提高算法班

线性基、 Manacher 、AC自动机

Mas

线性基

若向量空间 $\mathcal{V}$ 中的某一向量组 $\mathcal{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3, \cdots\}$ 是 $\mathcal{V}$ 的极大线性无关组

那么 $\mathcal{B}$ 称为向量空间 $\mathcal{V}$ 的一个**线性基** (也称为 **Hamel 基** , 简记为 **基**)

将 $\mathcal{V}$ 的维数记作 $\dim \mathcal{V}$, 定义为基的元素个数

向量空间 $\mathcal{V}$ 的线性基 $\mathcal{B}$ 满足如下性质

- $\mathcal{B}$ 具有极小性 , 即不存在 $\mathcal{B}$ 的子集也为线性基
- $\mathcal{V}$ 中所有向量都能唯一被 $\mathcal{B}$ 唯一线性组合得到

若 $\mathcal{V} = \mathbb{Z}_2^n$ 则称找到的基为 **异或线性基**

若 $\mathcal{V} = \mathbb{R}^n$ 则称找到的基为 **实数线性基**

异或线性基

对于由自然数组成的可重集 S ，将 S 中各元素在二进制下考虑

设 S 中元素最多有 L 个二进制位，那么 S 中各元素可视为 L 维向量

由于异或运算具有封闭性，S 可视为向量空间

考虑构造出 S 的线性基

使用 b[0 … L − 1] 存储 S 的线性基

对于 S 中的每个数 p，二进制下由高到低位扫描插入线性基中

设当前考虑到 p 的第 x 个二进制位，若 p 的第 x 个二进制位为 1

- 若 $\mathrm{b}[x]$ 不存在，令 $\mathrm{b}[x] \leftarrow p$ 并结束扫描
- 若 $\mathrm{b}[x]$ 存在，令 $p \leftarrow p \oplus \mathrm{b}[x]$ 继续向后扫描

若 S 有 n 个元素 且 最多有 L 个二进制位， 构造其线性基时间复杂度 $O(n\mathrm{L})$

异或线性基

考虑构造出 11 , 4 , 19 , 31 的线性基，二进制下考虑 01011_2 , 00100_2 , 10011_2 , 11111_2

插入 01011_2 , $b[3] = 01011_2$

插入 00100_2 , $b[2] = 00100_2$

插入 10011_2 , $b[4] = 10011_2$

插入 11111_2

$$11111_2 \oplus b[4] = 01100_2$$

$$01100_2 \oplus b[3] = 00111_2$$

$$00111_2 \oplus b[2] = 00011_2$$

令 $b[1] = 00011_2$ ，得到线性基

$$\{10011_2, 01011_2, 00100_2, 00011_2\}$$

若再插入 3 ?

异或线性基

不难通过归纳法证明 贪心插入构造 方式的正确性

类似的也可通过归纳法证明 高斯-约旦消元法 的正确性

借鉴 高斯-约旦消元法 的处理思路

若需要令 $\mathrm{b}[x] \leftarrow p$

将 p 的低位 1 消去，同时将高位基第 x 个二进制位消去

即

枚举 $i \in [0, x-1]$

　　若 p 的第 i 个二进制位为 1 令 $p \leftarrow p \oplus \mathrm{b}[i]$

枚举 $i \in [x+1, L]$

　　若 $\mathrm{b}[i]$ 的第 x 个二进制位为 1 令 $\mathrm{b}[i] \leftarrow \mathrm{b}[i] \oplus p$

右侧代码一次插入操作时间复杂度依然为 $O(\mathrm{L})$

```
bool insert(long long p)
{
    for (int i = MAXL; ~i; i--)
        if (p & (1LL << i))
        {
            if (!b[i])   // 需要插入
            {
                for (int j = 0; j < i; j++)  //消去 p 低位
                    if (p & (1LL << j))  p ^= b[j];
                for (int j = i + 1; j <= MAXL; j++) //消去其它基低位
                    if (b[j] & (1LL << i)) b[j] ^= p;
                b[i] = p;
                return true;
            }
            p ^= b[i];
        }
    return false;
}
```

异或线性基

实验舱 青少年编程 走近科学 走进名校

将 S 视作异或线性方程组

进行 高斯-约旦消元 操作，同样可得出 S 的线性基

考虑构造出 11，4，19，31 的线性基，二进制下考虑 01011_2，00100_2，10011_2，11111_2

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

解得

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

不难看出得到结果为行最简矩阵

若 S 有 n 个元素 且 最多有 L 个二进制位， 构造其线性基时间复杂度 $O(n\mathrm{L}^2)$

若使用 bitset 可优化至 $O(\frac{n\mathrm{L}^2}{w})$

异或线性基

贪心构造异或线性基具有如下性质

- 线性基不存在 异或和 为 0 的非空子集
- 线性基中二进制位最高位不同

高斯消元法构造出的线性基满足如下性质:

- 高斯消元后的矩阵是一个行简化阶梯形矩阵

该性质包含了贪心法构造的线性基满足的两条性质

根据定义，异或线性基的 **张成空间** 与 原向量 **张成空间** 一致

那么对于原向量的异或子集可转为对 异或线性基 考虑

#922、最大异或和

题目描述

给由 n 个数组成的一个可重集 S ,求一个集合 $T \subseteq S$

使

$$T_1 \text{ xor } T_2 \text{ xor } \ldots \text{ xor } T_{|T|}$$

最大

输入格式

第一行一个数 n

第二行 n 个数,表示集合 S

输出格式

$T_1 \text{ xor } T_2 \text{ xor } \ldots \text{ xor } T_{|T|}$ 的最大值

数据规模

对于全部的数据 $1 \leq n \leq 1000, 0 \leq S_i \leq 10^{18}$

思路1

若使用贪心插入构造出的线性基（行最简型）

从二进制低位到高位异或所有线性基即可

若从 **低到高** 第 x 位存在线性基 b[x]

