字串演算法基石

樂正

IOICAMP 2017

fenzhangs@gmail.com

February 11, 2017

字串是什麼

字串是什麼

Definition (字元集)

字元集是一個符號集合，記做 Σ。字元是 Σ 中的元素。

Definition (字串)

給定一個字元集 Σ，一個**字串**是一些字元的有序序列，我們寫作

$$A = a_1 a_2 \cdots a_{|A|}$$

其中 $a_i \in \Sigma$。我們把 $|A|$ 叫作字串的長度。

字串是什麼

Definition (字元集)

字元集是一個符號集合，記做 Σ。字元是 Σ 中的元素。

Definition (字串)

給定一個字元集 Σ，一個**字串**是一些字元的有序序列，我們寫作

$$A = a_1 a_2 \cdots a_{|A|}$$

其中 $a_i \in \Sigma$。我們把 $|A|$ 叫作字串的長度。

- 小寫英文字母是一個字元集、鋼系元素是一個字元集。
- 基因是由字元集 $\{A, T, C, G\}$ 組成的字串。
- C 語言中 c-string 是 $\{1, 2, 3, 4, \cdots, 255\}$ 組成的字串，0 作為結束字元。

字串是什麼

Definition (子字串 (Substring))

給定一個字串 $A = a_1 a_2 \cdots a_{|A|}$，設 $n = |A|$。
一個**子字串**是其連續的一段 $a_i a_{i+1} a_{i+2} \cdots a_j$ 記作 $A[i, j]$ 或 $A[i, j+1)$。

如果 $A =$ `"hellohello"`，則

- $A[2, 4] = A[2, 5) =$ `"ell"`，$a_2 =$ `"e"`, $a_3 =$ `"l"`, $a_4 =$ `"l"`
- $A[1, 3] = A[1, 4) =$ `"hel"`，$a_1 =$ `"h"`, $a_2 =$ `"e"`, $a_3 =$ `"l"`
- `"heo"` 是 A 的子序列，但不是 A 的子字串

字串是什麼

Definition (子序列 (Subsequence))

給定一個字串 $A = a_1 a_2 \cdots a_{|A|}$，設 $n = |A|$。
一個**子序列**是一個字串 $B = a_{q_1} a_{q_2} a_{q_3} \cdots a_{q_m}$，其中
$1 \leq q_1 < q_2 < q_3 < \cdots < q_{m-1} < q_m \leq n$。

如果 $A =$ `"hellohello"`，則

- $q_1 = 1, q_2 = 2, q_3 = 5$ 時 $B =$ `"heo"`，B 是 A 的子序列，但不是 A 的子字串。
- $q_1 = 5, q_2 = 6, q_3 = 7, q_4 = 8$ 時 $B =$ `"ohel"`，B 是 A 的子序列，也是 A 的子字串。

字串是什麼

Definition (前綴 (Prefix))

給定一個字串 $A = a_1 a_2 \cdots a_{|A|}$，設 $n = |A|$。
一個 A 的**前綴**是 A 的一個子字串 $a_1 a_2 a_3 \cdots a_h$，其中 $h \leq n$，記作 $P_A(h)$。

如果 $A =$ `"hello"`，則

- $P_A(1) =$ `"h"`
- $P_A(2) =$ `"he"`
- $P_A(3) =$ `"hel"`
- $P_A(4) =$ `"hell"`
- $P_A(5) =$ `"hello"`

字串是什麼

Definition (後綴 (Suffix))

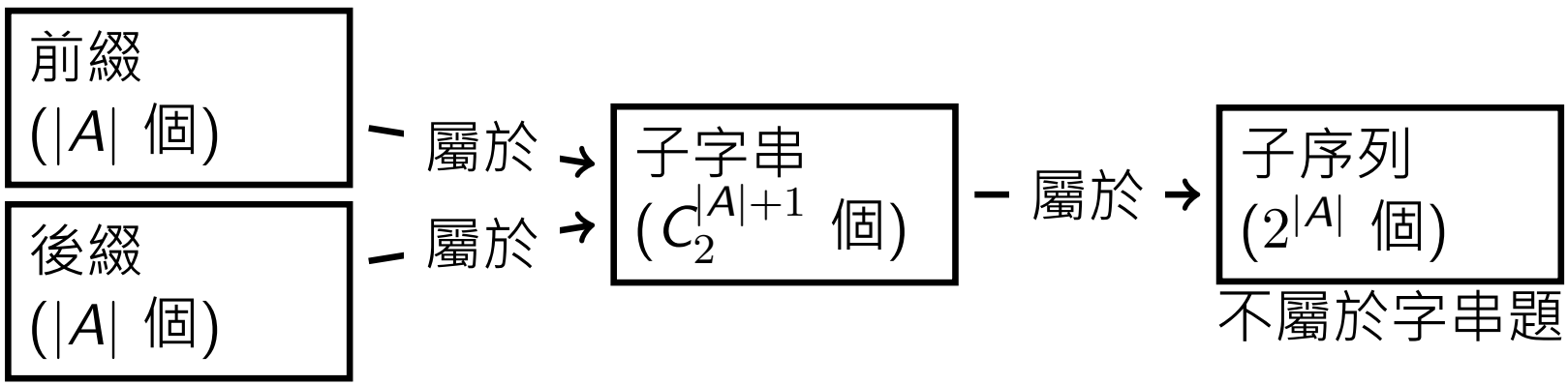
給定一個字串 $A = a_1 a_2 \cdots a_{|A|}$，設 $n = |A|$。
一個 A 的**後綴**是 A 的一個子字串 $a_k a_{k+1} a_{k+2} \cdots a_n$，其中 $1 \leq k$，記作 $S_A(k)$。並讓所有後綴的集合稱作 $\sigma(A)$。

如果 $A =$ `"hello"`，則

- $S_A(1) =$ `"hello"`
- $S_A(2) =$ `"ello"`
- $S_A(3) =$ `"llo"`
- $S_A(4) =$ `"lo"`
- $S_A(5) =$ `"o"`
- $\sigma(A) = \{S_A(1), S_A(2), S_A(3), S_A(4), S_A(5)\}$

字串演算法是什麼

由前面的定義，我們可以看出下圖的關係。



字串演算法是在探討的是字串和其**子字串**之間相關的性質。

字串題分類器

Theorem (字串題分類器)

1. *題目和有序序列無關* $\Rightarrow 97\%$ *非字串題。*
2. *題目的要求跟非連續子序列有關* $\Rightarrow 87\%$ *非字串題。*
3. *題目* $\mathcal{O}\left(n^2\right)$ *窮舉子字串會過* $\Rightarrow$ *不需要使用演算法加速。*
4. *題目要求子字串符合某種模式，例如：匹配、迴文、重複，就很有可能是我們所要討論的字串題。*

memo: 講教這個的原因。

習題真或假

A 是字串，對所有 $1 \leq i \leq j \leq |A|$，$A[i,j] \in \bigcup_{k=1}^{|A|} \sigma(P_A(k))$ 恆真？

習題真或假

A 是字串，對所有 $1 \leq i \leq j \leq |A|$，$A[i,j] \in \bigcup_{k=1}^{|A|} \sigma(P_A(k))$ 恆真？

真，$P_A(j) = a_1 a_2 \cdots a_j$
$\Rightarrow S_{P_A(j)}(i) = a_i a_{i+1} \cdots a_j = A[i,j]$
$\Rightarrow A[i,j] \in \sigma(P_A(j))$

習題真或假

A 是字串，對所有 $1 \leq i \leq j \leq |A|$，$S_{P_A(j)}(i) = P_{S_A(i)}(j)$ 恆真？

習題真或假

A 是字串，對所有 $1 \le i \le j \le |A|$，$S_{P_A(j)}(i) = P_{S_A(i)}(j)$ 恆真？

假，代入 $A = \texttt{"hellohello"}, i = 3, j = 7$
$\Rightarrow S_{P_A(7)}(3) = S_{\texttt{"hellohe"}}(3) = \texttt{"llohe"}$
$\neq P_{S_A(3)}(7) = S_{\texttt{"llohello"}}(7) = \texttt{"llohell"}$

正確的敘述為 $S_{P_A(j)}(i) = P_{S_A(i)}(j - i + 1)$。

滾動哈希

Theorem (Rabin-Karp rolling hash function)

給定相異質數 p, q，令

$$\begin{aligned} f(A) &= a_1 p^{n-1} + a_2 p^{n-2} + \cdots + a_{n-1} p + a_n \pmod{q} \\ &= \sum_{i=1}^{n} a_i p^{n-i} \pmod{q} \end{aligned} \tag{1}$$

假如 $p = 100, q = 10^9 + 7$,

$f(\texttt{"abbab"})$

$= 97(100)^4 + 98(100)^3 + 98(100)^2 + 97(100)^1 + 98(100)^0$

$= 9798989798 = 798989735 \pmod{10^9 + 7}$

變化視窗

Theorem (哈盾加一之術)

