Manacher, PAM

calabash_boy

2022 年 4 月 28 日

回文串

定义

反转串 R(S)：一个字符串 $S = S[1]S[2]\cdots S[n]$，其反串为 $R(S) = S[n]S[n-1]\cdots S[1]$。

回文串：满足 $S = R(S)$ 的串为回文串。

回文串

定义

反转串 R(S)：一个字符串 $S = S[1]S[2]\cdots S[n]$，其反串为 $R(S) = S[n]S[n-1]\cdots S[1]$。

回文串：满足 $S = R(S)$ 的串为回文串。

回文中心：

1. 奇（长度）回文串，回文中心为 $S[\frac{n+1}{2}]$，如 *abcba*
2. 偶（长度）回文串，回文中心为 $S[\frac{n}{2}]$ 与 $S[\frac{n}{2}+1]$ 中间，如 *abc|cba*。

回文串

定义

反转串 R(S)：一个字符串 $S = S[1]S[2]\cdots S[n]$，其反串为 $R(S) = S[n]S[n-1]\cdots S[1]$。

回文串：满足 $S = R(S)$ 的串为回文串。

回文中心：

1. 奇（长度）回文串，回文中心为 $S[\frac{n+1}{2}]$，如 *abcba*
2. 偶（长度）回文串，回文中心为 $S[\frac{n}{2}]$ 与 $S[\frac{n}{2}+1]$ 中间，如 *abc|cba*。

回文半径 L：回文中心到回文串的左右端点的距离相等，此距离称为回文半径。

如 *abcba* 半径为 3，*abcd|dcba* 半径为 4。

回文串

定义

反转串 R(S)：一个字符串 $S = S[1]S[2]\cdots S[n]$，其反串为 $R(S) = S[n]S[n-1]\cdots S[1]$。

回文串：满足 $S = R(S)$ 的串为回文串。

回文中心：

1. 奇（长度）回文串，回文中心为 $S[\frac{n+1}{2}]$，如 *abcba*
2. 偶（长度）回文串，回文中心为 $S[\frac{n}{2}]$ 与 $S[\frac{n}{2}+1]$ 中间，如 *abc|cba*。

回文半径 L：回文中心到回文串的左右端点的距离相等，此距离称为回文半径。

如 *abcba* 半径为 3，*abcd|dcba* 半径为 4。

常用二元组 $<$ **回文中心，回文半径** $>$ 来表示一个回文子串

回文串

性质

长度与半径的关系：

1. 奇回文串：$|S| = 2L - 1$
2. 偶回文串：$|S| = 2L$

回文串

性质

长度与半径的关系：

1. 奇回文串：$|S| = 2L - 1$
2. 偶回文串：$|S| = 2L$

回文半径的二分性：回文半径-1 等价于同时删掉回文串的首尾字母，依然是回文串。

回文串

性质

长度与半径的关系：

1. 奇回文串：$|S| = 2L - 1$
2. 偶回文串：$|S| = 2L$

回文半径的二分性：回文半径-1 等价于同时删掉回文串的首尾字母，依然是回文串。

回文串和 Border：对于回文串 S，回文前（后）缀等价于Border

求回文半径

问题背景

给出一个字符串 S，求每个回文中心的回文半径。包括一个字母作为中心以及两个字母中间的位置作为中心。

求回文半径

问题背景

给出一个字符串 S，求每个回文中心的回文半径。包括一个字母作为中心以及两个字母中间的位置作为中心。

不动脑子的做法

利用回文半径的二分性质，预处理 S 和 $R(S)$ 的 Hash，然后利用二分 +Hash 求每个中心的回文半径。复杂度 $O(n \log n)$

Manacher

前期处理

为了将偶回文串的处理方式与奇回文串统一起来，将 S 的每两个字母中间，以及开头结尾插入 $\#$。
例如 $S = bccbeb$，预处理后变为 $S^{\#} = \#b\#c\#c\#b\#e\#b\#$。

Manacher

前期处理

为了将偶回文串的处理方式与奇回文串统一起来，将 S 的每两个字母中间，以及开头结尾插入 $\#$。
例如 $S = bccbeb$，预处理后变为 $S^{\#} = \#b\#c\#c\#b\#e\#b\#$。
因此所有回文串都变成奇数长度，且首尾一定是 $\#$。例如

1. 原始偶回文串：$bccb^{\#} = \#b\#c\#c\#b\#$，长度为 9，回文中心是 $\#$
2. 原始奇回文串：$beb^{\#} = \#b\#e\#b\#$，长度为 7，回文中心是 e

Manacher

前期处理

为了将偶回文串的处理方式与奇回文串统一起来，将 S 的每两个字母中间，以及开头结尾插入 $\#$。
例如 $S = bccbeb$，预处理后变为 $S^{\#} = \#b\#c\#c\#b\#e\#b\#$。
因此所有回文串都变成奇数长度，且首尾一定是 $\#$。例如

1. 原始偶回文串：$bccb^{\#} = \#b\#c\#c\#b\#$，长度为 9，回文中心是 $\#$
2. 原始奇回文串：$beb^{\#} = \#b\#e\#b\#$，长度为 7，回文中心是 e

同时，所有极长回文子串长度一定为奇数：因为极长回文子串一定以 $\#$ 开头结尾。

Manacher

前期处理

为了将偶回文串的处理方式与奇回文串统一起来，将 S 的每两个字母中间，以及开头结尾插入 $\#$。

例如 $S = bccbeb$，预处理后变为 $S^{\#} = \#b\#c\#c\#b\#e\#b\#$。

因此所有回文串都变成奇数长度，且首尾一定是 $\#$。例如

1. 原始偶回文串：$bccb^{\#} = \#b\#c\#c\#b\#$，长度为 9，回文中心是 $\#$
2. 原始奇回文串：$beb^{\#} = \#b\#e\#b\#$，长度为 7，回文中心是 e

同时，所有极长回文子串长度一定为奇数：因为极长回文子串一定以 $\#$ 开头结尾。

容易发现：$|S^{\#}| = 2|S| + 1$，以及 $|S| = \frac{|S^{\#}|-1}{2} = \lfloor \frac{|S^{\#}|}{2} \rfloor$。容易验证此关系对回文半径依然适用。

Manacher

定义

$Len[i]$ 表示以 i 为回文中心的最大回文半径。
最右回文串 P：所有已求得的回文串中，右端点最靠右的一个。

Manacher

定义

$Len[i]$ 表示以 i 为回文中心的最大回文半径。
最右回文串 P：所有已求得的回文串中，右端点最靠右的一个。

算法流程

从左到右求每个位置的回文半径，同时维护最右回文串 $S[L, R]$ 及其回文中心 p。

Manacher

定义

$Len[i]$ 表示以 i 为回文中心的最大回文半径。
最右回文串 P：所有已求得的回文串中，右端点最靠右的一个。

算法流程

从左到右求每个位置的回文半径，同时维护最右回文串 $S[L,R]$ 及其回文中心 p。
设当前 $1,2,\cdots,i-1$ 位置的 Len 已经求出，当前需要求 $Len[i]$，根据 i 与 $[L,R]$ 的关系，总共分三类情况讨论：

Manacher

算法流程

1、$i > R$:

以 i 为回文中心，向左向右暴力拓展，求得回文半径 $Len[i]$，同时最右回文串会变为:

$p = i$

$L = i - Len[i] + 1$

$R = i + Len[i] - 1$

Manacher

算法流程

1、$i > R$:

以 i 为回文中心，向左向右暴力拓展，求得回文半径 $Len[i]$，同时最右回文串会变为:

$p = i$

$L = i - Len[i] + 1$

$R = i + Len[i] - 1$

Manacher

算法流程

2、$i \leq R$:

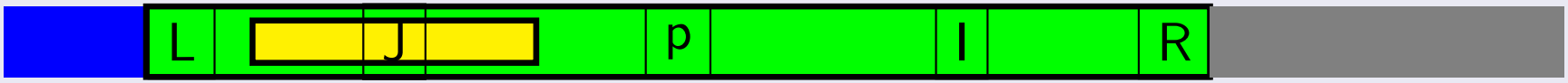
由于回文串的对称性：最右回文串 P 的左半和右半是对称的。找到 i 关于 p 的对称位置 $j = 2p - i$。