这意味着该位异或和可为 1

将其将其加入结果集合

#922、最大异或和

b_x可能消去 b[0], b[1], ⋯ , b[$x-1$] 的贡献

显然 b[0], b[1], ⋯ , b[$x-1$] 贡献之和不可能超过该位的贡献

思路2

若并非行最简型态的线性基

从二进制从 **高到低** 逐位考虑线性基

若异或上线性基 b[x] 能使答案变大，那么将 b[x] 加入结果集合

该策略正确性证明与 思路2 类似

若要求异或最小值，需特殊考虑 0

若要求非 0 最小异或和仅需从 **低到高** 第一个线性基即可

#923、k 大异或和

题目描述

给定你由 N 个整数构成的整数序列
你可以从中选取一些(**至少一个**)进行异或运算,从而得到很多不同的结果

请问所有能得到的不同的结果中第 k 小的结果是多少

注意:只选取一个数字进行运算,则结果为该数字本身

输入格式

第一行包含整数 T ,表示共有 T 组测试数据

对于每组测试数据,第一行包含整数 N

第二行包含 N 个整数(均在 $1 \sim 10^{18}$ 之间),表示完整的整数序列

第三行包含整数 Q ,表示询问的次数

第四行包含 Q 个整数 $k_1, k_2, \ldots, k_Q$,表示 Q 个询问对应的 k

数据规模

对于全部的数据 $1 \leq N, Q \leq 10000, 1 \leq k_i \leq 10^{18}$

输出格式

对于每组测试数据

第一行输出 `Case #C:` ,其中 `C` 为顺序编号(从 1 开始)

接下来 Q 行描述 Q 次询问的结果

每行输出一个整数,表示第 i 次询问中第 k_i 小的结果

如果能得到的不同结果的总数少于 k_i ,则输出 `-1`

求出序列的 异或线性基 $\mathcal{B}$ (将其处理为行最简形式)

并将其按照线性基大小关系排序

向量空间中的向量 $\boldsymbol{u}$ 可被 $\mathcal{B}$ 唯一线性组合得到

对于 $\mathcal{B}$ 任意非空子集 , 不难想到共有 $2^{|\mathcal{B}|} - 1$ 种组合情况

若发现 $|\mathcal{B}| < n$ 说明存在向量能被其它向量 线性组合 得到

那么原序列有 $2^{|\mathcal{B}|}$ 个异或和

若 K 超过异或和个数显然无解

#923、k 大异或和

记 $\mathcal{B}$ 中第 x 小线性基为 $\mathrm{b}[x]$

将 K 二进制拆分 $(\mathrm{k}_L \cdots \mathrm{k}_1 \mathrm{k}_0)_2$ 其二进制具有如下性质：

- 高位上的 1 比低位上的 1 更能使 K 更大
- 低位上的 0 修改为 1 一定会使 K 更大

$\mathcal{B}$ 具有如下性质：

- 选择「控制较高位上的 1 的元素」更能使异或和更大
- 选择「控制较高位上的 1 的元素」后，再选择「控制更低位上 1 的元素」一定会使异或和更大

这说明 $\mathcal{B}$ 的选取与否和 K 的二进制对应

逐位考虑 k_x，若 k_x 为 1 那么将 $\mathrm{b}[x]$ 计入贡献即可

若原序列能够异或得到 0，应当令 K 减一

一次询问时间复杂度 $O(\log \mathrm{K})$

#3267、幂集排名

题目描述

已知一个长度为 n 的正整数序列 A (下标从 1 开始)

令

$$S = \{x|1 \le x \le n\}$$

S 的幂集 2^S 定义为 S 所有子集构成的集合

定义映射

$$f : 2^S \to Z$$

那么

$$f(\emptyset) = 0$$

$$f(T) = \bigoplus_{t \in T} A_t$$

> $\emptyset$ 为空集，$\bigoplus$ 为异或操作

将 2^S 中每个集合的 f 值计算出来，从小到大排成一行记为序列 B (下标从 1 开始)

给定一个数, 那么这个数在序列 B 中第 1 次出现时的下标是多少呢?

输入格式

第一行一个数 n , 为序列 A 的长度

接下来一行 n 个数, 为序列 A , 用空格隔开

最后一个数 Q , 为给定的数

输出格式

共一行一个整数

为 Q 在序列 B 中第一次出现时的下标模 10086 的值

数据规模

对于全部的数据 $1 \le N \le 10^5$ 其他所有输入均不超过 10^9

考虑求出 A 的异或线性基 B

若不考虑重复

根据 Q 的二进制位不难确定其排名

#3267、幂集排名

每个数出现次数均为 $2^{n-|\mathcal{B}|}$ 次

证明

不在线性基中的元素个数为 $n-|\mathcal{B}|$，任选这些元素构成的子集 S

S 中任意元素都能被 $\mathcal{B}$ 唯一线性组合得到，即 $\mathcal{B}$ 有唯一线性组合与 x 的异或和为 0

从 $\mathcal{B}$ 选出一个线性组合 x ，令 x 与 S 中的任意元素 y 搭配

再从 $\mathcal{B}$ 中选出 y 对应组合，显然此时异或和为 y，即 $\boldsymbol{x}$ **至少** 出现 $2^{n-|\mathcal{B}|}$ 次

可能存在 $\mathcal{B}$ 中线性基在 x 的组合中被选取，也在 y 的组合中被选取，但并不影响结论

由于 y 的线性组合唯一，那么 x **至多** 出现 $2^{n-|\mathcal{B}|}$

综上每个数出现次数均为 $2^{n-|\mathcal{B}|}$ 次

求出其排名后 $\times 2^{n-|\mathcal{B}|}$ 即为答案

时间复杂度 $O(n\log\max(Q))$

#3265、装备购买

题目描述

脸哥最近在玩一款神奇的游戏,这个游戏里有 n 件装备,每件装备有 m 个属性

其中第 i 件装备用向量 $\mathbf{z_i} = (a_1, \ldots, a_j, \ldots, a_m)$ 表示

每个装备需要花费 c_i ,现在脸哥想买一些装备,但是脸哥很穷,所以总是盘算着怎样才能花尽量少的钱买尽量多的装备

对于脸哥来说,如果一件装备的属性能用购买的其他装备组合出(也就是说脸哥可以利用手上的这些装备组合出这件装备的效果)那么这件装备就没有买的必要了

严格的定义是,如果脸哥买了 $\mathbf{z_{i_1}}, \ldots, \mathbf{z_{i_p}}$ 这 p 件装备

那么对于任意待决定的 $\mathbf{z_h}$,不存在 $b_1, \ldots, b_p$ 使得

$$b_1\mathbf{z_{i_1}} + \ldots + b_p\mathbf{z_{i_p}} = \mathbf{z_h}$$

其中 b_i 均是实数

那么脸哥就会买 $\mathbf{z_h}$,否则 $\mathbf{z_h}$ 对脸哥就是无用的了,自然不必购买

如 $\mathbf{z_1} = (1,2,3),\ \mathbf{z_2} = (3,4,5),\ \mathbf{z_h} = (2,3,4),\ b_1 = \frac{1}{2},\ b_2 = \frac{1}{2}$

就有 $b_1\mathbf{z_1} + b_2\mathbf{z_2} = \mathbf{z_h}$,那么如果脸哥买了 $\mathbf{z_1}$ 和 $\mathbf{z_2}$ 就不会再买 $\mathbf{z_h}$ 了

脸哥想要在买下最多数量的装备的情况下花最少的钱,你能帮他算一下吗?