假設 f 由(1)給出，則

$$f(P_A(i)) \equiv p \times f(P_A(i-1)) + a_i \pmod{q} \tag{2}$$

Theorem (哈盾相消之術)

假設 f 由(1)給出，則

$$f(A[i,j]) \equiv f(P_A(j)) - p^{j-i+1} \times f(P_A(i-1)) \pmod{q} \tag{3}$$

哈盾加一之術

例如 $A = \texttt{"abbababbbab"}, p = 100, q = 10^9 + 7$

i	0	1	2	3
$f(P_A(i))$	0	97	9798	979898
i	4	5	6	7
$f(P_A(i))$	97989897	798989735	898973044	897303875
i	8	9	10	
$f(P_A(i))$	730386975	38697086	869708677	

加一之術可以讓我們 $\mathcal{O}(|A|)$ **預處理**出所有 $f(P_A(i))$ 的值。

哈盾相消之術

例如 $A = \texttt{"abbababbbab"}, p = 100, q = 10^9 + 7$
$f(A[6, 10]) = f(P_A(10)) - p^{10-6+1} f(P_A(5))$
$= 869708677 - (100)^5 798989735 = 798989735 = f(P_A(5))$

相消之術可以讓我們 $\mathcal{O}(1)$ **查詢**出所有 $f(A[i, j])$ 的值。

註：$f(P_A(i))$ 和 p^i 都要預處理才能 $\mathcal{O}(1)$ 回答。

哈希賭局

給兩個字串 A 和 B，且 p, q 都是質數

- 如果 $A = B$ 則對所有 (p, q) 都滿足 $f(A) = f(B)$。
- 如果 $A \neq B$ 則大約對 $\frac{1}{q}$ 的 (p, q) 滿足 $f(A) = f(B)$、$\frac{q-1}{q}$ 的 (p, q) 滿足 $f(A) \neq f(B)$。

哈希賭局

給兩個字串 A 和 B，且 p, q 都是質數

- 如果 $A = B$ 則對所有 (p, q) 都滿足 $f(A) = f(B)$。
- 如果 $A \neq B$ 則大約對 $\frac{1}{q}$ 的 (p, q) 滿足 $f(A) = f(B)$、$\frac{q-1}{q}$ 的 (p, q) 滿足 $f(A) \neq f(B)$。

Theorem (哈希賭局)

若 $f(A) = f(B)$，我們就賭 $A = B$；若 $f(A) \neq f(B)$，則一定 $A \neq B$。

選擇夠多對 (p, q)，可以讓 $A \neq B$ 且 $f(A) = f(B)$ 的機率趨近於 $\prod_{i=1}^{k} \frac{1}{q_i}$，競賽上通常兩對 (p, q) 足以。

Longest Common Substring

例題 (Longest Common Substring)

記 $sub(A)$ 為 A 所有子字串的集合。給兩個字串 s_0, s_1 找 $sub(s_0) \cap sub(s_1)$ 中最長的字串之長度。
$|s_0|, |s_1| \leq 10^5$ ，$\Sigma =$ *小寫字母*。

範例一：

$s_0 =$ `"banana"`

$s_1 =$ `"cianaic"`

$sub(s_0) \cap sub(s_1) =$ {`"a"`, `"b"`, `"ab"`, `"ba"`, `"aba"`}

輸出 $= |$`"aba"`$| = 3$

範例二：

$s_0 =$ `"abcdefghi"`

$s_1 =$ `"efgabcdhi"`

$sub(s_0) \cap sub(s_1) =$ $sub($`"abcd"`$) \cup sub($`"efg"`$) \cup sub($`"hi"`$)$

輸出 $= |$`"abcd"`$| = 4$

Longest Common Substring

- 若 "abcd" $\in sub(s_0) \cap sub(s_1)$ 則
  $sub(\texttt{"abcd"}) \subseteq sub(s_0) \cap sub(s_1) \Rightarrow$ 所求具有二分搜性質。

Longest Common Substring

- 若 "abcd" $\in sub(s_0) \cap sub(s_1)$ 則 $sub(\text{"abcd"}) \subseteq sub(s_0) \cap sub(s_1) \Rightarrow$ 所求具有二分搜性質。
- 要判斷有沒有長度 L 的 LCS，可以先分別求出兩邊長度恰 L 的子字串的哈希值，假設是 set_0, set_1，利用**哈希賭博**，若 $set_0 \cap set_1 \neq \phi$，則極有可能存在兩個一樣的子字串；若 $set_0 \cap set_1 = \phi$，則一定不存在兩個一樣的子字串。

Longest Common Substring

- 若 "abcd" $\in sub(s_0) \cap sub(s_1)$ 則 $sub(\text{"abcd"}) \subseteq sub(s_0) \cap sub(s_1) \Rightarrow$ 所求具有二分搜性質。
- 要判斷有沒有長度 L 的 LCS，可以先分別求出兩邊長度恰 L 的子字串的哈希值，假設是 set_0, set_1，利用**哈希賭博**，若 $set_0 \cap set_1 \neq \phi$，則極有可能存在兩個一樣的子字串；若 $set_0 \cap set_1 = \phi$，則一定不存在兩個一樣的子字串。
- 找兩個集合的交集可以用資料結構或排序雙指針做到 $\mathcal{O}(n)$ 或 $\mathcal{O}(n \log n)$。

Longest Common Substring

- 若 "abcd" $\in sub(s_0) \cap sub(s_1)$ 則 $sub(\text{"abcd"}) \subseteq sub(s_0) \cap sub(s_1) \Rightarrow$ 所求具有二分搜性質。
- 要判斷有沒有長度 L 的 LCS，可以先分別求出兩邊長度恰 L 的子字串的哈希值，假設是 set_0, set_1，利用**哈希賭博**，若 $set_0 \cap set_1 \neq \phi$，則極有可能存在兩個一樣的子字串；若 $set_0 \cap set_1 = \phi$，則一定不存在兩個一樣的子字串。
- 找兩個集合的交集可以用資料結構或排序雙指針做到 $\mathcal{O}(n)$ 或 $\mathcal{O}(n \log n)$。
- 二分搜總共要判 $\mathcal{O}(\log n)$ 次，每次 $\mathcal{O}(n)$，總複雜度 $\mathcal{O}(n \log n)$。
- 每次判斷時相當於做了 $\mathcal{O}(|set_0||set_1|) = \mathcal{O}\left(n^2\right)$ 次比較，所以用兩對 (p, q) 比較保險。

Longest Common Substring

假設 $s_0 = \texttt{"banana"}, s_1 = \texttt{"cianaic"}$

判斷 s_0, s_1 有沒有長度 4 的 LCS？
$set_0 = \{f(\texttt{"bana"}), f(\texttt{"anan"}), f(\texttt{"nana"})\}$
$set_1 = \{f(\texttt{"cian"}), f(\texttt{"iana"}), f(\texttt{"anai"}), f(\texttt{"naic"})\}$
$set_0 \cap set_1 = \phi \Rightarrow$ 沒有長度 4 的 LCS。

Longest Common Substring

假設 $s_0 = \texttt{"banana"}, s_1 = \texttt{"cianaic"}$

判斷 s_0, s_1 有沒有長度 4 的 LCS？
$set_0 = \{f(\texttt{"bana"}), f(\texttt{"anan"}), f(\texttt{"nana"})\}$
$set_1 = \{f(\texttt{"cian"}), f(\texttt{"iana"}), f(\texttt{"anai"}), f(\texttt{"naic"})\}$
$set_0 \cap set_1 = \phi \Rightarrow$ 沒有長度 4 的 LCS。

判斷 s_0, s_1 有沒有長度 3 的 LCS？
$set_0 = \{f(\texttt{"ban"}), f(\texttt{"ana"}), f(\texttt{"nan"}), f(\texttt{"ana"})\}$
$set_1 = \{f(\texttt{"cia"}), f(\texttt{"ian"}), f(\texttt{"ana"}), f(\texttt{"nai"}), f(\texttt{"aic"})\}$
$set_0 \cap set_1 = \{f(\texttt{"ana"})\} \Rightarrow$ 極有可能有長度 3 的 LCS。

哈希應用

Theorem (最長共同前綴)

對兩字串 A, B，定義他倆的 LCP(最長共同前綴) 為：

$$LCP(A, B) = \max_{A[1,L]=B[1,L]} (L)$$

哈希應用

Theorem (最長共同前綴)

對兩字串 A, B，定義他倆的 LCP(最長共同前綴) 為：

$$LCP(A, B) = \max_{A[1,L]=B[1,L]} (L)$$

$A[1, x] = B[1, x]$ 代表 $a_1 = b_1, a_2 = b_2, \cdots, a_{x-1} = b_{x-1}, a_x = b_x$

哈希應用

Theorem (最長共同前綴)