由于 $Len[j]$ 是已经求得的，根据最右回文串的对称性，$Len[i]$ 可以直接继承 $Len[j]$ 在最右回文串范围内的部分，根据 $Len[j]$ 是否超出了最右回文串的范围，继续讨论两种情况。

Manacher

算法流程

2.1、$j - Len[j] + 1 > L$

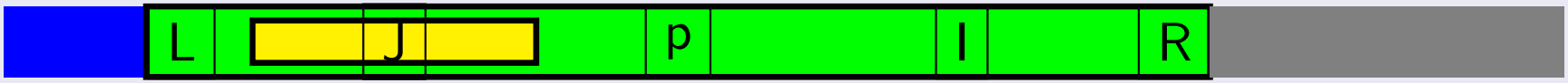


由于 $Len[j]$ 没有超出最右回文串的表示范围，由对称性，可以确定 $Len[i] = Len[j]$，且不能够再拓展。

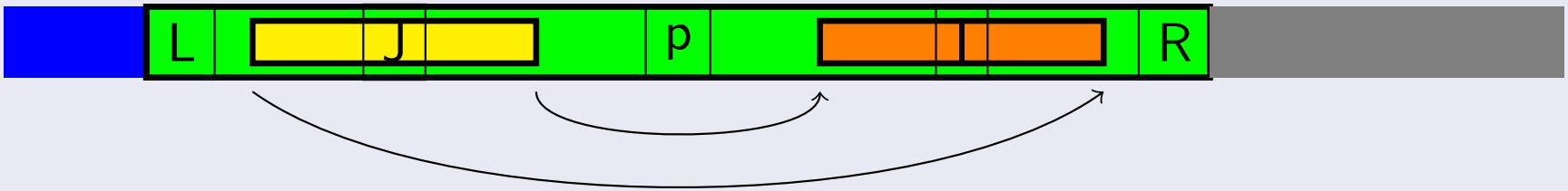
Manacher

算法流程

2.1、$j - Len[j] + 1 > L$



由于 $Len[j]$ 没有超出最右回文串的表示范围，由对称性，可以确定 $Len[i] = Len[j]$，且不能够再拓展。



此时，最右回文串没有发生变化。

Manacher

算法流程

2.2、$j - Len[j] + 1 \leq L$

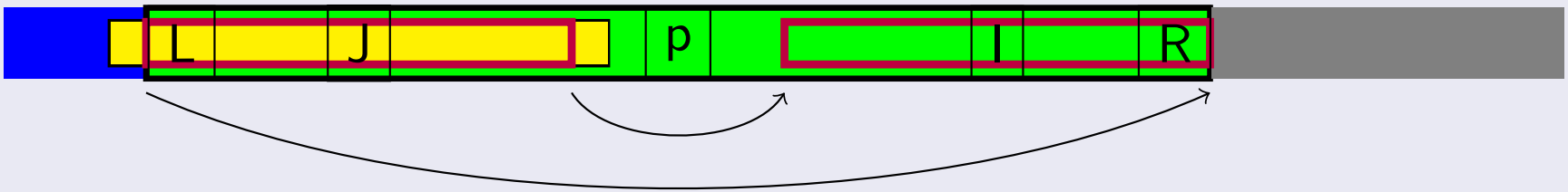
由于 $Len[j]$ 超出了最右回文串的范围，且灰色部分的值未知，因此 $Len[i]$ 至多只能继承到最右回文串范围内的 $Len[j]$，即 $Len[i] < -j - L + 1$

Manacher

算法流程

2.2、$j - Len[j] + 1 \leq L$

由于 $Len[j]$ 超出了最右回文串的范围，且灰色部分的值未知，因此 $Len[i]$ 至多只能继承到最右回文串范围内的 $Len[j]$，即 $Len[i] < -j - L + 1$



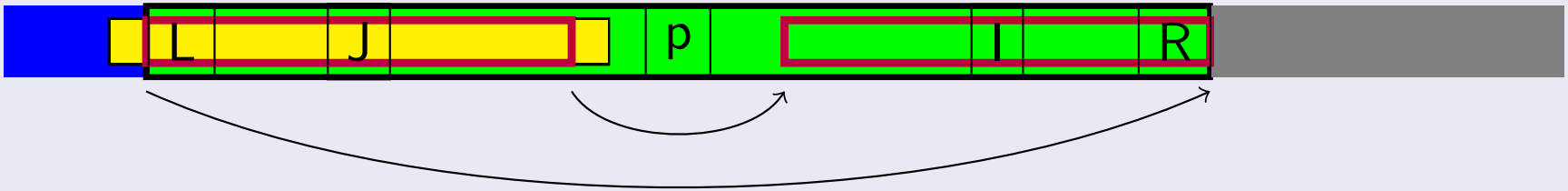
之后由于灰色部分的值未知，因此需要继续向左向右暴力拓展。

Manacher

算法流程

2.2、$j - Len[j] + 1 \leq L$

由于 $Len[j]$ 超出了最右回文串的范围，且灰色部分的值未知，因此 $Len[i]$ 至多只能继承到最右回文串范围内的 $Len[j]$，即 $Len[i] < -j - L + 1$



之后由于灰色部分的值未知，因此需要继续向左向右暴力拓展。

最右回文串会更新为 i。

Manacher

复杂度分析

每次暴力匹配一定伴随着最右回文串右端点 R 的右移。因此复杂度为线性。

Manacher

复杂度分析

每次暴力匹配一定伴随着最右回文串右端点 R 的右移。因此复杂度为线性。

用法

1. 求每个回文中心的回文半径
2. 求本质不同回文串：在 Manacher 中，新的回文串一定出现在使得最右串右移的时候。因此本质不同回文串至多 n 个，把所有更新最右回文串去重即得到本质不同回文串。

例题 1

例题 1

给出一个字符串 S，和 Q 次询问，每次询问 $S[L, R]$ 有多少个回文子串。

例题 1

例题 1

给出一个字符串 S，和 Q 次询问，每次询问 $S[L, R]$ 有多少个回文子串。

题解

从询问入手，由于回文串天生的对称性，因此可以把问题分成左右两半来思考：

例题 1

例题 1

给出一个字符串 S，和 Q 次询问，每次询问 $S[L, R]$ 有多少个回文子串。

题解

从询问入手，由于回文串天生的对称性，因此可以把问题分成左右两半来思考：
如果回文串的中心在询问区间的左半边，那么左端点将称为回文半径长度的唯一限制，中心在右半边同理。

例题 1

例题 1

给出一个字符串 S，和 Q 次询问，每次询问 $S[L, R]$ 有多少个回文子串。

题解

从询问入手，由于回文串天生的对称性，因此可以把问题分成左右两半来思考：
如果回文串的中心在询问区间的左半边，那么左端点将称为回文半径长度的唯一限制，中心在右半边同理。
利用 Manacher 预处理回文半径，之后问题将变为二维数点。

例题 2

某谷 P1659

给出一个字符串 S，$|S| \leq 1000,000$。求前 K 大的回文子串长度乘积。$K \leq 1000,000,000$。

例题 2

某谷 P1659

给出一个字符串 S，$|S| \leq 1000,000$。求前 K 大的回文子串长度乘积。
$K \leq 1000,000,000$。

题解

首先用 Manacher 处理每个位置的回文半径，于是每个位置代表了一系列的回文串:
长度分别为 $X, X-2, X-4, \cdots, 0/1$。
从大到小扫描，边合并边计算乘积即可。

例题 3

某谷 P4555

给出一个字符串 S，求最长的双回文子串。
回文双子串 T 定义为：可以从一个位置切开，使得前缀和后缀都是回文串。
$|S| \leq 100,000.$

例题 3

某谷 P4555

给出一个字符串 S，求最长的双回文子串。
回文双子串 T 定义为：可以从一个位置切开，使得前缀和后缀都是回文串。
$|S| \leq 100,000.$

题解

可以枚举拼接点，然后分别最大化以该点为左端点和右端点的回文串长度。
即等价于最大化左侧回文串和右侧回文串的回文半径。

例题 3

某谷 P4555

给出一个字符串 S，求最长的双回文子串。
回文双子串 T 定义为：可以从一个位置切开，使得前缀和后缀都是回文串。
$|S| \leq 100,000.$

题解

可以枚举拼接点，然后分别最大化以该点为左端点和右端点的回文串长度。
即等价于最大化左侧回文串和右侧回文串的回文半径。
利用 Manacher 求出回文半径之后，配合线段树等数据结构进行区间更新和查询即可。

