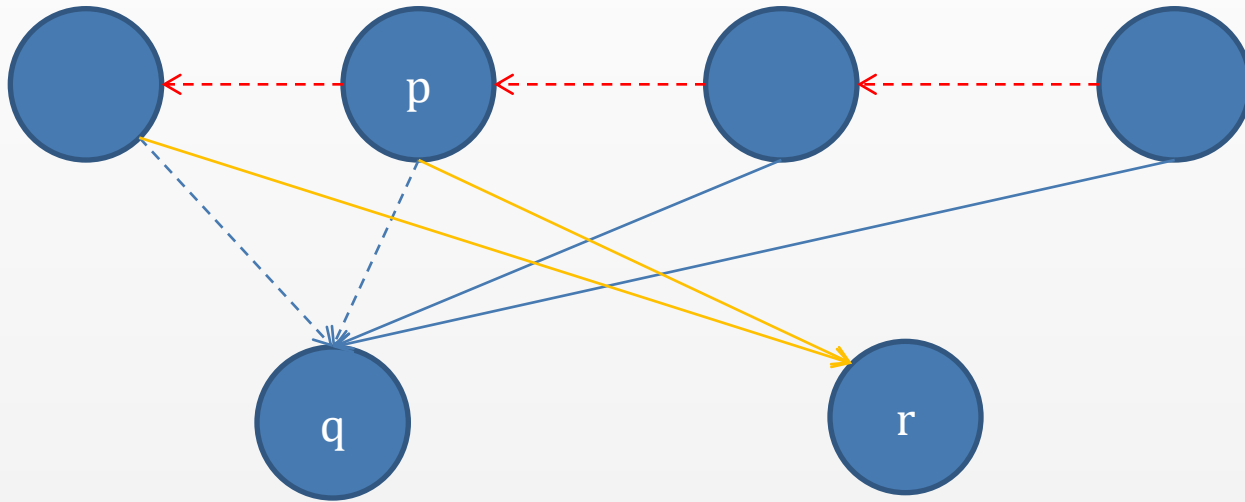
构造SAM

- ◇ 到这里，SAM的构造过程就都介绍完了。
- ◇ （若没有理解可以（再）看一遍实例）
- ◇ 我还要补充证明一点：一个点的所有入边(next边)来自last树上一条完整的路径，换句话说，对于每个非根节点 i ，都存在一条last树上的路径 (a,b) （ a 是 b 的祖先）使得这条路径上所有的点都有一条连向 i 的next边，且所有连向 i 的next边都来自这条路径上的点。

构造SAM

- ◆ 证明如下：
- ◆ 在2.的加边过程中，显然所有边都来自连续的一段last树上的路径（以下简称last路径）。
- ◆ 在3.2.2的拆点过程中， r 的所有next出边和 q 的相同，所以不会使任何点的last路径断裂。
- ◆ 在把指向 q 的边改成指向 r 时，由于是把 p 及其所有祖先指向 q 的边指向 r （如下页图）， q 的last路径断裂成两部分但依然分别完整。

构造SAM



(这样修改之后q和r依然满足存在last路径)

- ◆ 因为任何修改都不会破坏last路径，所以该命题成立。

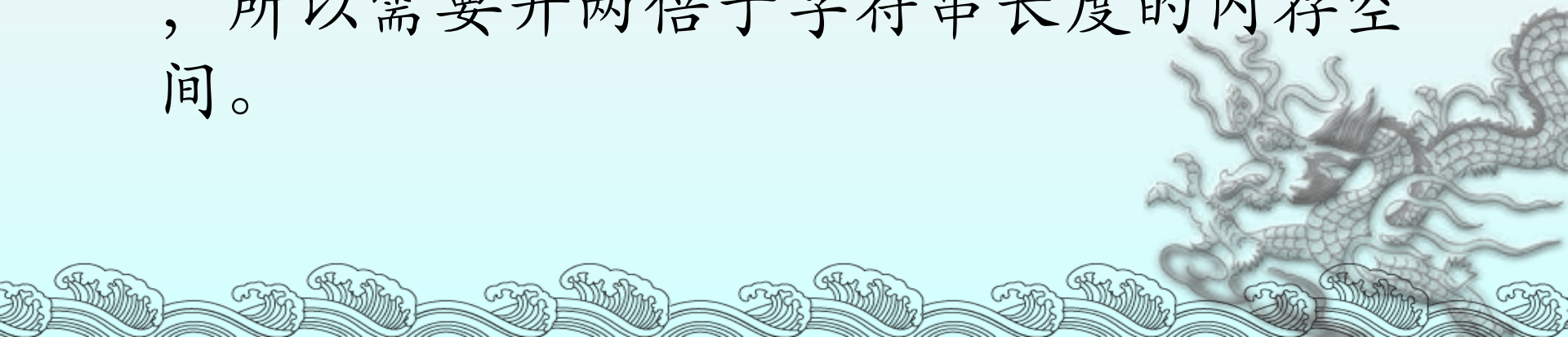
构造SAM

- ◆ 下面讲一下注意事项：
- ◆ 1.我们发现一个点的max确定之后就不再改变，而min的确定和改变和该点的父亲的设立和改变有关。而且一个节点的min一定等于其父亲的max+1。所以我们完全可以不用记min，只记录max。