對兩字串 A, B，定義他倆的 LCP(最長共同前綴) 為：

$$LCP(A, B) = \max_{A[1,L]=B[1,L]} (L)$$

$A[1, x] = B[1, x]$ 代表 $a_1 = b_1, a_2 = b_2, \cdots, a_{x-1} = b_{x-1}, a_x = b_x$

$\Rightarrow$ 對所有 $1 \leq y < x$ 都滿足 $A[1, y] = B[1, y]$

哈希應用

Theorem (最長共同前綴)

對兩字串 A, B，定義他倆的 LCP(最長共同前綴) 為：

$$LCP(A, B) = \max_{A[1,L]=B[1,L]} (L)$$

$A[1, x] = B[1, x]$ 代表 $a_1 = b_1, a_2 = b_2, \cdots, a_{x-1} = b_{x-1}, a_x = b_x$

$\Rightarrow$ 對所有 $1 \leq y < x$ 都滿足 $A[1, y] = B[1, y]$

$\Rightarrow$ $LCP(A, B)$ 是有二分搜性質的，因此我們只要二分搜 L，每次使用哈希之術判斷 $f(A[1, L]) = f(B[1, L])$ 是否為真。

哈希應用

Theorem (子字串的字典序比大小)

對字串 $A = a_1 a_2 \cdots a_n$，設 $L = LCP(S_A(i), S_A(j))$

$$S_A(i) < S_A(j) \Leftrightarrow a_{i+L} < a_{j+L}$$

$A =$ "abaxabao" 我們想比較 $S_A(1)$ 和 $S_A(5)$ 的字典序，
$L = LCP(S_A(1), S_A(5)) = |\text{"aba"}| = 3$

$a_{1+3} = x, a_{5+3} = o \Rightarrow a_{1+3} > a_{5+3} \Leftrightarrow S_A(1) > S_A(5)$

哈希應用

Theorem (哈希求最長奇數長度迴文)

給字串 A，求 A 的最長奇數長度迴文子字串的長度。

1. *複製 B 使 $b_i = a_{|A|-i+1}$，也就是反轉 A*
2. *預處理 A 和 B 哈希值*
3. *枚舉中心點位置 x，找出從 x 往左和往右的 LCP 長度，得到以 x 為中心點最長的迴文長度*
4. *對所有中心點的最長迴文取最大值*

哈希應用

$A =$ "xyzabacabaoo"
$B =$ "ooabacabazyx"

哈希應用

$A =$ "xyzabacabaoo"
$B =$ "ooabacabazyx"

$x = 7$ 往右往左 $L = LCP(S_A(7), S_B(12 - 7 + 1)) = 4$
以 7 為中心點最長的奇數迴文 $=$ "abacaba"

哈希應用

$A =$ "xyzabacabaoo"
$B =$ "ooabacabazyx"

$x = 7$ 往右往左 $L = LCP(S_A(7), S_B(12 - 7 + 1)) = 4$
以 7 為中心點最長的奇數迴文 = "abacaba"

$x = 5$ 往右往左 $L = LCP(S_A(5), S_B(12 - 5 + 1)) = 2$
以 7 為中心點最長的奇數迴文 = "aba"

哈希應用

$A =$ "xyzabacabaoo"
$B =$ "ooabacabazyx"

$x = 7$ 往右往左 $L = LCP(S_A(7), S_B(12 - 7 + 1)) = 4$
以 7 為中心點最長的奇數迴文 = "abacaba"

$x = 5$ 往右往左 $L = LCP(S_A(5), S_B(12 - 5 + 1)) = 2$
以 7 為中心點最長的奇數迴文 = "aba"

- 偶數作法雷同。
- 此做法要大常數的 $\mathcal{O}(n \log n)$，最佳方法可以小常數的 $\mathcal{O}(n)$，你可能會被卡！

哈希融合之術

Theorem (哈希融合之術)

對兩字串 A, B，$f(AB) = p^{|B|} f(A) + f(B) \mod q$

$A =$ "abbab", $p = 100, q = 10^9 + 7$

$f($"abbababbab"$)$

$= f(AA)$

$= p^5 f(A) + f(A)$

$= (100)^5 798989735 + 798989735$

$= 869708677 \pmod q$

- 哈希可以套用到線段結構上做到動態修改
- $f(AAAA \cdots AAA)$ 可以快速冪求得

Pattern String

例題 (Pattern String)

定義字串 $A = a_1 a_2 \cdots a_n$ 做一次旋轉為 $r(A) = a_2 \cdots a_n a_1$。定義 $rot(A) = \{A, r(A), r(r(A)), \cdots\}$。定義 $A \sim B \Leftrightarrow rot(A) = rot(B)$。

$$\texttt{"abcde"} \sim \texttt{"bcdea"} \sim \texttt{"cdeab"} \sim \texttt{"deabc"} \sim \texttt{"eabcd"}$$

`"ruiruiruiruiruirui"` 可以壓縮成 `"ru(iru)5i"` 或 `"(rui)6"`。此題不允許兩層以上的壓縮，例如：`"((rui)2)3"` 就是不合法的壓縮。每次給兩個壓縮後的字串 S_0 和 S_1，問你 $rot(S_1)$ 中有沒有 S_0 的前綴。壓縮後的字串長不超過 11000，括號後的數字不超過 2×10^8，壓縮前的字串 $\Sigma =$ 小寫字母，括號內的字串長度不超過 10。

Pattern String

範例一：
$S_0 =$ "z(rz)3r(rui)2cumt"
$S_1 =$ "(rz)4"
$r(S_1) =$ "zrzrzrzr" $= P_{S_0}(8)$
輸出 true

範例二：
$S_0 =$ "z(rz)3r(rui)2cumt"
$S_1 =$ "zrzrrui"
$P_{S_0}(8) \notin rot(S_1)$
輸出 false

Pattern String

Theorem (壓縮字串的最小循環性質)

如果字串形如 $A(B)_kC$ *則*

$$\forall i \in [0, k],\ (B)_iCA(B)_{k-i} \geq \min((B)_kCA, CA(B)_k) \tag{4}$$

1. 假設 $BCA < CAB \Rightarrow (B)_kCA =$

   $$(B)_{k-1}BCA < (B)_{k-1}CAB < \cdots < BCA(B)_{k-1} < CA(B)_k$$

2. 假設 $BCA = CAB \Rightarrow (B)_kCA =$

   $$(B)_{k-1}BCA = (B)_{k-1}CAB = \cdots = BCA(B)_{k-1} = CA(B)_k$$

3. 假設 $BCA > CAB \Rightarrow (B)_kCA =$

   $$(B)_{k-1}BCA > (B)_{k-1}CAB > \cdots > BCA(B)_{k-1} > CA(B)_k$$

Pattern String

$$\texttt{catmew} < \texttt{mewcat} \Rightarrow$$
$$\texttt{(cat)3mew} < \texttt{(cat)2mewcat} < \texttt{catmew(cat)2} < \texttt{mew(cat)3}$$

Pattern String

$$\texttt{catmew} < \texttt{mewcat} \Rightarrow$$
$$\texttt{(cat)3mew} < \texttt{(cat)2mewcat} < \texttt{catmew(cat)2} < \texttt{mew(cat)3}$$

$$\texttt{catcat} = \texttt{catcat} \Rightarrow$$
$$\texttt{(cat)3cat} = \texttt{(cat)2catcat} = \texttt{catcat(cat)2} = \texttt{cat(cat)3}$$

Pattern String

$$\texttt{catmew} < \texttt{mewcat} \Rightarrow$$
$$\texttt{(cat)3mew} < \texttt{(cat)2mewcat} < \texttt{catmew(cat)2} < \texttt{mew(cat)3}$$

$$\texttt{catcat} = \texttt{catcat} \Rightarrow$$
$$\texttt{(cat)3cat} = \texttt{(cat)2catcat} = \texttt{catcat(cat)2} = \texttt{cat(cat)3}$$

$$\texttt{catbaa} > \texttt{baacat} \Rightarrow$$
$$\texttt{(cat)3baa} > \texttt{(cat)2baacat} > \texttt{catbaa(cat)2} > \texttt{baa(cat)3}$$

Pattern String

由我們可以知道可能是最小循環的左界只有
$\mathcal{O}\left(\text{壓縮後的字串長}\right)$ 種。
例如：$A =$ `(cat)3meow(baa)2`，那只有

`(cat)3meow(baa)2`,	`at(cat)2meow(baa)2c`,	`t(cat)2meow(baa)2ca`,
`meow(baa)2(cat)3`,	`eow(baa)2(cat)3m`,	`ow(baa)2(cat)3me`,
`w(baa)2(cat)3meo`,	`(baa)2(cat)3meow`,	`aabaa(cat)3meowb`,
`abaa(cat)3meowba`		