例题 3

某谷 P3501

给出一个 01 串 S，求最长的反对称子串。
反对称串 T 定义为：将 $R(T)$ 逐位取反之后等于原串 T，则 T 是一个反对称串，例如 010101。
$|S| \leq 500,000$

例题 3

某谷 P3501

给出一个 01 串 S，求最长的反对称子串。
反对称串 T 定义为：将 $R(T)$ 逐位取反之后等于原串 T，则 T 是一个反对称串，例如 010101。
$|S| \leq 500,000$

题解

在新的相等运算意义下进行 Manacher 即可。

例题 4

某谷 P4287

给出一个字符串 S，求最长的双倍回文子串。若一个串能表示成 $TR(T)TR(T)$ 的形式，则称为一个双倍回文串，例如 $abbaabba$。$|S| \leq 500,000$.

例题 4

某谷 P4287

给出一个字符串 S，求最长的双倍回文子串。若一个串能表示成 $TR(T)TR(T)$ 的形式，则称为一个双倍回文串，例如 *abbaabba*。$|S| \leq 500,000$.

题解

由于本质不同回文串只有 n 个，因此逐一检查是否是双倍回文即可。

例题 4

某谷 P4287

给出一个字符串 S，求最长的双倍回文子串。若一个串能表示成 $TR(T)TR(T)$ 的形式，则称为一个双倍回文串，例如 $abbaabba$。$|S| \leq 500,000$.

题解

由于本质不同回文串只有 n 个，因此逐一检查是否是双倍回文即可。由于新的回文串一定发生于更新最右回文串的时候，所以在发生暴力拓展的时候，顺便检查即可。

PAM

定义

Palindrome Automaton（回文自动机，回文树）是一种能够识别所有回文子串的数据结构，结构十分类似于之前讲的 ACAM。

PAM

定义

Palindrome Automaton（回文自动机，回文树）是一种能够识别所有回文子串的数据结构，结构十分类似于之前讲的 ACAM。

1. 节点：节点数至多 N 个，每个节点代表了一种回文串。用 $S(u)$ 表示节点 u 代表的回文串。$len[u] = |S(u)|$

PAM

定义

Palindrome Automaton（回文自动机，回文树）是一种能够识别所有回文子串的数据结构，结构十分类似于之前讲的 ACAM。

1. 节点：节点数至多 N 个，每个节点代表了一种回文串。用 $S(u)$ 表示节点 u 代表的回文串。$len[u] = |S(u)|$
2. 后继边：每个后继边上有一个字母。用 $trans(u, ch) = v$ 表示 u 节点有后继边 ch 指向 v 节点。则有 $S(v) = \mathbf{ch}S(u)\mathbf{ch}$，以及 $len[v] = len[u] + 2$

PAM

定义

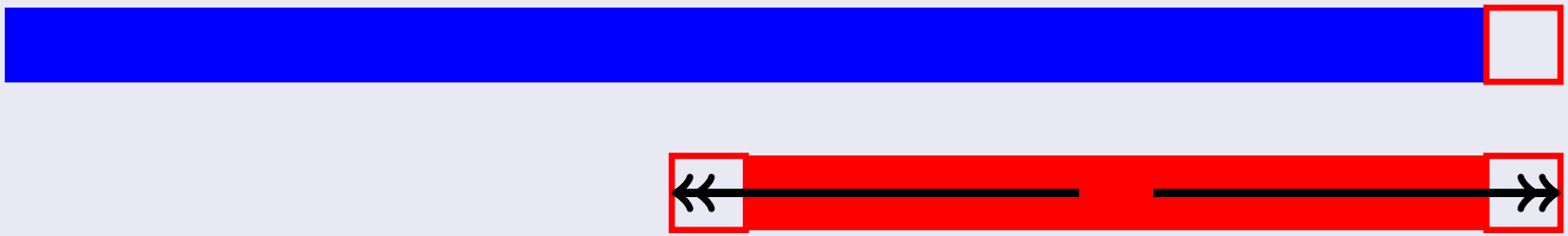
Palindrome Automaton（回文自动机，回文树）是一种能够识别所有回文子串的数据结构，结构十分类似于之前讲的 ACAM。

1. 节点：节点数至多 N 个，每个节点代表了一种回文串。用 $S(u)$ 表示节点 u 代表的回文串。$len[u] = |S(u)|$
2. 后继边：每个后继边上有一个字母。用 $trans(u, ch) = v$ 表示 u 节点有后继边 ch 指向 v 节点。则有 $S(v) = \mathbf{ch}S(u)\mathbf{ch}$，以及 $len[v] = len[u] + 2$
3. 失配边：每个节点都有一个失配边，用 $fail[u] = v$ 表示 u 节点的失配边指向了 v 节点。则有 $S(v)$ 是 $S(u)$ 的最大 Border，即最长回文后缀。

PAM

PAM 的构造

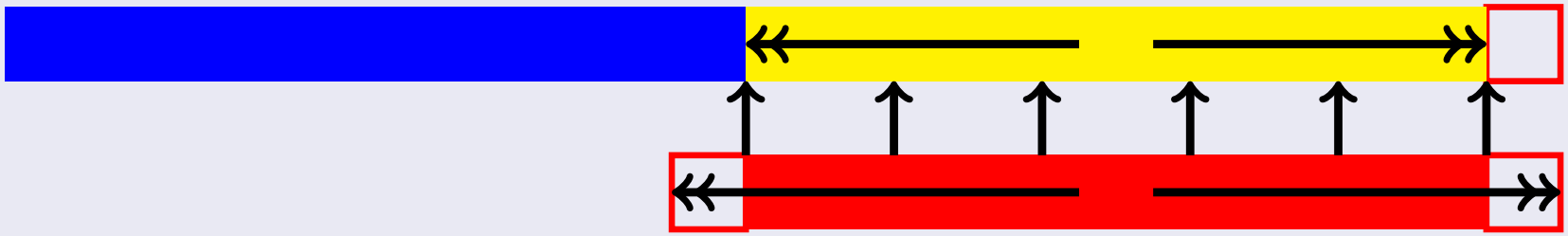
PAM 在构造时，实际上就是求每个前缀的最长回文后缀，方法是枚举前一个位置的回文后缀，即 fail 链。



PAM

PAM 的构造

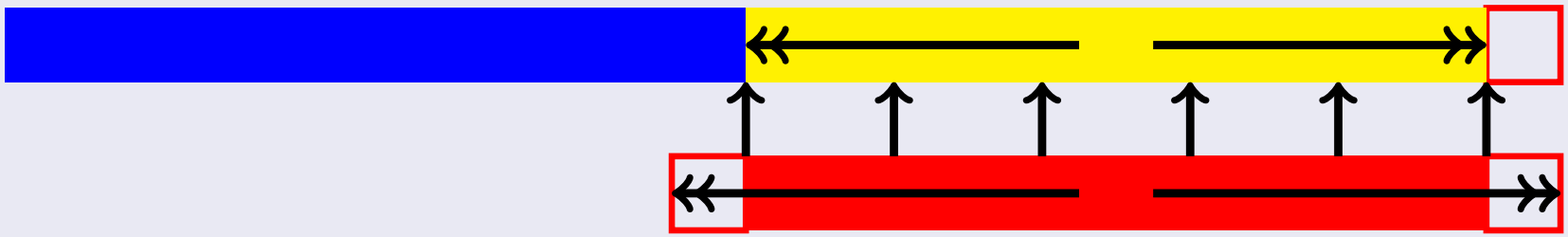
PAM 在构造时，实际上就是求每个前缀的最长回文后缀，方法是枚举前一个位置的回文后缀，即 fail 链。