输入格式

第一行两个数 n, m

接下来 n 行,每行 m 个数,其中第 i 行描述装备 i 的各项属性值

接下来一行 n 个数,其中 c_i 表示购买第 i 件装备的花费

输出格式

输出两个数
第一个数表示能够购买的最多装备数量
第二个数表示在购买最多数量的装备的情况下的最小花费

数据规模

对于 100% 的数据, $1 \le n, m \le 500, 0 \le a_j \le 1000$

#3265、装备购买

最终购买装备为一极大线性无关组，若 $\{\boldsymbol{z}_1,\boldsymbol{z}_2,\cdots,\boldsymbol{z}_n\}$ 实数线性基为 $\mathcal{B}$ 则第一问答案为 $|\mathcal{B}|$

现要求 $\mathcal{B}$ 的代价和最小，可将各向量按价值升序排序，插入构造线性基

设 $\{\boldsymbol{z}_{i_1},\boldsymbol{z}_{i_2},\cdots,\boldsymbol{z}_{i_p}\}$ 为代价和最小线性基，且其不包含代价最小向量 $\boldsymbol{z}_k$

根据线性基性质 $\boldsymbol{z}_k$ 可被线性组合得到即

$$\boldsymbol{z}_k = b_1\boldsymbol{z}_{i_1} + b_2\boldsymbol{z}_{i_2} + \cdots + b_p\boldsymbol{z}_{i_p} \Rightarrow \boldsymbol{z}_{i_p} = \frac{b_1\boldsymbol{z}_{i_1} + b_2\boldsymbol{z}_{i_2} + \cdots - \boldsymbol{z}_k}{b_p}$$

可用 $\boldsymbol{z}_k$ 替换 $\boldsymbol{z}_{i_p}$ 得到结果不劣

对于当前插入向量 $\boldsymbol{z}$，若无法放入 $b[x]$ 需将 $\boldsymbol{z}$ 第 x 个分量消去

本题对精度要求较高需使用 long double

时间复杂度 $O(nm^2)$

同样也可使用 高斯-约旦消元法 求解

Manacher

满足 $\forall\, 0 \le i < |S|,\ S[i] = S[\,|S| - 1 - i\,]$ 的字符串 S 被称为回文串

考虑问题：从字符串 S 中找出所有回文子串

在最坏情况下可能有 $O(n^2)$ 个回文串，若枚举回文串中心暴力往两侧扩展 时间复杂度 $O(n^2)$

考虑用一种更紧凑的方式表达回文串的信息

对于每个位置 $0 \le i < |S|$

- 令 $d_1[i]$ 表示以 i 为中心的长度为奇数的回文串个数
- 令 $d_2[i]$ 表示以 i 为中心的长度为偶数的回文串个数

二者也表示了以 i 为中心的最长回文串的半径长度

半径长度 $d_1[i]$，$d_2[i]$ 均为从位置 i 到回文串最右端位置包含的字符个数

以字符串 S = “abababc” 为例

Manacher

$S[3] = \mathrm{b}$ 存在 b, abc, bavab 三个奇数长度的回文串

$$\mathrm{a}\,\overbrace{\mathrm{b\,a\,b\,a\,b}}^{\mathrm{d_1[3]=3}}\,\mathrm{c}$$

以字符串 $S =$ “cbaabd” 为例

$S[3] = \mathrm{a}$ 存在 a, baab 两个偶数长度的回文串

$$\mathrm{c}\,\overbrace{\mathrm{b\,a\,a\,b}}^{\mathrm{d_2[3]=2}}\,\mathrm{d}$$

先考虑 $\mathrm{d_1}$ 的求解

维护右端点最靠右的回文区间 $[\mathrm{L}, \mathrm{R}]$，初始时不妨令 $\mathrm{L} = \mathrm{R} = 0$

考虑从 $0 \sim |S| - 1$ 顺次计算 $\mathrm{d_1}[i]$，在计算$\mathrm{d_1}[i]$ 的过程中利用已计算好的 $\mathrm{d_1}[0], \ldots, \mathrm{d_1}[i-1]$

假设 $\mathrm{d_1}[0 \cdots i-1]$ 已经求解完成， 考虑 $\mathrm{d_1}[i]$ 的求解

在计算 $\mathrm{d_1}[i]$ 的过程中

- 若 $i \le \mathrm{R}$

Manacher

i 在 $[\mathrm{L},\mathrm{R}]$ 中对称位置 $j=\mathrm{L}+\mathrm{R}-i$

$$\cdots \overbrace{S_L \cdots \underbrace{S_{j-d_1[j]+1} \cdots S_j \cdots S_{j+d_1[j]-1}}_{\text{palindrome}} \cdots \underbrace{S_{i-d_{1[j]}+1} \cdots S_i \cdots S_{i+d_1[j]-1}}_{\text{palindrome}} \cdots S_R}^{\text{palindrome}} \cdots$$