有可能是 A 的最小循環。

Pattern String

- 把 S_0 切成跟 S_1 一樣長。

Pattern String

- 把 S_0 切成跟 S_1 一樣長。
- 利用**壓縮字串的最小循環性質**和**哈希融合之術**分別求出 S_0 和 S_1 的最小循環可能集合。

Pattern String

- 把 S_0 切成跟 S_1 一樣長。
- 利用**壓縮字串的最小循環性質**和**哈希融合之術**分別求出 S_0 和 S_1 的最小循環可能集合。
- 利用**哈希賭博**，若兩個集合有交集就當它倆的最小循環相同；若無交集則不相同。

Pattern String

- 把 S_0 切成跟 S_1 一樣長。
- 利用**壓縮字串的最小循環性質**和**哈希融合之術**分別求出 S_0 和 S_1 的最小循環可能集合。
- 利用**哈希賭博**，若兩個集合有交集就當它倆的最小循環相同；若無交集則不相同。
- 設 $n =$壓縮後的字串長 , $m = 2 \times 10^8$，求出最小循環可能集合花費 $\mathcal{O}(n \log m)$，找兩集合交集花費 $\mathcal{O}(n)$，總共 $\mathcal{O}(n \log m)$。

哈希防駭之術

Theorem (哈希防駭之術)

在程式保留 n 對 (p, q) 並使用不確定性的隨機種子 (例如：時間)，隨機選其中 m 對 (p, q) 來使用。

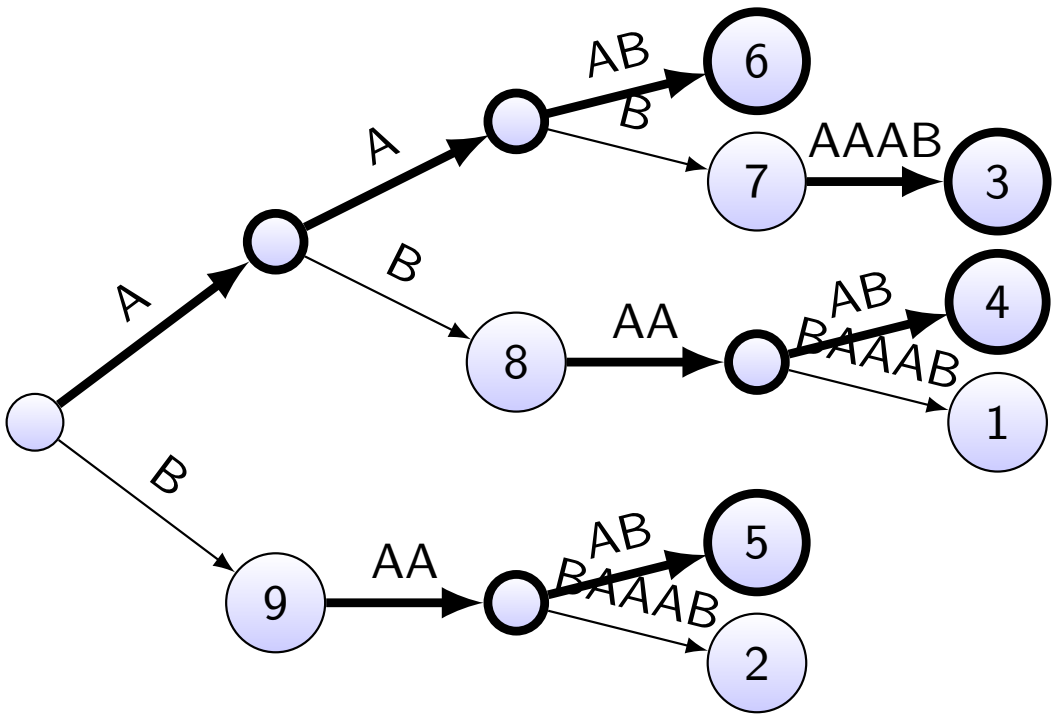
在一般測資固定的比賽，你只要用足夠大足夠多的 (p, q)，遇到碰撞頂多換一組，但是在 topcoder 或 codeforces 上，別人是可以刻意構造碰撞讓你判斷錯誤的，此時可以使用**哈希防駭之術**。

`http://codeforces.com/contest/533/room/19`

後綴樹和後綴數組

Definition (後綴樹 (suffix tree))

給一個字串 A，A 的後綴樹就是包含所有 $\sigma(A)$ 的字典樹。



後綴樹和後綴數組

Definition (後綴數組 (suffix array) 與排名數組 (rank array))

給一個字串 A，假設把 $\sigma(A)$ 排序後的序列為 $S_A(x_1), S_A(x_2), \cdots, S_A(x_{|A|})$，$A$ 的後綴數組 $sa = \{x_1, x_2, \cdots, x_{|A|}\}$，$A$ 的排名數組 $ra[x_i] = ra[sa[i]] = i$。

- 後綴數組的第 i 個數字代表第 i 小的後綴是第幾個後綴。
- 排名數組的第 i 個數字代表第 i 個後綴是第幾小的後綴。

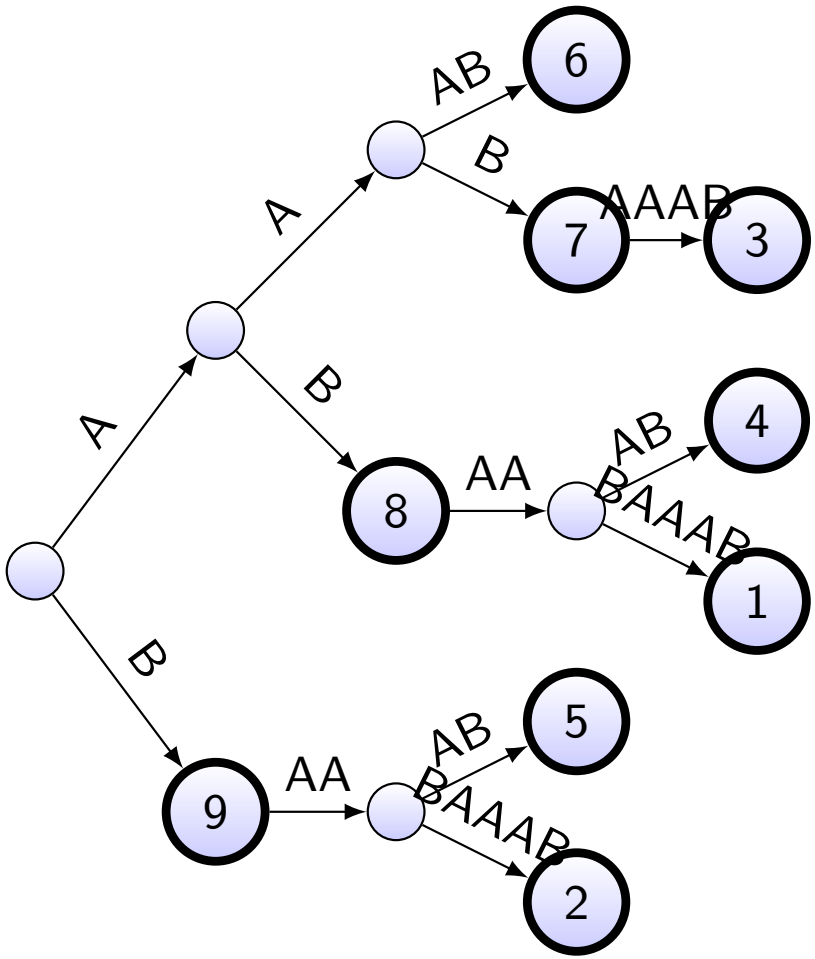
後綴樹和後綴數組

i	sa	$S_A(sa_i)$	hei	dep-hei	ra
1	6	AAAB	0	4	6
2	7	AAB	2	1	9
3	3	AABAAAB	3	4	
4	8	AB	1	1	
5	4	ABAAAB	2	4	
6	1	ABAABAAAB	4	5	
7	9	B	0	1	
8	5	BAAAB	1	4	
9	2	BAABAAAB	3	5	