PAM

PAM 的构造

PAM 在构造时，实际上就是求每个前缀的最长回文后缀，方法是枚举前一个位置的回文后缀，即 fail 链。

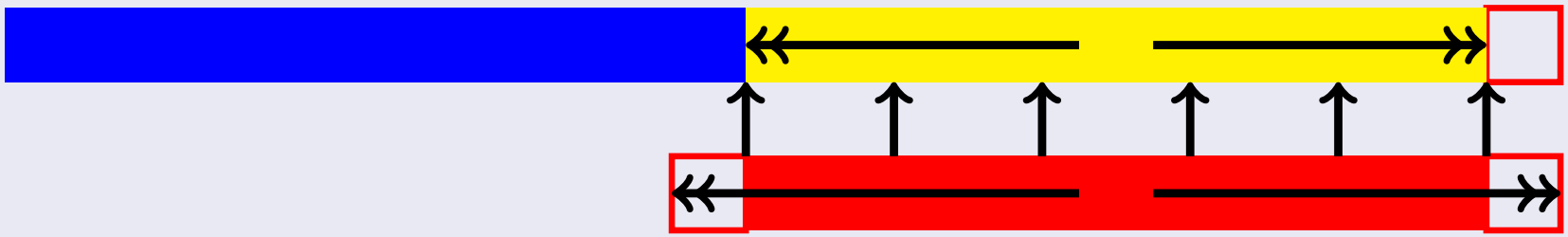


PAM 特殊之处在于：对于奇回文串和偶回文串需要有两个根节点。
由于长度为 2 的回文串是偶根的后继节点，长度为 1 的回文串是奇根的后继节点。
因此偶根长度应设为 0，奇根应该设为-1。

PAM

PAM 的构造

PAM 在构造时，实际上就是求每个前缀的最长回文后缀，方法是枚举前一个位置的回文后缀，即 fail 链。

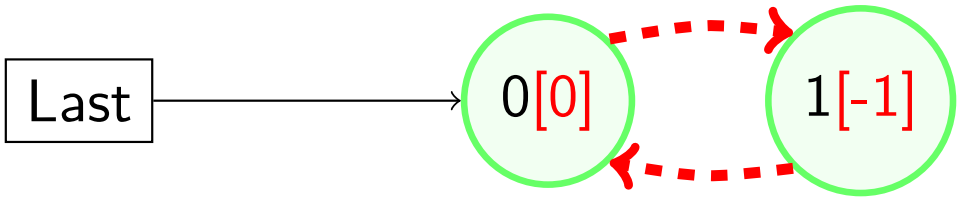


PAM 特殊之处在于：对于奇回文串和偶回文串需要有两个根节点。
由于长度为 2 的回文串是偶根的后继节点，长度为 1 的回文串是奇根的后继节点。
因此偶根长度应设为 0，奇根应该设为-1。
方便起见，可以令偶根的失配边指向奇根，奇根的失配边指向偶根。

PAM

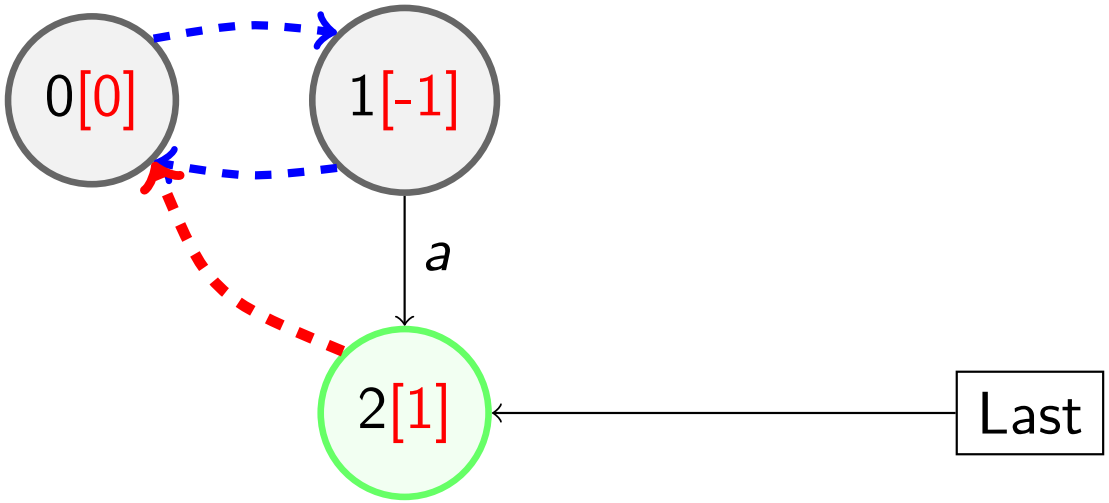
$S = abacabbaca$.

1 表示奇根，0 表示偶根。方括号表示长度。Last 指针指向偶根 0.



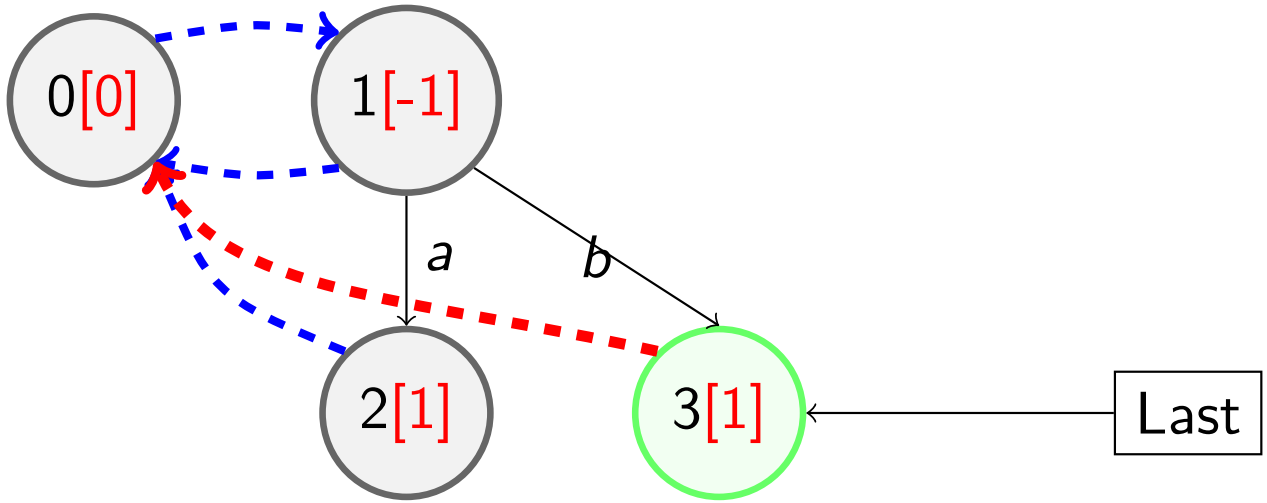
PAM

$S = abacabbaca$.