显然 $\mathrm{d}_1[i] \ge \min(\mathrm{d}_1[j], \mathrm{R}-i+1)$

若 $\mathrm{d}_1[j] < \mathrm{R}-i+1$

则 $\mathrm{d}_1[i] = \mathrm{d}_1[i-\mathrm{L}]$

否则 $\mathrm{d}_1[j] \ge \mathrm{R}-i+1$

此时 $\mathrm{S}_{i+d_1[j]-1}$ 将落在 $[\mathrm{L},\mathrm{R}]$ 区间外

不妨先令 $\mathrm{d}_1[i] = \mathrm{R}-i+1$

再枚举下一个字符扩展 $\mathrm{d}_1[i]$ 直到无法扩展

Manacher

- 若 $i > \mathrm{R}$

  从 $\mathrm{S}[i]$ 开始比较，暴力求出 $\mathrm{d}_1[i]$

在求出 $\mathrm{d}_1[i]$ 后，若 $i + \mathrm{d}_1[i] - 1 > \mathrm{R}$

需要更新 $[\mathrm{L}, \mathrm{R}]$，即令 $\mathrm{L} = i$， $\mathrm{R} = i + \mathrm{d}_1[i] - 1$

d_2 的求解与 $\mathrm{d}_1[i]$ 类似仅需考虑一些边界细节

与 **Z Algorithm** 类似，不难看出 R 至多移动 |S| 次

时间复杂度 $O(|\mathrm{S}|)$

该算法由 Glenn K. Manacher 在 1975 年提出，被称为 Manacher 算法

国内因其音译与 马拉车 类似，也会将其称为马拉车算法）

Manacher

将字符串 S 进行改造

在 S 中间隔插入 |S| + 1 个 # , 得到长度为 2|S| + 1 的字符串 S'

如 S = “abababc”

对应的 S' = “#a#b#a#b#a#b#c#”

S' 中的 # 对应 S 的空 , 即将对称点全部转到字符上考虑

S 中计算的 $d_1[i]$ 在 S' 中必以 # 结尾

S 中以字母为对称点的极大回文子串

设其最大半径为 $m + 1$

则该极大回文串在 S' 中必然也以对应字符为中心

在 S' 中为最大半径为 $2m + 2$ 的极长回文子串

$$a\ \overbrace{b\ a\ b}^{m+1}\ a\ b\ c$$

$$\#\ a\ \overbrace{\#\ b\ \#\ a\ \#\ b}^{2m+2}\ \#\ a\ \#\ b\ \#\ c\ \#$$

Manacher

S 中计算的 $d_2[i]$ 在 S' 中也必以 # 结尾

S 中以 空 为对称点的极大回文子串

在 S' 中以 # 为对称点

设其长度为 $m+1$

在 S' 中为最大半径为 $2m+1$ 的极长回文子串

综上 S' 中 $d_1[i]-1$ 即为 S 中极长回文子串长度

$$\mathrm{a}\overbrace{\mathrm{c\ a}}^{\mathrm{m}}\mathrm{b\ b\ a\ c}$$

$$\#\ \mathrm{a}\overbrace{\#\ \mathrm{c}\ \#\ \mathrm{a}\ \#\ \mathrm{b}\ \#}^{2\mathrm{m}+1}\ \mathrm{b}\ \#\ \mathrm{a}\ \#\ \mathrm{c}\ \#$$

该结论建立了 S' 的 d_1 同 S 的 d_1 和 d_2 间的关系

所以仅需对 S' 使用 d_1 的求解方式，即可得出 S 的 d_1 和 d_2

上述分析建立在 $2 \mid m$ 的基础上，但并不影响最终结论

#3914、 Manacher

题目描述

给出一个只由小写英文字符组成的字符串 S

求 S 中最长回文串的长度

输入格式

一行一个小写英文字符字符串 S

输出格式

一个整数表示答案

输入样例

```
aaa
```

输出样例

```
3
```

数据规模

对于全部的数据 $1 \le |S| \le 10^7$

```cpp
vector<int> getD(string s)
{
    int n = s.size();
    vector<int> d(n, 1);
    for (int i = 0, l = 0, r = 0; i < n; i++)
    {
        if (i <= r)
            d[i] = min(d[l + r - i], r - i + 1);
        while (i - d[i] + 1 > 0 && i + d[i] < n && s[i - d[i]] == s[i + d[i]])
            d[i]++;
        if (i + d[i] - 1 > r)
            l = i - d[i] + 1, r = i + d[i] - 1;
    }
    return d;
}

vector<int> manacher(string s)
{
    string t;
    for (auto &&c : s)
        t += string("#") + c;
    auto res = getD(t + "#");
    return vector<int>(begin(res) + 1, end(res) - 1);
}
```

#2795、扩展回文串

题目描述

输入多个字符串

对于每个字符串 S ,求出一个字符串 S'

S' 需要满足：

- S 为 S' 的前缀
- S' 是一个回文字符串
- $|S'|$ 应尽可能小

对于每个 S ,输出 S' ,每行输出以换行符结尾

输入格式

输入包含多组数据

每组输入一各仅由小写字母组成的字符串 S

输出格式

每组输出输出一行 S'

数据规模

对于全部的数据, $|S|$ 仅由大小写字母组成 $|S| \le 10^5, \sum |S| \le 10^6$

记 $\mathbf{rev}(\mathrm{S})$ 为 S 的翻转串

$\mathrm{S} + \mathbf{rev}(\mathrm{S})$ 必然是回文串，考虑将其变短

思路4

$$\mathrm{s}\overbrace{\mathrm{y\,c\,y}}^{\mathrm{S}}\quad \underbrace{\mathrm{y\,c\,y}}_{\mathbf{rev}(\mathrm{S})}\mathrm{s}$$

不难发现 公共部分 必为 S 的后缀

且容易证明该后缀必为回文串

仅需找出右边界为 $|\mathrm{S}| - 1$ 的最长回文串

在 S' 中满足 $i + \mathrm{d}_1[i] - 1 = |S'|$ 即为满足上述条件的回文串

Manacher 求解即可，时间复杂度 $O(\sum|\mathrm{S}|)$

Aho-Corasick automaton

AC 自动机(Aho – Corasick automaton)