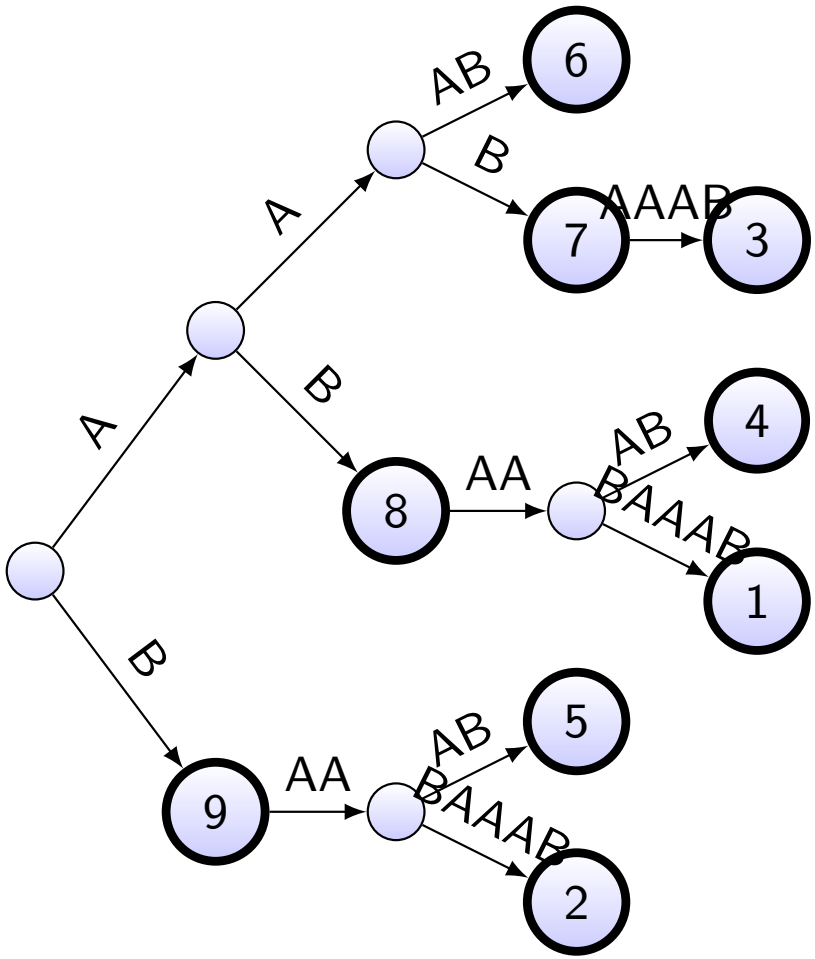
後綴樹和後綴數組

i	sa	$S_A(sa_i)$	hei	dep-hei	ra
1	6	AAAB	0	4	6
2	7	AAB	2	1	9
3	3	AABAAAB	3	4	3
4	8	AB	1	1	5
5	4	ABAAAB	2	4	8
6	1	ABAABAAAB	4	5	1
7	9	B	0	1	2
8	5	BAAAB	1	4	4
9	2	BAABAAAB	3	5	7

後綴樹和後綴數組

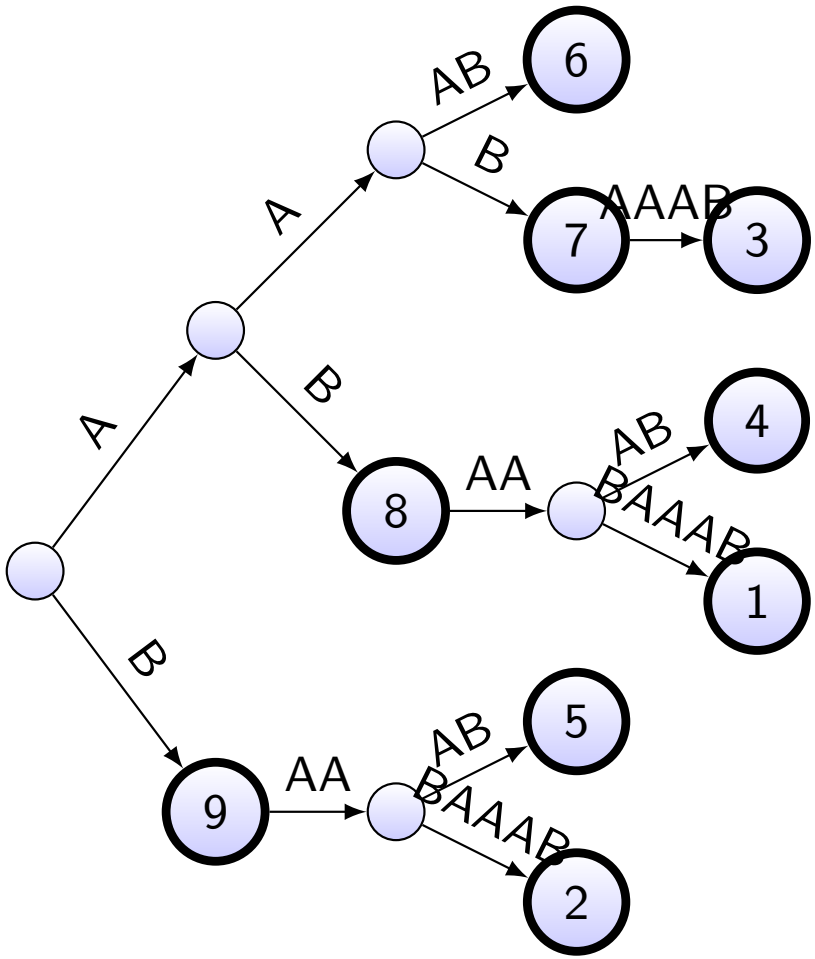


後綴樹和後綴數組



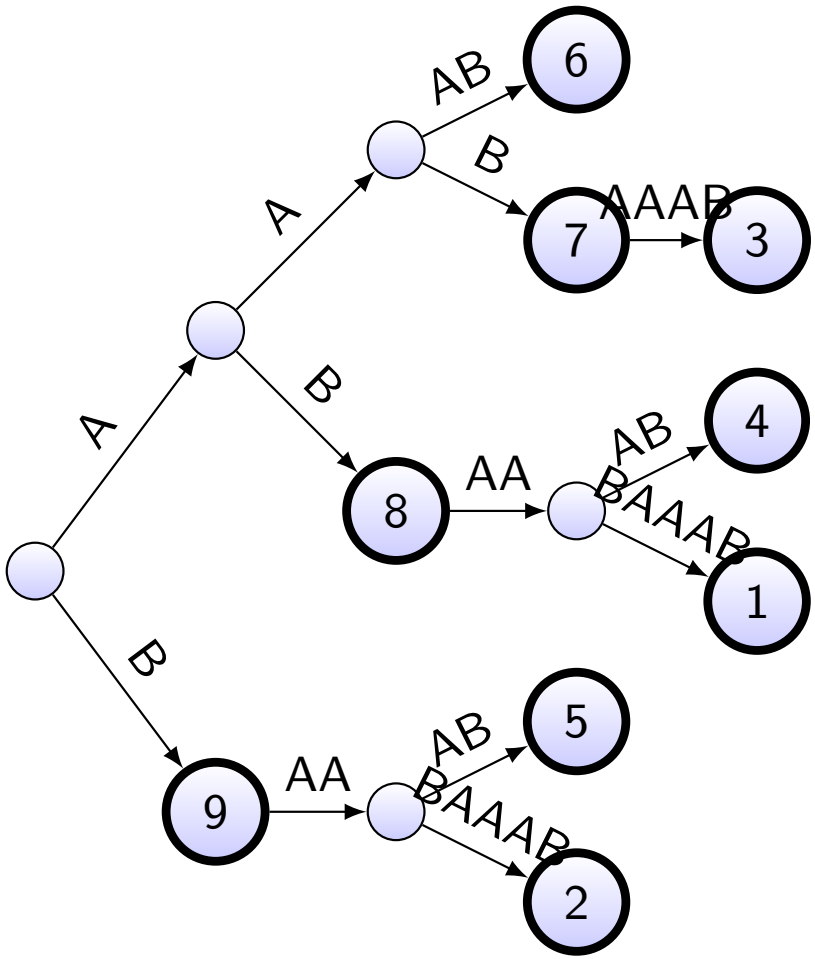
- DFS 順序：
  $6, 7, 3, 8, 4, 1, 9, 5, 2 = sa$