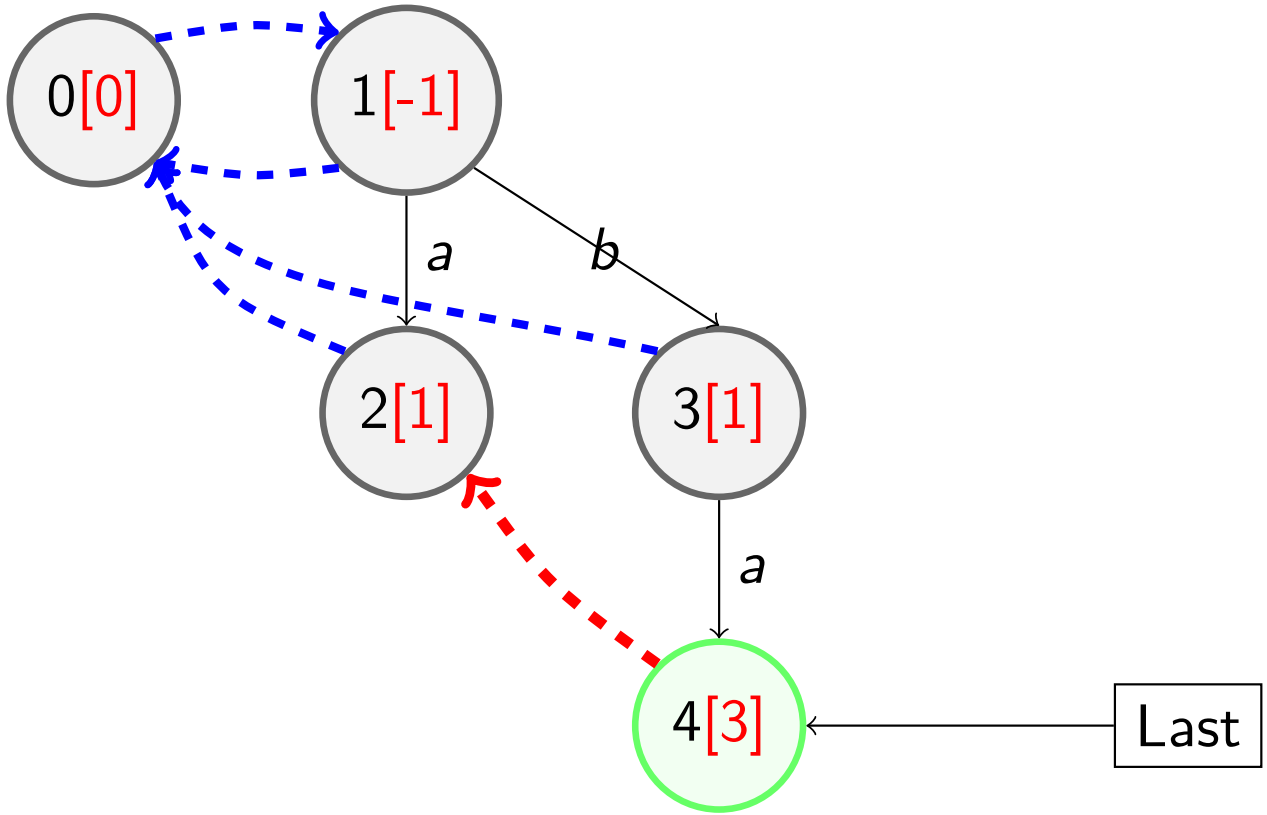
PAM

$S = abacabbaca.$



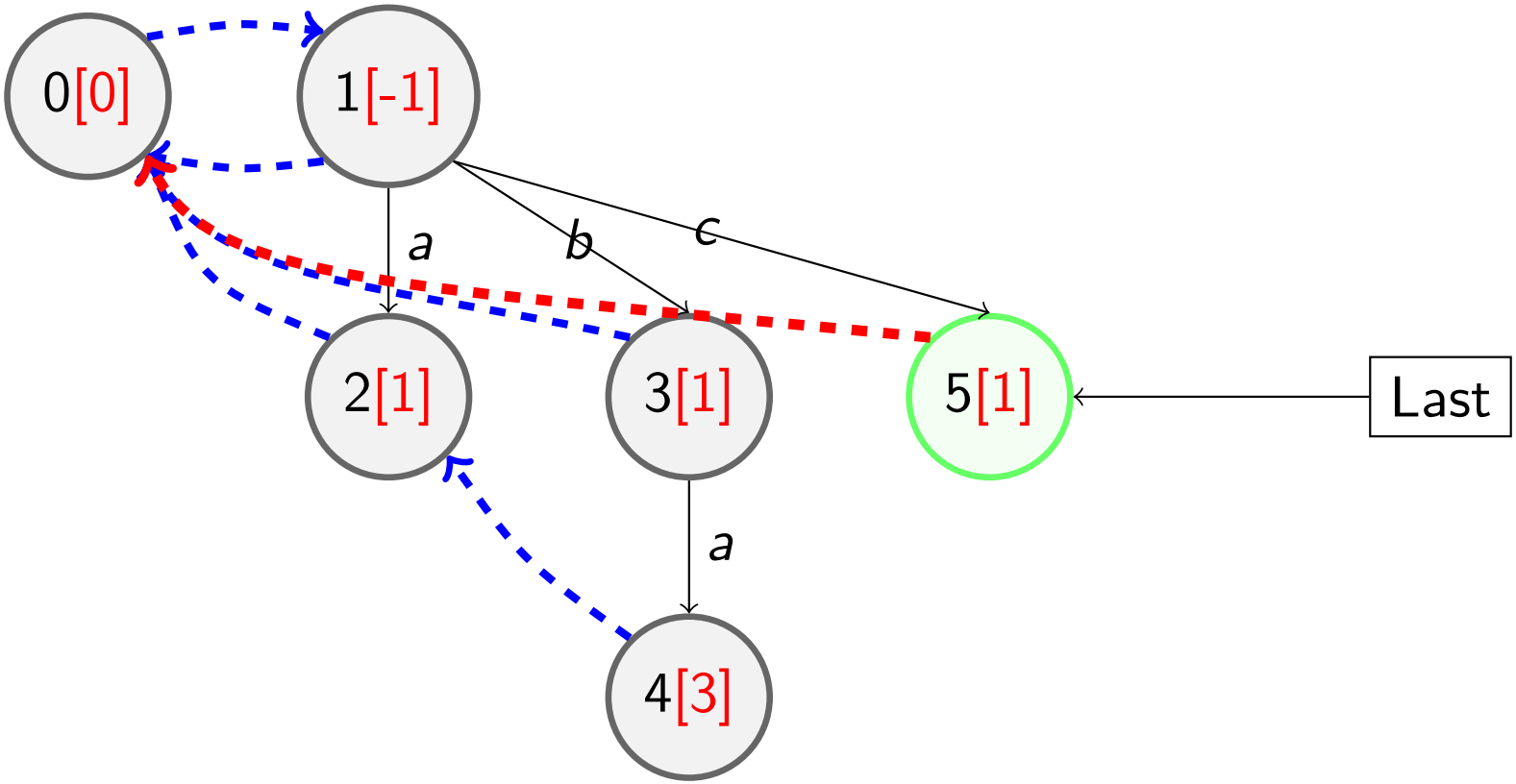
PAM

$S = abacabbaca$.