该算法在 1975 年产生于贝尔实验室，是一种多模匹配算法是对 Trie 的扩展

AC 自动机是以 Trie 的结构为基础,结合 KMP 的思想建立

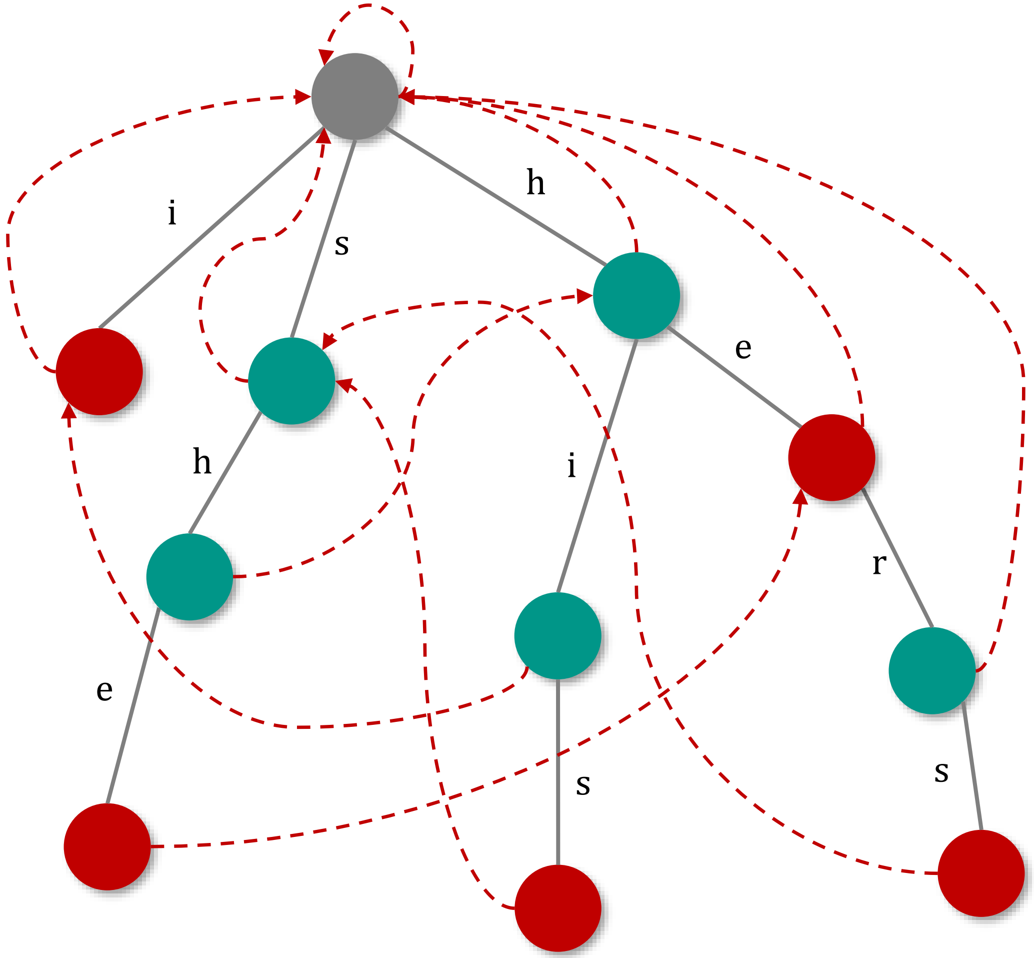
建立一个 AC 自动机有两个步骤

- 基础的 Trie 结构

  将所有的模式串构成一棵 Trie

- KMP 的思想

  对 Trie 树上所有的结点构造失配指针



i

he

his

she

hers

Aho-Corasick automaton

从根节点开始尝试匹配　u s h e r s h e i

she 匹配成功

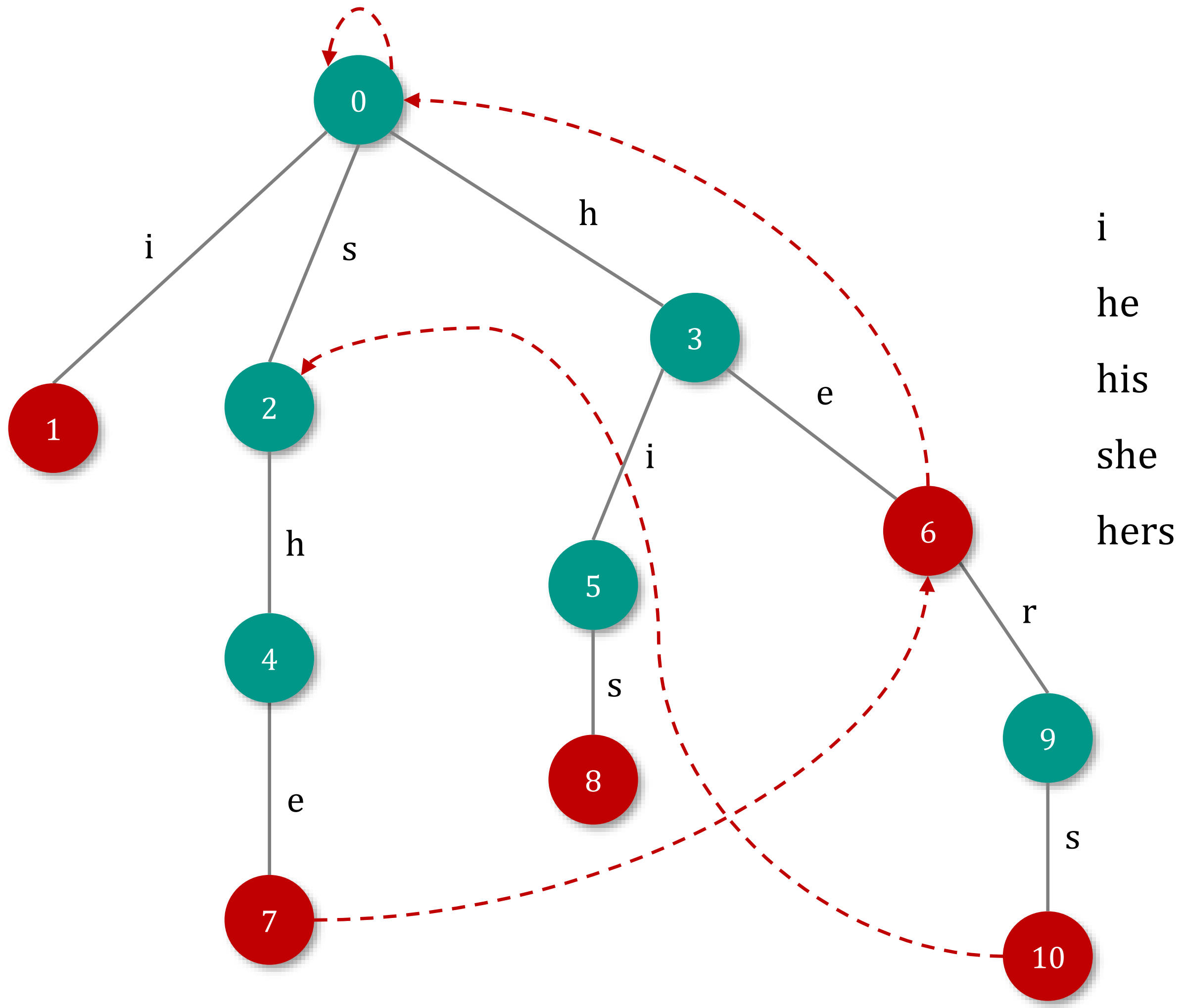
he 匹配成功

hers 匹配成功

she 匹配成功

he 匹配成功

i 匹配成功



Aho-Corasick automaton

Trie 中结点表示某个模式串的前缀，后续将其称作状态

一个结点表示一个状态， Trie 的边就是状态的转移

对于若干个模式串 $\mathrm{T}_1, \mathrm{T}_2 \ldots, \mathrm{T}_n$ 将它们构建 Trie 后所有状态集合记作 Q