後綴樹和後綴數組



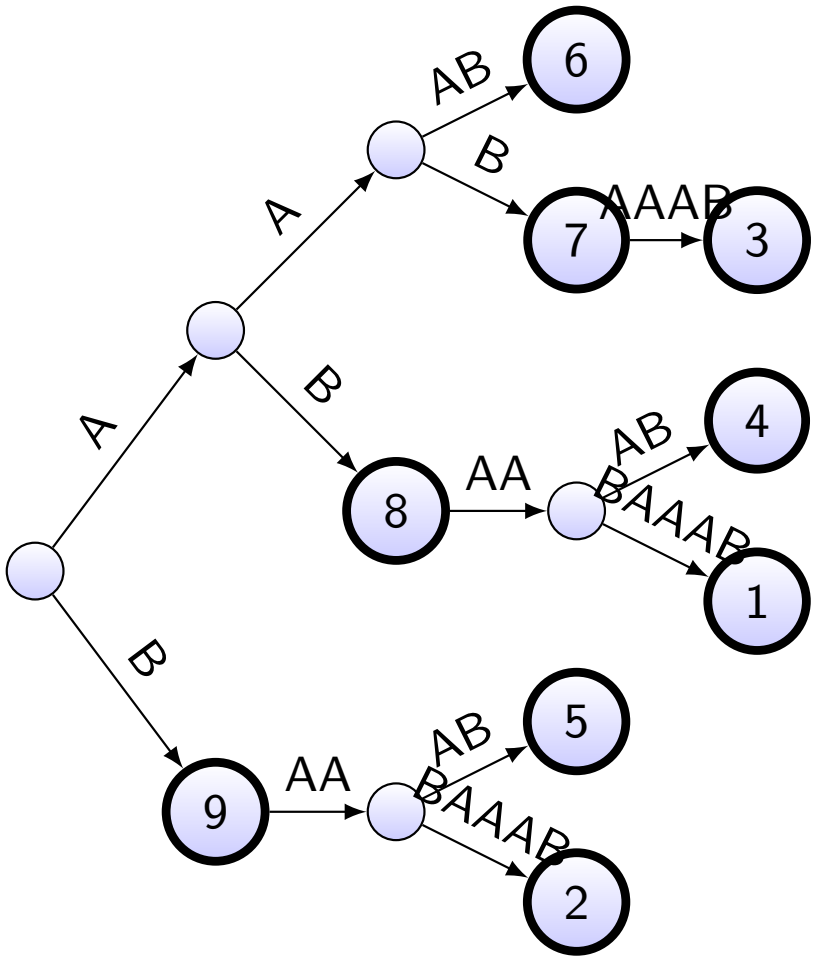
- DFS 順序：
  $6, 7, 3, 8, 4, 1, 9, 5, 2 = sa$
- $LCP(\texttt{"AAAB"}, \texttt{"AABAAAB"})$
  $= depth(LCA(6, 3)) = 2$

後綴樹和後綴數組



- DFS 順序：
  $6, 7, 3, 8, 4, 1, 9, 5, 2 = sa$
- $LCP(\texttt{"AAAB"}, \texttt{"AABAAAB"})$
  $= depth(LCA(6, 3)) = 2$
- $LCP(\texttt{"ABAAAB"}, \texttt{"AABAAAB"})$
  $= depth(LCA(4, 3)) = 1$

後綴樹和後綴數組



- DFS 順序：
  $6, 7, 3, 8, 4, 1, 9, 5, 2 = sa$
- $LCP(\texttt{"AAAB"}, \texttt{"AABAAAB"})$
  $= depth(LCA(6, 3)) = 2$
- $LCP(\texttt{"ABAAAB"}, \texttt{"AABAAAB"})$
  $= depth(LCA(4, 3)) = 1$
- $LCP(\texttt{"BAAAB"}, \texttt{"BAABAAAB"})$
  $= depth(LCA(5, 2)) = 3$

不想建樹 ~

備註

因為我們不想建後綴樹，但又想求 LCP，所以我們就用後綴數組來代替後綴樹。

後綴數組倍增法

Theorem (分治後綴比較)

對兩個 A 的等長子字串 $A[i, i+2k), A[j, j+2k)$ ，
$A[i, i+2k) < A[j, j+2k) \Leftrightarrow$

$$\begin{cases} A[i, i+k) < A[j, j+k) & \text{or} \\ A[i, i+k) = A[j, j+k) & \text{and } A[i+k, i+2k) < A[j+k, j+2k) \end{cases}$$

- $A =$ "aaaaaaab" 和 $B =$ "aaaaaac" 比較，先比前半 $A[1, 4] = B[1, 4]$ 再比後半 $A[5, 8] < B[5, 8]$ 得 $A < B$。

後綴數組倍增法

Definition (ra_k)

設 $ra_k[i]$ 代表把 $1 \le j \le |A|$ 內所有 $A[j, j+k)$ 排序後第 i 個是第幾名 (對於 $|A| - k + 1 < j$，我們把缺的部分補上小於 Σ 中所有字元的新字元 $)

設 $A =$ "caaaaaab"：

- $ra_1 = \{3, 1, 1, 1, 1, 1, 1, 2\}$ 因為 "a" < "b" < "c"。
- $ra_2 = \{4, 1, 1, 1, 1, 1, 2, 3\}$ 因為 "aa" < "ab" < "b$" < "ca"。
- $ra_4 = \{6, 1, 1, 1, 2, 3, 4, 5\}$ 因為 "aaaa" < "aaab" < "aab$" < "ab$$" < "b$$$" < "caaa"。

後綴數組倍增法

利用**分治後綴比較**，我們能得到 $A[i, i+2k) < A[j, j+2k)$
$\Leftrightarrow ra_k(i) < ra_k(j)$ or $(ra_k(i) = ra_k(j)$ and $ra_k(i+k) < ra_k(j+k))$。

這代表只要知道 ra_k，我們就能把 $A[i, i+2k)$ 排序。

而只要知道 $A[i, i+2k)$ 的排序序列，我們就能求出 ra_{2k}。

後綴數組倍增法

Theorem (後綴數組構造方法之倍增法)

1. *使用* ra_k *排序* $A[i, i+2k)$ *(得到* sa_{2k} *)*
2. *比較* $A[i, i+2k)$ *的排序序列 (* sa_{2k} *) 的相鄰元素得到每個人的排名*
3. *對應 2. 得到的排名擺到* ra_{2k} *中*
4. $k \leftarrow 2 \times k$
5. *若* $k < |A|$ *回到 1. 否則結束*

後綴數組倍增法

Theorem (後綴數組構造方法之倍增法)

1. *使用 ra_k 排序 $A[i, i+2k)$ (得到 sa_{2k})*
2. *比較 $A[i, i+2k)$ 的排序序列 (sa_{2k}) 的相鄰元素得到每個人的排名*
3. *對應 2. 得到的排名擺到 ra_{2k} 中*
4. $k \leftarrow 2 \times k$
5. *若 $k < |A|$ 回到 1. 否則結束*

分析：如果 1. 我們直接用快速排序這樣每次迭代都是 $\mathcal{O}(n \log n)$，總複雜度 $\mathcal{O}\left(n(\log n)^2\right)$。

改進：由於 ra_k 的值介於 $[1, n]$ 所以我們可以使用 radix sort 取代快速排序，每次迭代消耗 $\mathcal{O}(n)$，總複雜度 $\mathcal{O}(n \log n)$。

後綴數組倍增法

例如 $A =$ "caaaaaab", $k = 2$：

1. 已知 $ra_2 = \{4, 1, 1, 1, 1, 1, 2, 3\}$ 排序得 $sa_4 = \{2, 3, 4, 5, 6, 7, 8, 1\}$
2. 比較 $sa_4[i]$ 和 $sa_4[i+1]$ 的大小可以得到對應的名次為 $\{1, 1, 1, 2, 3, 4, 5, 6\}$
3. 名次對應回去得 $ra_4 = \{6, 1, 1, 1, 2, 3, 4, 5\}$
4. $k \leftarrow k \times 2$

溫馨提示：講義上已經提供一個 $\mathcal{O}(n \log n)$ 的模板。

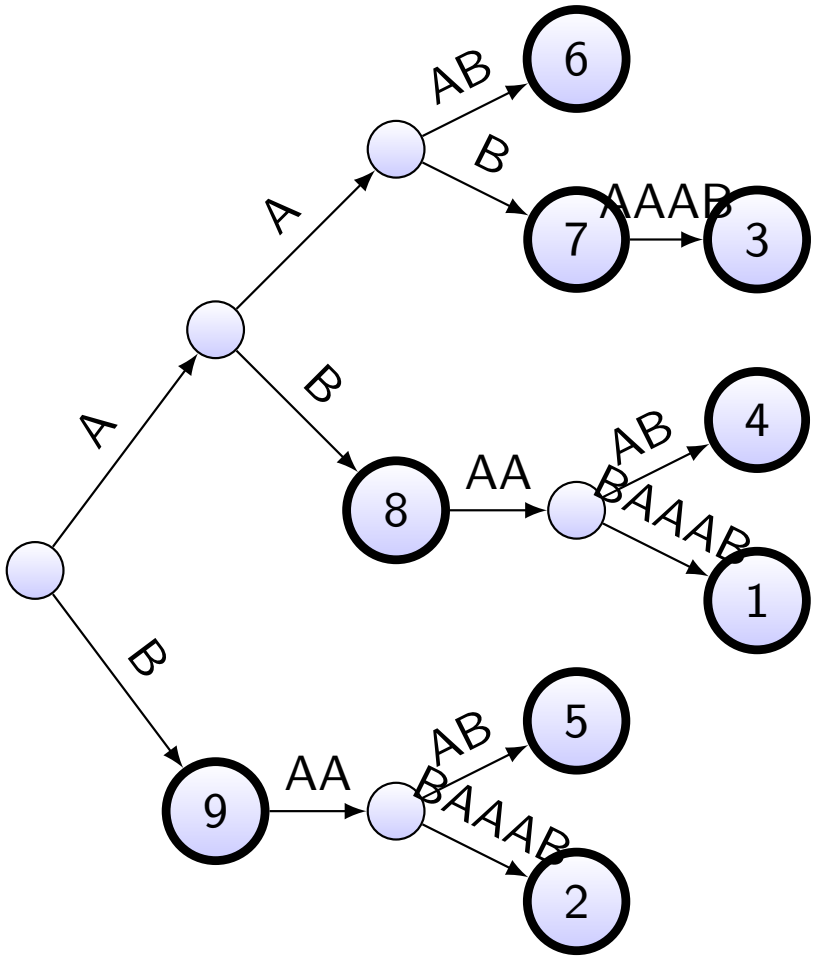
hei 數組

Definition (hei 數組 (height array or LCP array))