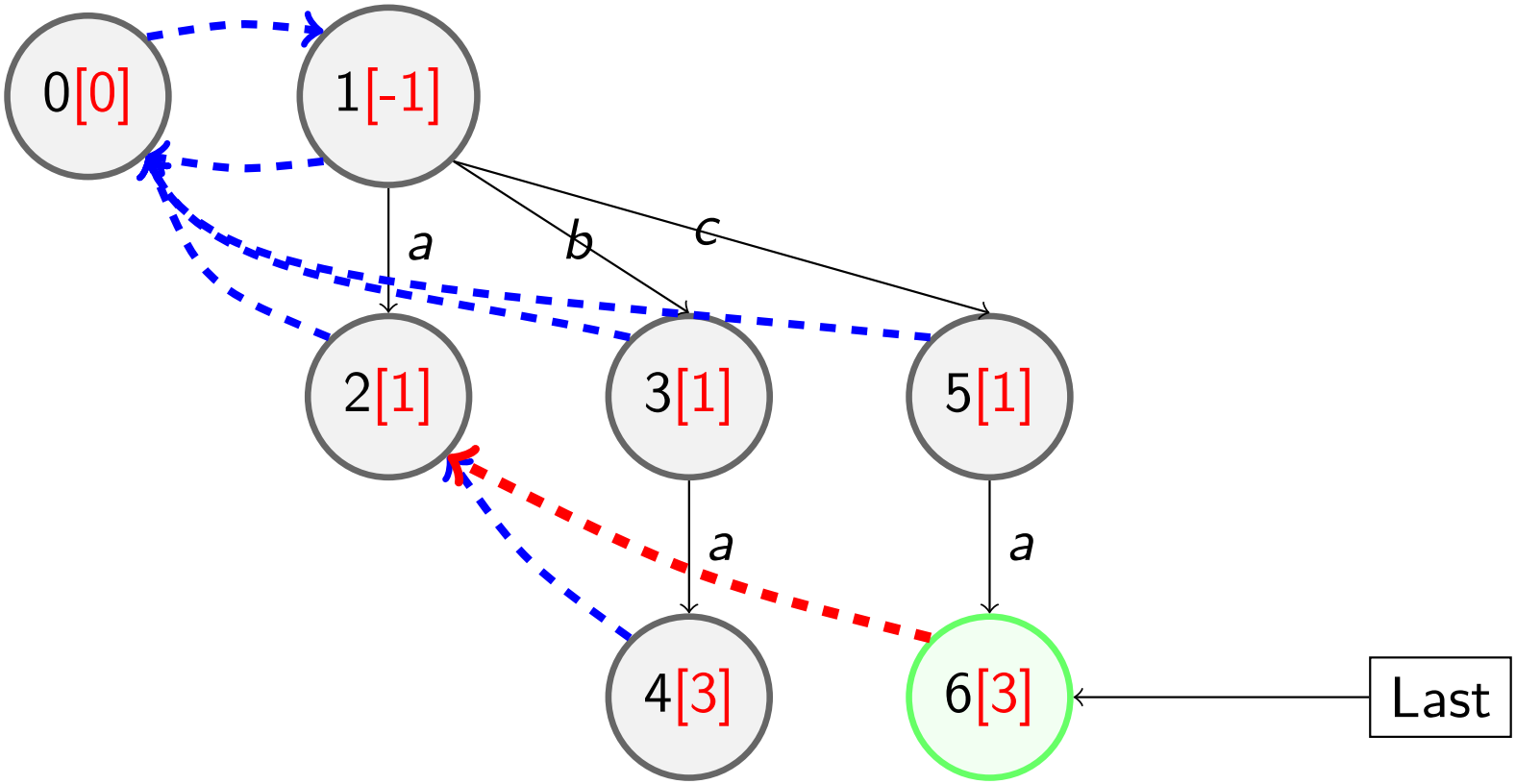
PAM

$S = abacabbaca$.



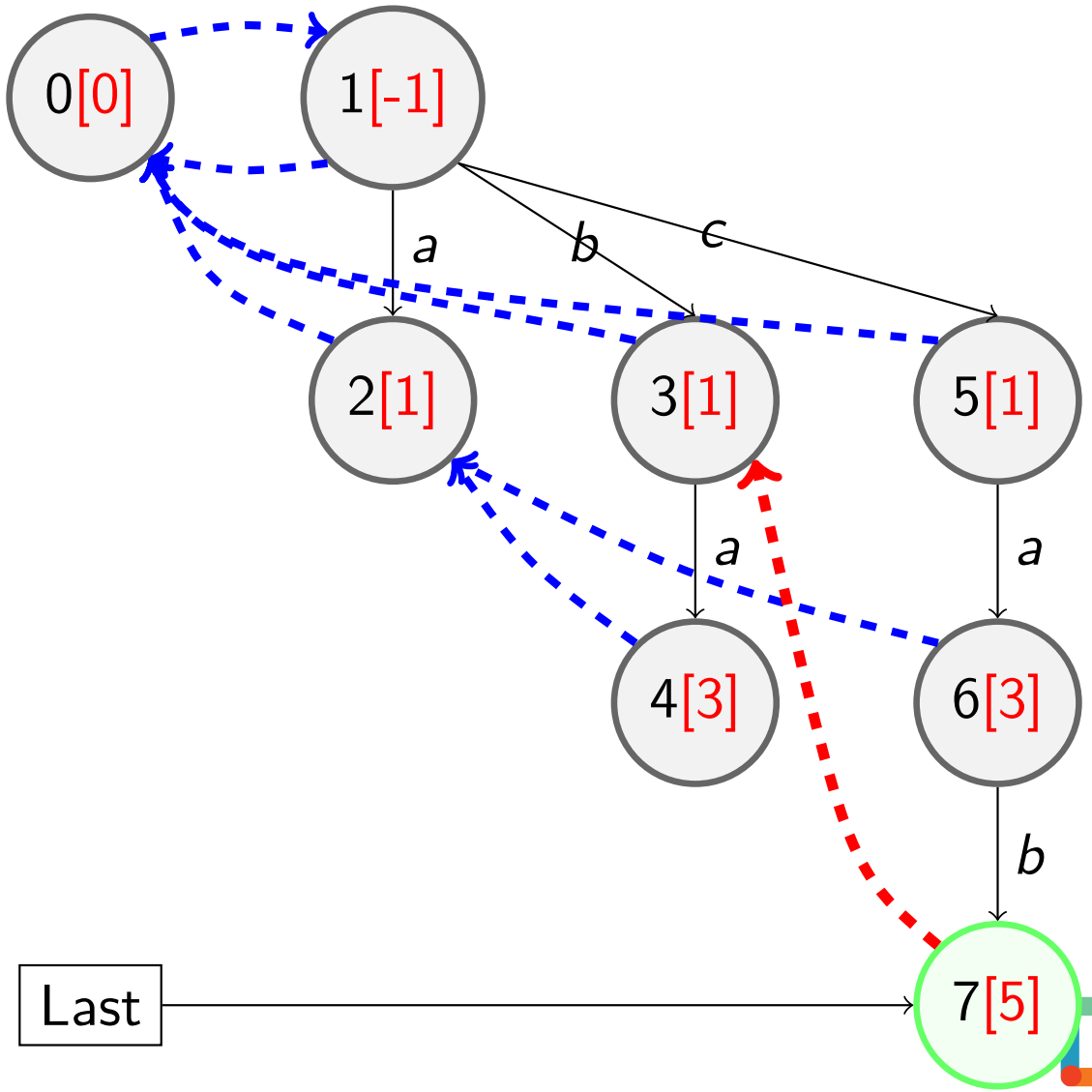
PAM

$S = abacabbaca$.