AC 自动机利用 fail 指针来辅助多模式串匹配

状态 u 的 fail 指针指向另一个状态 v

其中 $v \in Q$ 且 v 是 u 的最长后缀(即在若干个后缀状态中取最长作为 fail 指针)

fail 指针与 π 数组对比

- 共同点：两者都用于失配时跳转
- 不同点： π 数组为最长真公共前后缀

  fail 指针指向所有模式串前缀中匹配当前状态的最长后缀

Aho-Corasick automaton

考虑当前的结点 u , 设 u 的父结点为 p

p 通过字符 c 的边指向 u , 即 $\delta(p,c)=u$

假设深度小于 u 的所有结点的 fail 指针都已求得

- 若 $\delta(\mathrm{fail}_p\,, c)$ 存在

  令 $\mathrm{fail}_u \leftarrow \delta(\mathrm{fail}_p\,, c)$

  相当于在 p 和 fail_p 后加一字符 c , 分别对应 u 和 fail_u

- 若 $\delta(\mathrm{fail}_p\,, c)$ 不存在

  那么继续找到 $\delta\left(\mathrm{fail}_{\mathrm{fail}_p}\,, c\right)$ 重复上一判断过程

  直到 fail 指针指向根结点

将模式串插入 Trie 后， BFS 求解即可

Aho-Corasick automaton

重点观察节点 8 的 fail 指针构造

找到节点8 的父结点 5

其中 $fail_5 = 1$

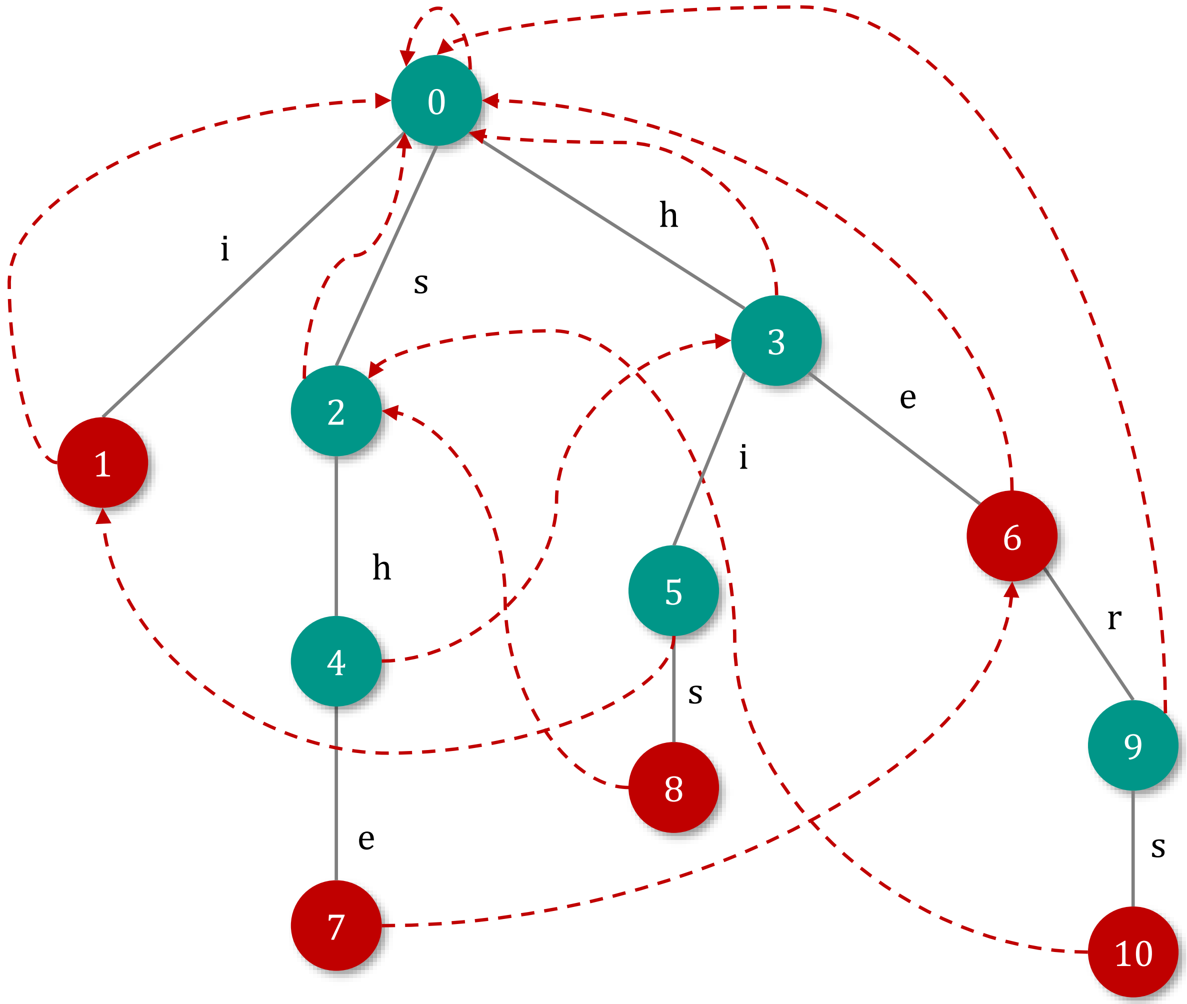
然而节点 1 不存在字符 s 连出边

继续跳到 $fail_1$

其中 $fail_1 = 0$