對一個字串 A，其後綴數組為 sa，

對所有 $1 < i \leq n$，$hei_A[i] = LCP(S_A(sa[i-1]), S_A(sa[i]))$
為了方便，定義 $hei_A[1] = 0$。

hei 數組



i	sa	$S_A(sa_i)$	hei
1	6	AAAB	0
2	7	AAB	2
3	3	AABAAAB	3
4	8	AB	1
5	4	ABAAAB	2
6	1	ABAABAAAB	4
7	9	B	0
8	5	BAAAB	1
9	2	BAABAAAB	3

求 hei 數組

Theorem (hei 數組性質)

字串 A，後綴數組 sa，排名數組 ra，對所有 $1 < ra[j] \leq n$

$$\begin{aligned} hei_A[ra[j]] &= LCP(S_A(sa[ra[j]-1]), S_A(sa[ra[j]])) \\ &= LCP(S_A(sa[ra[j]-1]), S_A(j)) \\ &\geq LCP(S_A(sa[ra[j-1]-1]), S_A(j-1)) - 1 \\ &= hei_A[ra[j-1]] - 1 \end{aligned} \tag{5}$$

- "ABAAAB" 和 "ABAABAAAB" 的 $LCP = |\text{"ABAA"}| = 4$
  "ABAABAAAB" 拿掉一字元變成 "BAABAAAB"
  代表 "BAABAAAB" 和 "ABAAAB" 拿掉一字元匹配至少會是 $|\text{"ABAA"}| - 1$。

求 hei 數組

Theorem (hei 數組構造方法)

1. 令 $k \leftarrow 0$
2. 從 $j = 1$ 到 $j = n$ 窮舉 $S_A(j)$
3. 若 $S_A(j)$ 不是排名最小則的和比自己排名小一名的人從長度 k 開始匹配
4. $hei[ra[j]] \leftarrow k$
5. $k \leftarrow \min(k-1, 0)$ 回到 2.

求 hei 數組

Theorem (hei 數組構造方法)

1. 令 $k \leftarrow 0$
2. *從* $j = 1$ *到* $j = n$ *窮舉* $S_A(j)$
3. *若* $S_A(j)$ *不是排名最小則的和比自己排名小一名的人從長度* k *開始匹配*
4. $hei[ra[j]] \leftarrow k$
5. $k \leftarrow \min(k-1, 0)$ *回到 2.*

溫馨提示：講義上已經提供一個模板。

分析：迴圈中 k 最長到 n，每次 $++j$ 時 k 至多減少一，得 k 被加的次數不超過 $2n = \mathcal{O}(n)$。

New Distinct Substrings

例題 (New Distinct Substrings)

記 $sub(A)$ 為 A 所有子字串的集合。給兩個字串 A 輸出 $sub(A)$ 中不重複元素的個數。$|A| \leq 50000$

範例一：

$A =$ "ABABA"。

$|sub(A)| = |\{$"A", "B", "AB", "BA", "ABA", "BAB", "ABAB", "BABA", "ABABA"$\}| = 9$

輸出 9。

範例二：

$A =$ "CCCCC"

$|sub(A)| = |\{$"C", "CC", "CCC", "CCCC", "CCCCC"$\}| = 5$

輸出 5。

New Distinct Substrings

仔細想一想，你會發現這題計算的就是 A 的後綴樹上有多少點。

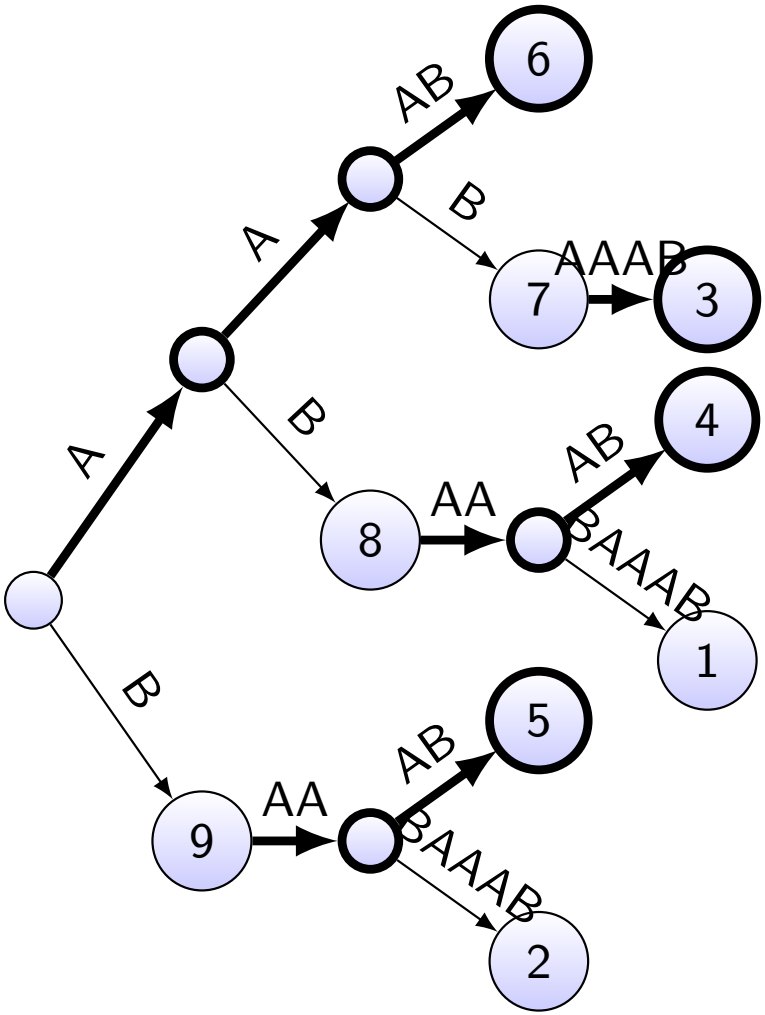
假設你有一棵樹的某個子集的 DFS 序列 $d_1, d_2, \cdots, d_n$，你想知道這些點往上到根的聯集點數有多少個，這個個數會等於

$$dep(d_1) + \sum_{i=2}^{n} [dep(d_i) - dep(LCA(d_{i-1}, d_i))]$$

，而後綴數組就是包含所有後綴樹的葉子的 DFS 序列。
代入上式，答案

$$= dep(sa_1) + \sum_{i=2}^{n} (dep(sa_i) - he[i]) = \frac{(n+1)n}{2} - \sum_{i=1}^{n} he[i]$$

New Distinct Substrings



i	sa	$S_A(sa_i)$	hei	dep -hei
1	6	AAAB	0	4
2	7	AAB	2	1
3	3	AABAAAB	3	4
4	8	AB	1	1
5	4	ABAAAB	2	4
6	1	ABAABAAAB	4	5
7	9	B	0	1
8	5	BAAAB	1	4
9	2	BAABAAAB	3	5