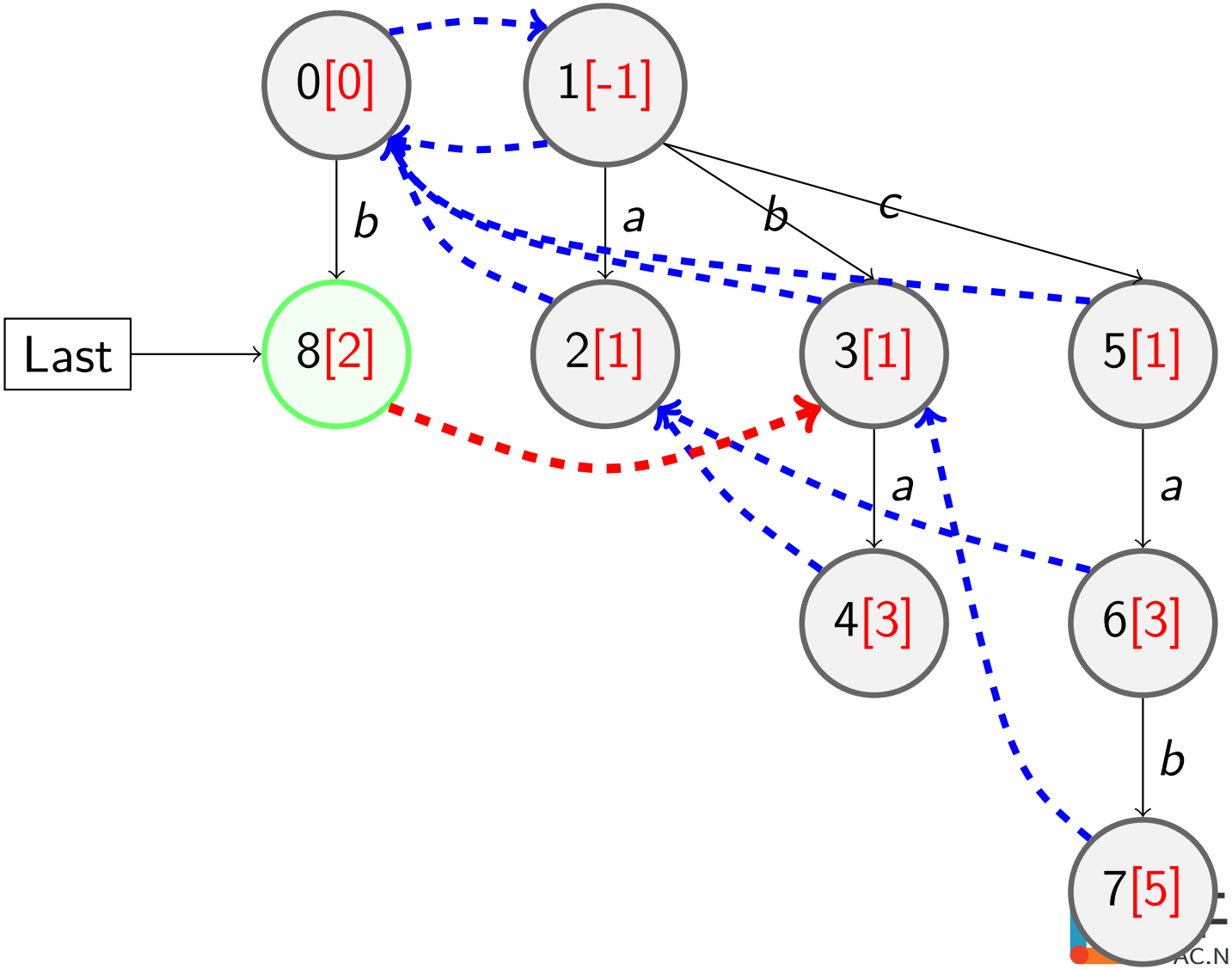
PAM

$S = abacabbaca.$



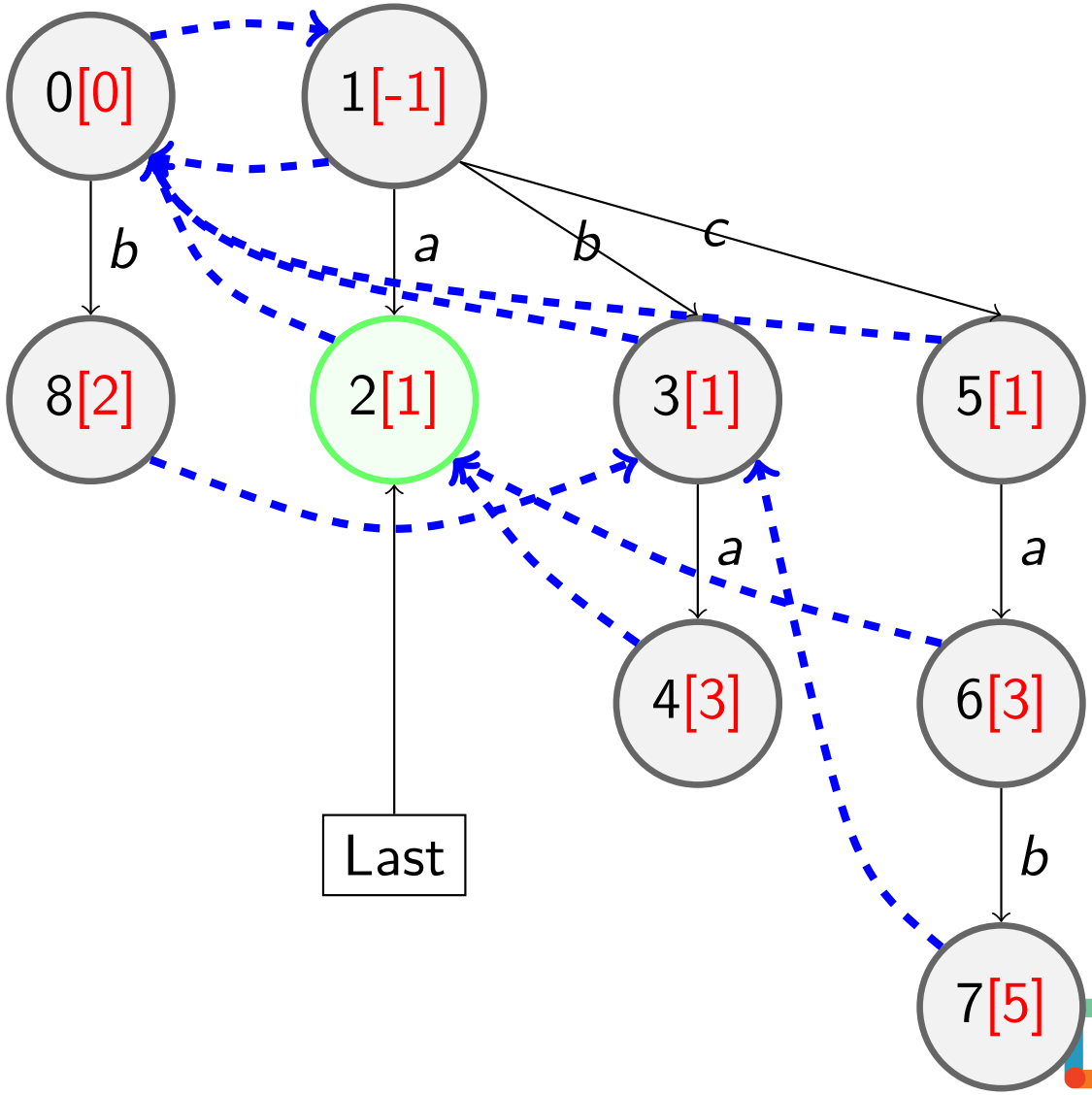
PAM

$S = abacabbaca.$



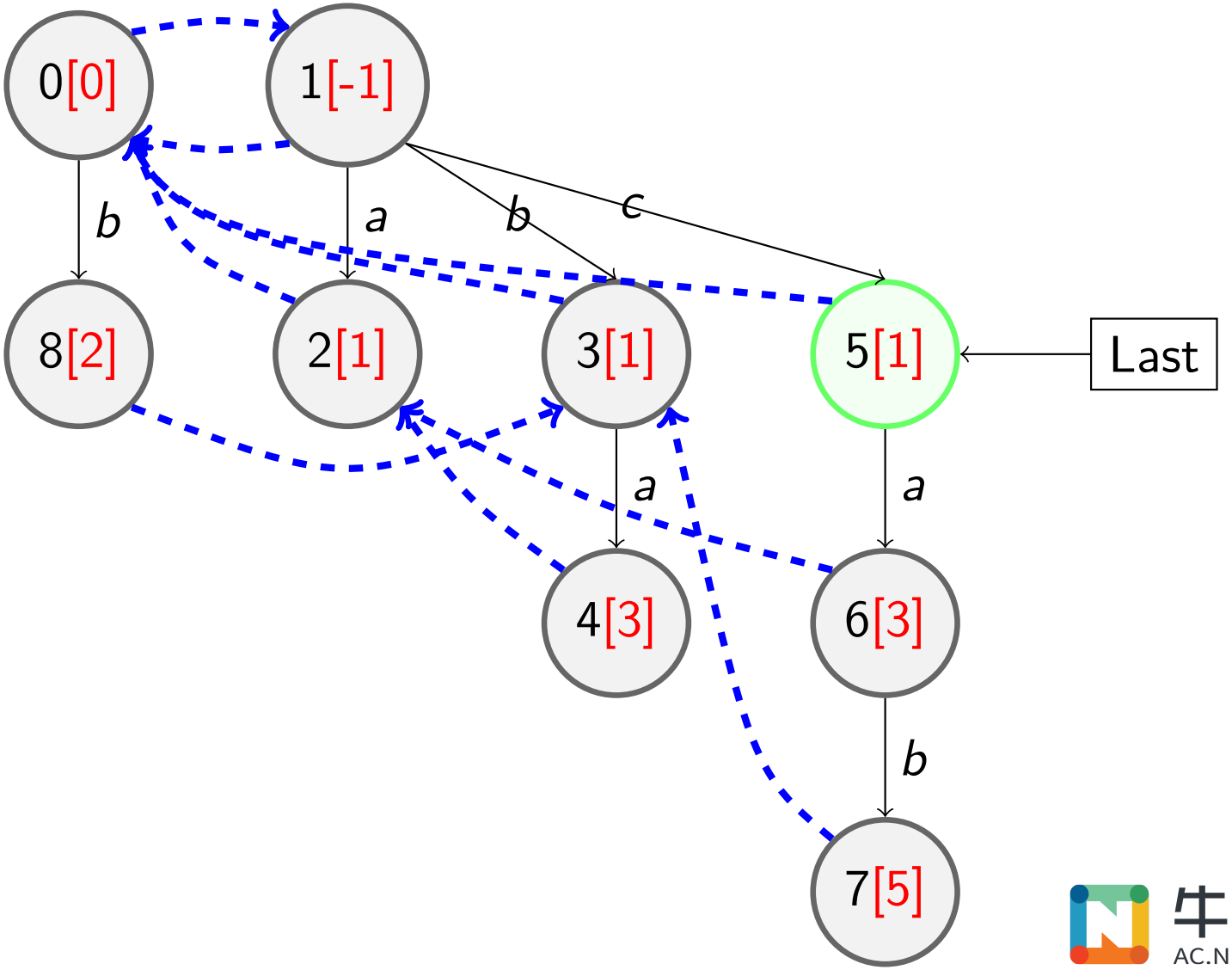
PAM

$S = abacabbaca$.