节点 0 存在字符 s 连出的边，指向节点 2

所以 $fail_8 = 2$



Aho-Corasick automaton

当 BFS 遍历到节点 5 时，原策略是找 fail 指针

跳到 $\mathrm{fail}_5 = 1$ 发现不存在字符 s 连出的边

于是跳到 $\mathrm{fail}_1 = 0$

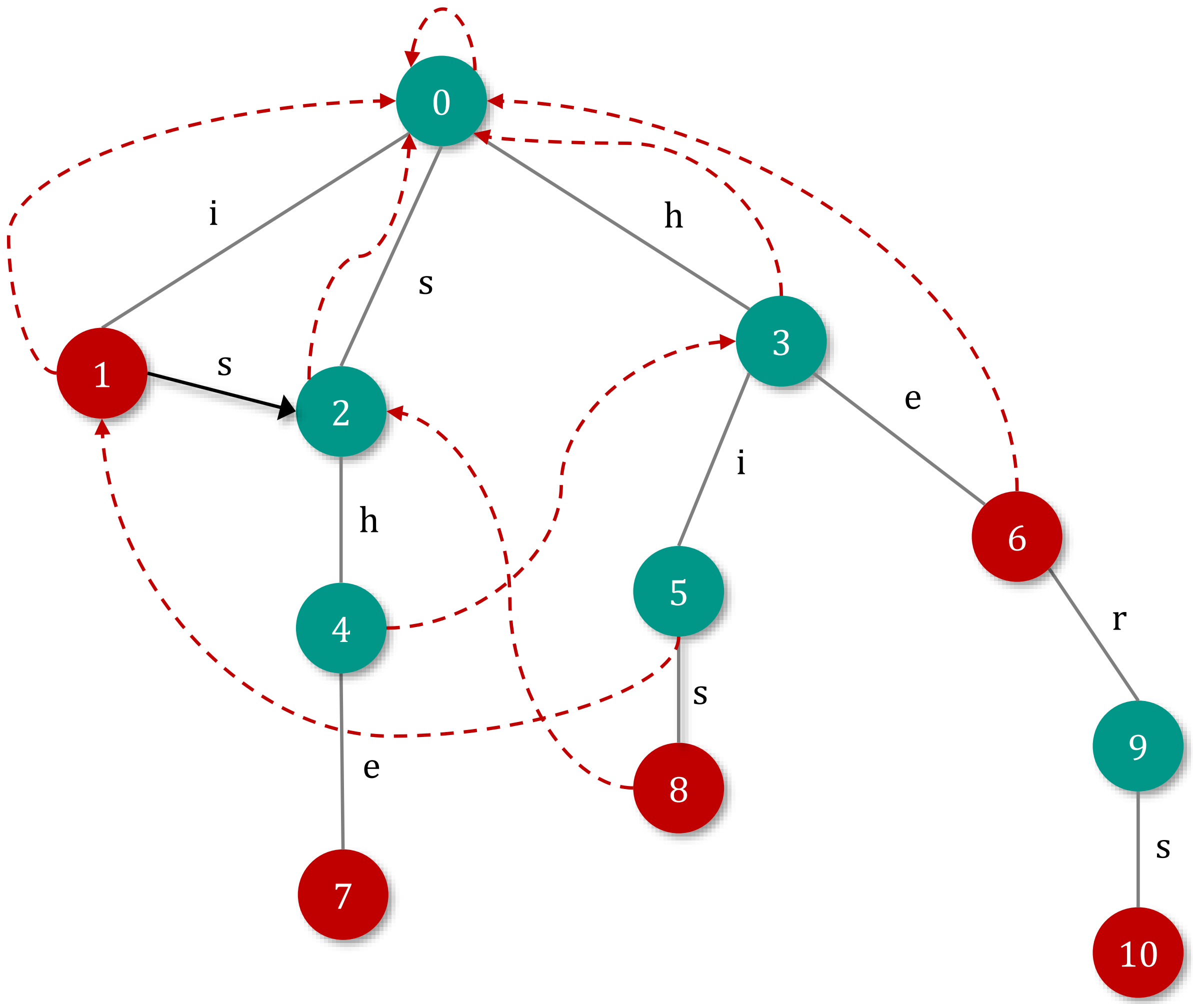
发现有 $\delta(0,'s') = 2$，于是 $\mathrm{fail}_8 = 2$

但是有了黑边后，跳到 $\mathrm{fail}_5 = 1$ 后

直接走 $\delta(0,'s') = 2$ 可到达结点 2

增加黑色边虽然修改了 Trie 结构但不影响后续查询

将 Trie 从树形变为了字典图



Aho-Corasick automaton

```
void build()
{
    queue<int> q;
    for (int i = 0; i < 26; i++)
        if (t[0][i])
            q.push(t[0][i]);
    while (q.size())
    {
        int p = q.front();
        q.pop();
        for (int i = 0; i < 26; i++)
            if (t[p][i])
                fail[t[p][i]] = t[fail[p]][i], q.push(t[p][i]);
            else
                t[p][i] = t[fail[p]][i];
    }
}
```

能否将 Trie 根节点直接入队?