构造SAM

- ◆ 2. 我们在把连向q的边改成连向r的边时，从p开始沿着last指针修改，由于last路径的性质，只要有一个祖先不存在连向q的边，那么其所有祖先也一定不存在连向q的边。所以此时即可停止修改。
- ◆ 3. 由于每加一个字母最多需要新建2个节点，所以需要开两倍于字符串长度的内存空间。



SAM的性质

- ◆ 构造好一个SAM以后，我们就可以好好地利用它了。但是SAM要怎么用呢？我现在来介绍一下SAM的一些基本性质。
- ◆ 首先我们得知道S的每一个子串 $S[l,r]$ 对应哪一个节点。
- ◆ $S[l,r]$ 是S第 r 个前缀长度为 $(r-l+1)$ 的后缀。由SAM的定义可得 $S[l,r]$ 对应的节点为第 r 个节点及其祖先中，
唯一满足 $\min \leq r-l+1 \leq \max$ 的节点。

SAM的性质

- ◆ 然后我们要知道每个节点接受了哪些子串。
- ◆ 易证在构造过程中Root、 $1 \sim |S|$ 号节点构成了主链。
- ◆ i 号点若在主链上（即 i 为Root或 $1 \leq i \leq |S|$ ），那么根据min,max的定义，该点接受了子串 $S[i-\min+1,i], S[i-\min,i], S[i-\min-1,i] \dots S[1,i]$ 。
- ◆ 一个点还可能作为别的主链上的点的祖先。所以还要算上这些前缀中长度在 $[\min, \max]$ 之间的后缀。

SAM的性质

- ◆ 所以一个节点接受的S的子串在S上画出来大概是这样的：（X代表这里有一个字符）
- ◆ S=“XABCAXXXABCABCAXX”



SAM的性质

- ◆ SAM可以不重不漏地接受字符串S的每一个子串。因为每个点每种字母的next边都只有一条，所以一定不会重复。因为构造时保证对于第 i 个前缀仅加入了 $(i+1)$ 个后缀，所以不会遗漏，也不会多余。



SAM的性质

- ◆ 知道了这些，我们就可以进行一些统计工作。
- ◆ 首先我们可以知道一个串S有多少个本质不同的子串。
- ◆ 因为任意两个节点不可能接受一个相同的字符串（否则不满足之前证的不重复），所以我们只要把每个点接受的本质不同的字符串个数加起来即可。而对于一个点，这个数字为 $(\max - \min + 1)$ 。

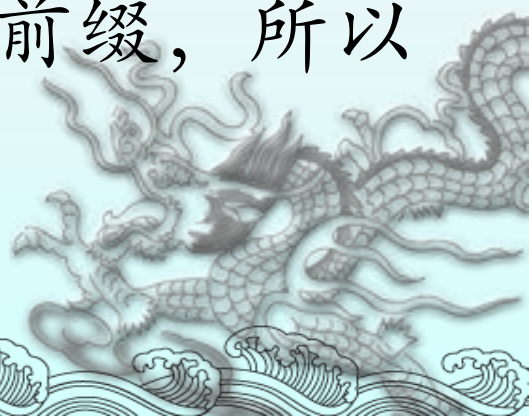
SAM的性质

- ◆ 然后我们还可以计算给定的字符串出现的次数。
- ◆ 如果SAM不能接受该串，那么显然出现次数为0。
- ◆ 否则该串一定对应了SAM的某个节点。
- ◆ 因为一个节点接受的串为last子树中主链上的点（代表一些前缀）的长度在某个区间内的后缀，所以该串在这些前缀中作为后缀出现，所以该串出现的次数为last子树中主链上的点的个数。



SAM的性质

- ◆ SAM还有一个不是很有用的性质，就是如果把编号大于 $|S|$ 的节点和它的父亲合并的话，那么last树变成了KMP数组！也就是说，一个主链上的点的最近的主链上的祖先是它的border。
- ◆ KMP数组记录了一个串所有前缀的border。
- ◆ 根据定义一个串的border是它的前缀，所以它必然对应主链上一个点。



SAM的性质

- ◆ 根据定义一个串的border也是它的后缀，所以第 i 个前缀的border必然对应着 i 号点的一个祖先。
- ◆ 如果 i 号点的一个祖先不在主链上，那么它必然不能接受第 i 个前缀的任一个前缀（因为第 i 个前缀的任意一个前缀一定对应主链上的一个点，而一个字符串最多对应一个节点），所以 i 号点的最近的在主链上的祖先接受了第 i 个前缀的border。

SAM入门——从根开始

- ◆ 介绍了这些，相信你已经基本掌握了SAM的相关知识。
- ◆ SAM入门——从根开始。
- ◆ 感谢你的阅读。