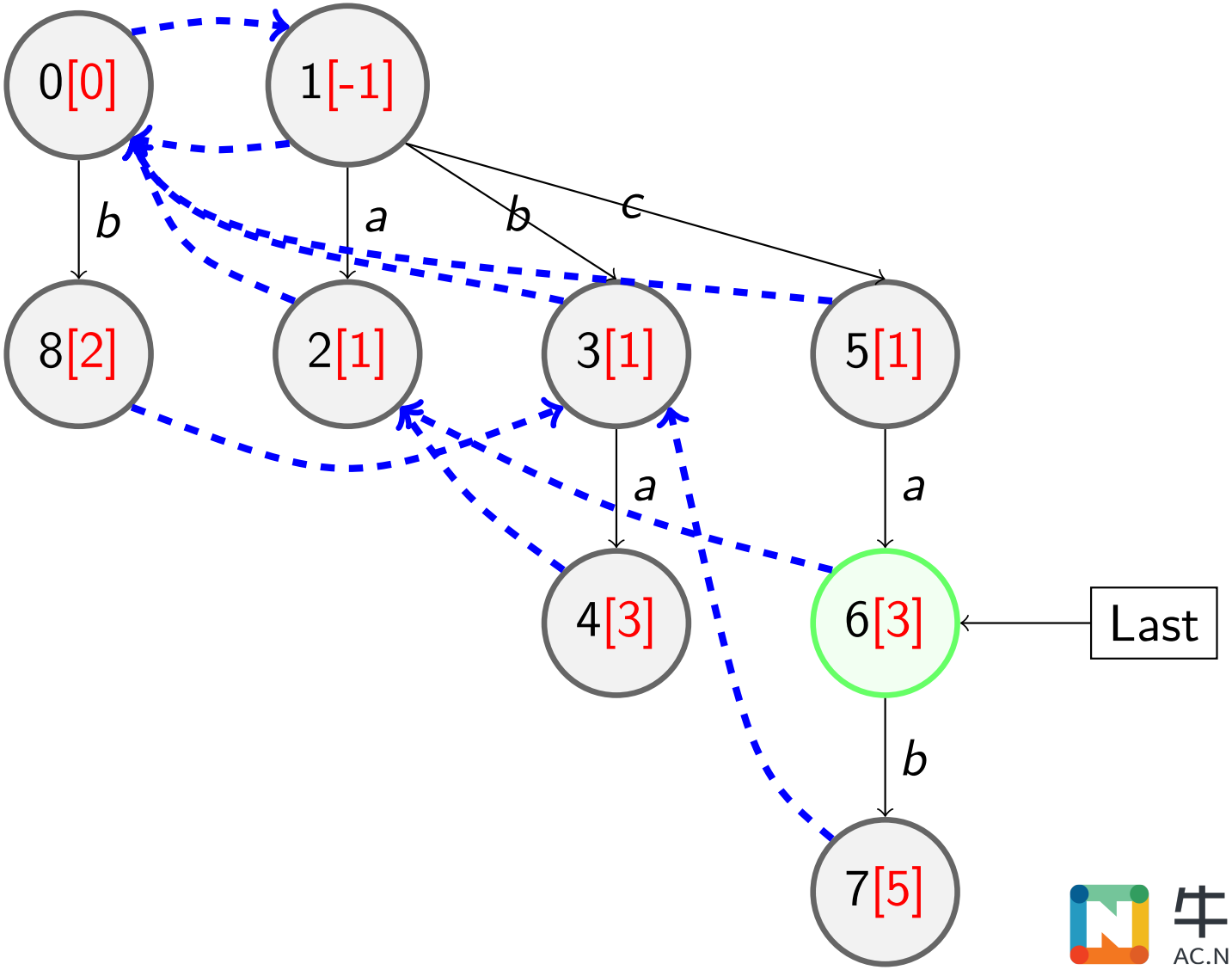
PAM

$S = abacabbaca$.



PAM

$S = abacabbaca$.



PAM

复杂度分析

使用 AC 自动机完全相同的势能分析方法，即可得知时间复杂度为线性。
后继边至多 N 条，朴素数组存边空间复杂度为 $|S| \cdot |\Sigma|$。Hash 表存边可以做到线性，和期望 $O(1)$ 随机访问。
如果字符集太大，可以使用 Map 或者手写线段树存边。

例题 5

某谷 P3649

给出一个字符串 S，定义 S 的子串 T 的价值为 $F(T) = |T| \cdot CNT(T)$，其中 $CNT(T)$ 表示 T 在 S 中的出现次数。
求 S 价值最大的回文子串。

例题 5

某谷 P3649

给出一个字符串 S，定义 S 的子串 T 的价值为 $F(T) = |T| \cdot CNT(T)$，其中 $CNT(T)$ 表示 T 在 S 中的出现次数。
求 S 价值最大的回文子串。

题解

PAM 可以求出所有本质不同子串，且只有 N 个，那么本题实际上就是要统计每种回文串的出现次数。

例题 5

某谷 P3649

给出一个字符串 S，定义 S 的子串 T 的价值为 $F(T) = |T| \cdot CNT(T)$，其中 $CNT(T)$ 表示 T 在 S 中的出现次数。
求 S 价值最大的回文子串。

题解

PAM 可以求出所有本质不同子串，且只有 N 个，那么本题实际上就是要统计每种回文串的出现次数。
可以在构建 PAM 的时候额外维护一个 cnt 数组，表示每个 PAM 节点对应多少个位置的最长回文后缀，也就是每个点有多少次成为了 $Last$ 节点。

例题 5

某谷 P3649

给出一个字符串 S，定义 S 的子串 T 的价值为 $F(T) = |T| \cdot CNT(T)$，其中 $CNT(T)$ 表示 T 在 S 中的出现次数。
求 S 价值最大的回文子串。

题解

PAM 可以求出所有本质不同子串，且只有 N 个，那么本题实际上就是要统计每种回文串的出现次数。
可以在构建 PAM 的时候额外维护一个 cnt 数组，表示每个 PAM 节点对应多少个位置的最长回文后缀，也就是每个点有多少次成为了 $Last$ 节点。
这样每个节点代表的回文串的出现次数等于，Fail 树子树的和。
卡常小技巧：节点编号从大到小就是这棵 Fail 树的拓扑排序。

例题 6

2019 徐州网络赛

给出一个字符串 S，定义一个字符串的价值为：出现字母的种类数。
求 S 所有回文子串的价值之和。

例题 6

2019 徐州网络赛

给出一个字符串 S，定义一个字符串的价值为：出现字母的种类数。
求 S 所有回文子串的价值之和。

题解

本题除了求每个本质不同子串的出现次数外，还需要求每个本质不同子串出现的字母种类数。
可以在 PAM 的每个节点额外维护一个 mask，表示这个点代表的回文串用到了哪些字母，在新建节点的时候顺便维护即可。

拓展姿势

Palindrome Series

用于解决枚举回文后缀的 DP：
由于回文串的特殊性质：回文串的回文后缀一定是 Border。所以枚举回文后缀等价于枚举最大回文后缀的 Border，而 Border 具有良好的等差数列性质，PAM 的 Fail 链接就是 Border 链接。
因此：Palindrome Series= PAM+Border Series
Link