若直接将根入队,那么根的子节点的 fail 指针将指向自己

与实际情况不符

若主串为 S , 有 n 个模式串 $\mathrm{T}_1, \mathrm{T}_1, \cdots, \mathrm{T}_n$, 字符集大小为 K

构建 AC 自动机时间复杂度为 $O(\sum|\mathrm{T}_i| \times K)$

#657、AC 自动机

题目描述

给定 n 个长度不超过 50 的由小写英文字母组成的单词准备查询
以及一篇长为 m 的文章

文中出现了多少个待查询的单词

多组数据

输入格式

第一行一个整数 T ,表示数据组数

对于每组数据,第一行一个整数 n
接下去 n 行表示 n 个单词,最后一行输入一个字符串,表示文章

输出格式

对于每组数据,输出一个数
表示文中出现了多少个待查询的单词

数据范围

对于全部数据, $1 \leq n \leq 10^4, 1 \leq m \leq 10^6$

样例输入

```
1
5
she
he
say
shr
her
yasherhs
```

样例输出

```
3
```

#657、AC 自动机

在 AC 自动机上移动

若到达单词节点，累加次数

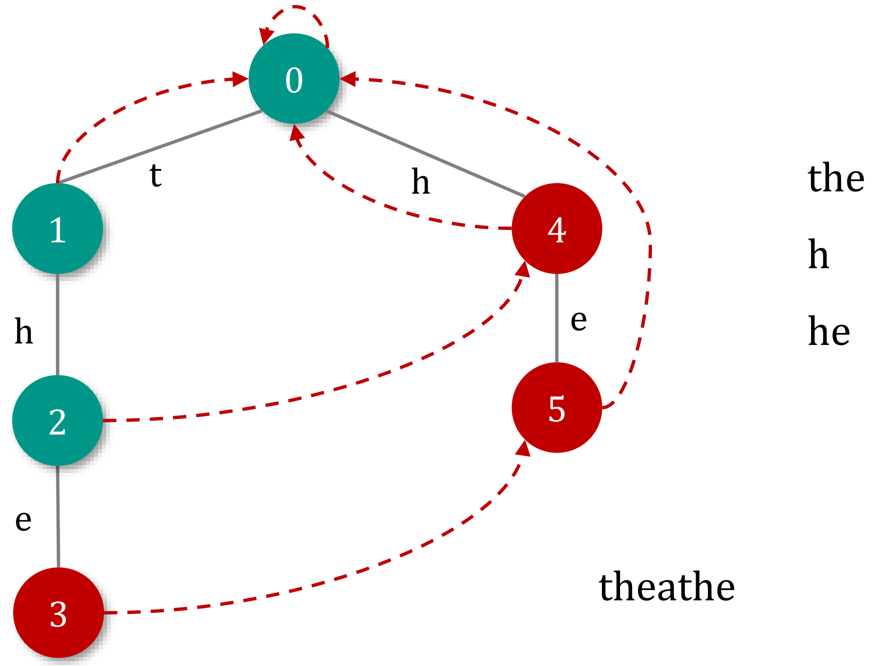
对于每一个节点都应通过 fail 指针上跳

统计前缀中出现的模式串

为了避免重复统计应将其次数清空

若主串为 S，有 n 个模式串 $\mathrm{T}_1, \mathrm{T}_1, \cdots, \mathrm{T}_n$，字符集大小为 K

一次查找时间复杂度为 $O(|\mathrm{S}| + \sum|\mathrm{T}| \times K)$




```
int query(char str[])
{
    int p = 0, res = 0;
    for (int i = 0; str[i]; i++)
    {
        p = t[p][str[i] - 'a'];
        for (int j = p; j && cnt[j] != -1; j = fail[j])
            res += cnt[j], cnt[j] = -1;
    }
    return res;
}
```

#3268、又是AC自动机

题目描述

有 N 个由小写字母组成的模式串 $S_1, S_2, \cdots, S_N$ 以及一个文本串 T

每个模式串可能会在文本串中出现多次

你需要找出哪些模式串在文本串 T 中出现的次数最多

输入格式

输入含多组数据

保证输入数据不超过 50 组

每组数据的第一行为一个正整数 N ,表示共有 N 个模式串

接下来 N 行,每行一个模式串 S_i

下一行是文本串 T

保证不存在两个相同的模式串

输入结束标志为 $N = 0$

输出格式

对于每组数据,第一行输出模式串最多出现的次数

接下去若干行每行输出一个出现次数最多的模式串,按输入顺序排列

数据规模

对于全部的数据 $1 \le N \le 150, 1 \le |S_i| \le 70, 1 \le |T| \le 10^6$

```
int query(char str[])
{
    int p = 0, res = 0;
    for (int i = 0; str[i]; i++)
    {
        p = t[p][str[i] - 'a'];
        for (int j = p; j; j = fail[j])
            cnt[j]++;
    }
    for (int i = 1; i <= pos; i++)
        if (idx[i])
            res = max(res, cnt[i]);
    return res;
}
```

时间复杂度

是否依然为 $O(|S| + \sum|T| \times K)$?

#3269、还是AC自动机

实验舱 青少年编程 LAB 走近科学 走进名校

题目描述

给你一个文本串 S 和 n 个模式串 $T_{1\sim n}$

请你分别求出每个模式串 T_i 在 S 中出现的次数

输入格式

第一行包含一个正整数 n 表示模式串的个数

接下来 n 行,第 i 行包含一个由小写英文字母构成的非空字符串 T_i

最后一行包含一个由小写英文字母构成的非空字符串 S

数据不保证任意两个模式串不相同

输出格式

输出包含 n 行

其中第 i 行包含一个非负整数表示 T_i 在 S 中出现的次数

数据规模

对于 100% 的数据, $1 \le n \le 2 \times 10^5, T_{1\sim n}$ 的长度总和不超过 2×10^5

保证 S 的长度不超过 2×10^6

#3269、还是AC自动机

若字符集大小为 K

朴素上跳 fail 指针，最坏情况下单词查找时间复杂度接近 $O(|S| \times \sqrt{\sum|T| \times K})$

考虑 fail 指针所代表边，不难发现各节点出度为 1

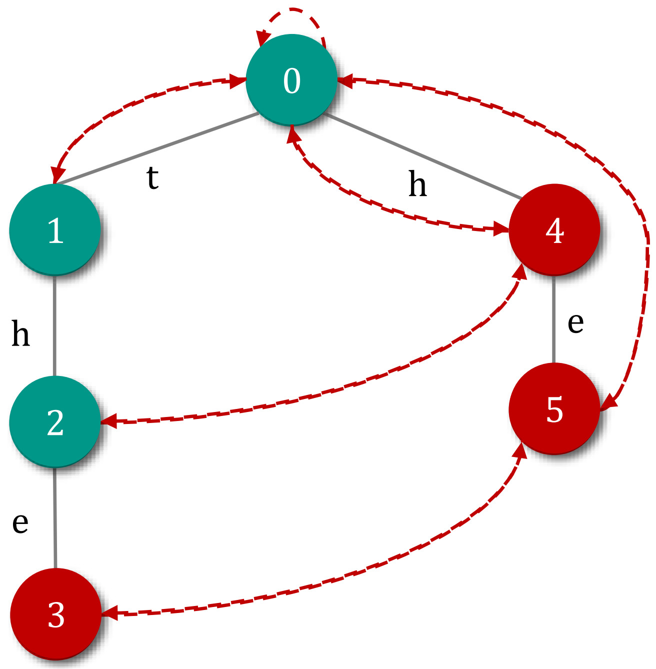
可将 fail 指针所代表边建成一颗 fail 树

对于每次 fail 指针上跳统计

本质为 fail 树上做 点到根的路径 统计

可转为 树上点差分 维护

一次查找时间复杂度为 $O(|S| + \sum|T| \times 26)$



实验舱

青少年编程

走近科学 走进名校

谢谢观看