用後綴數組找 *LCP*

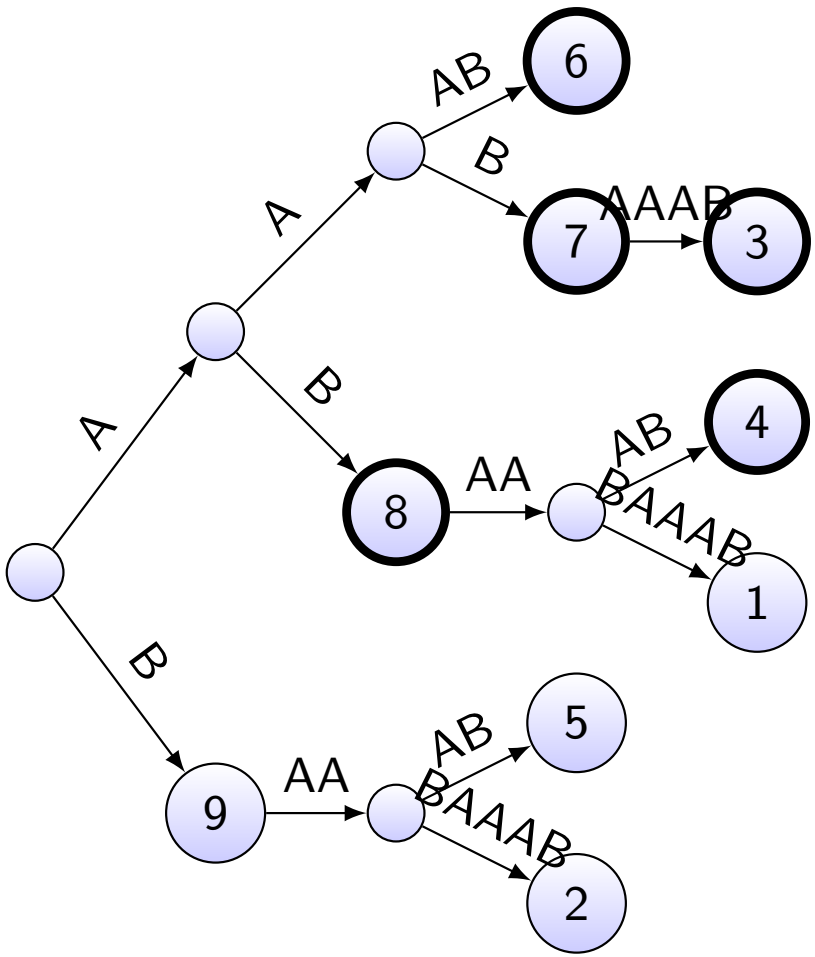
Theorem (用後綴數組找 *LCP*)

$ra[i] = ra[j]$ 瑣碎，不失一般性假設 $ra[i] < ra[j]$，

$$\begin{aligned} LCP(S_A(i), S_A(j)) &= dep(LCA(sa[ra[i]], sa[ra[j]])) \\ &= \min(hei[ra[i]+1], hei[ra[i]+2], \cdots, hei[ra[j]]) \end{aligned}$$

由來是：對任意一棵樹，一個包含所有葉子的 dfs 序列 $d_1, d_2, \cdots, d_n$，對所有 $i < j$ 都滿足 $LCA(d_i, d_j) \in \{LCA(d_i, d_{i+1}), LCA(d_{i+1}, d_{i+2}), \cdots, LCA(d_{j-1}, d_j)\}$。又 *LCA* 子樹中的點深度都比 *LCA* 深，所以深度最小值就是 *LCA* 的深度。

用後綴數組找 *LCP*



$$
\begin{aligned}
&LCP(S_A(4), S_A(6)) \\
&= dep(LCA(4,6)) = \min(\\
&dep(LCA(6,7)),\ dep(LCA(7,3)), \\
&dep(LCA(3,8)),\ dep(LCA(8,4))) \\
&= \min(he[2], he[3], he[4], he[5])
\end{aligned}
$$

用後綴數組找 *LCP*

利用**用後綴數組找** *LCP*，我們套用區間最小值問題的解法 (RMQ)，終於我們會用後綴數組找 *LCP* 了。
區間最小值問題我們假設使用 sparse table，預處理 $\mathcal{O}(n \log n)$，回答 $\mathcal{O}(1)$。

Maximum repetition substring

例題 (Maximum repetition substring)

對一個字串 A，設 $f(A) = \max_{(B)_n=A} n$。例如：
$f(\texttt{"abababab"}) = f((\texttt{"ab"})_4) = 4$、$f(\texttt{"ioi"}) = 1$、$f(\texttt{"aaa"}) = 3$。
給一個字串 C，你要輸出 f 最大的子字串，若有多組解你要輸出字典序最小的那一個。$|C| \leq 10^5$。

範例一：
$C =$ "ccababababc"
$f(\texttt{"abababab"}) = 3$
最大。
輸出 "abababab"。

範例二：
$C =$ "daabbccaa"
$f(\texttt{"aa"}) = f(\texttt{"bb"}) = f(\texttt{"cc"}) = f(\texttt{"dd"}) = 2$ 最大。
其中 "aa" 字典序最小。
輸出 "aa"。

Maximum repetition substring

第一步考慮 $i < j, L = LCP(S_C(i), S_C(j))$ 的特性：

$$L \geq j - i \Leftrightarrow C[i, j) = C[j, 2j - i) \Leftrightarrow C[i, 2j - i) = (C[i, j))_2$$

同理：

$$L \geq t(j - i) \Leftrightarrow C[i, i + (t + 1)(j - i)) = (C[i, j))_{t+1}$$

$C =$ "xxcabcabcabcabcaxx", $LCP(S_C(4), S_C(7)) = 10$

xxc | abc abc abc | abc axx

xxc abc | abc abc abc | axx

$\lfloor \frac{LCP(S_C(4), S_C(7))}{7 - 4} \rfloor = 3$ 代表後面重複 3 次，加自己 4 次。

memo：訂正 $t + 1$

Maximum repetition substring

如果只用第一步的性質，我們需要窮舉所有 i, j，會花 $\mathcal{O}\left(n^2\right)$。
第二步，考慮若 $n \geq 2, A = (B)_n \in sub(C)$，則 A 一定會包含 $c_{|B|}, c_{2|B|}, c_{3|B|}, \cdots, c_{\lfloor \frac{|C|}{|B|} \rfloor |B|}$ 的其中連續兩個。

例如：
$C =$ "xxcabcabcabcabcaxx" $=$ "xxc(abc)4axx"
$B =$ "(abc)4" 包含 c_6, c_9。

Maximum repetition substring

利用包含連續兩個的性質，我們窮舉 B (被重複的串) 的長度，再窮舉 $(i, j) = (t \times k, (t+1) \times k)$，並令 $L = LCP(S_C(i), S_C(j)) + LCP(P_C(i)^R, P_C(j)^R) - 1$，(A^R 為反轉)，$\lfloor \frac{L}{k} \rfloor$ 就是 $|B| = k$ 且包含 c_i 和 c_j 最大的重複數量。

例如：$C =$ "ccababab c"，窮舉到 $|B| = 2$ 時，會詢問

- $L_{2,4} = LCP(S_C(2), S_C(4)) + LCP(P_C(2)^R, P_C(4)^R) - 1$
  $= 0 + 0 - 1 = -1$
- $L_{4,6} = LCP(S_C(4), S_C(6)) + LCP(P_C(4)^R, P_C(6)^R) - 1$
  $= 3 + 2 - 1 = 4$
- $L_{6,8} = LCP(S_C(6), S_C(8)) + LCP(P_C(6)^R, P_C(8)^R) - 1$
  $= 1 + 4 - 1 = 4$

Maximum repetition substring

對所有 k 最大的重複數量取最大值，我們就可以得到 f 函數的最大值。

其中我們要求 $LCP(P_C(i)^R, P_C(j)^R)$，具體作法就是把字串反轉複製一份後，分別做原版的 LCP 和反轉版的 LCP。(和求迴文時雷同)

我們可以從 i, j 得到它代表的字串的起始位置和長度，找字典序最小即是找 ra 最小的起始點。

Maximum repetition substring

分析：
固定 k 後我們需要枚舉的 i, j 數為 $\mathcal{O}\left(\frac{|C|}{k}\right)$。
詢問總數 $\sum_{k=1}^{|C|} \mathcal{O}\left(\frac{|C|}{k}\right) = \mathcal{O}\left(|C| \sum_{k=1}^{|C|} \frac{1}{k}\right) = \mathcal{O}(|C| ln |C|)$，每次 LCP 用 $\mathcal{O}(1)$ 回答。預處理花費 $\mathcal{O}(|C| \log |C|)$。
總複雜度 $\mathcal{O}(|C| \log |C|)$。

如果用哈希之術來 LCP 那複雜度會變成 $\mathcal{O}\left(|C|(\log |C|)^2\right)$

AC 自動機

Definition (pre 集合和失敗指針)

對一個字串集合 X，

1. $pre(X)$ 代表字串集合 X 中所有字串的前綴所形成的集合。
2. 每個節點的失敗指針所指到的是**這個前綴的後綴中除了它自己屬於 $pre(X)$ 最長的一個**。

AC 自動機

Definition (pre 集合和失敗指針)

對一個字串集合 X，

1. $pre(X)$ 代表字串集合 X 中所有字串的前綴所形成的集合。
2. 每個節點的失敗指針所指到的是**這個前綴的後綴中除了它自己屬於 $pre(X)$ 最長的一個**。

- $L = \{\texttt{"ioi"}, \texttt{"iloli"}, \texttt{"olioi"}, \texttt{"ioioi"}\}$
- $pre(L) = \{$ `"i"`, `"o"`, `"il"`, `"io"`, `"ol"`, `"ilo"`, `"ioi"`, `"oli"`, `"ilol"`, `"ioio"`, `"olio"`, `"iloli"`, `"ioioi"`, `"olioi"`$\}$
- `"olioi"` 的失敗指針會指到 `"ioi"`，因為 `"lioi"` 不在 $pre(L)$ 中。

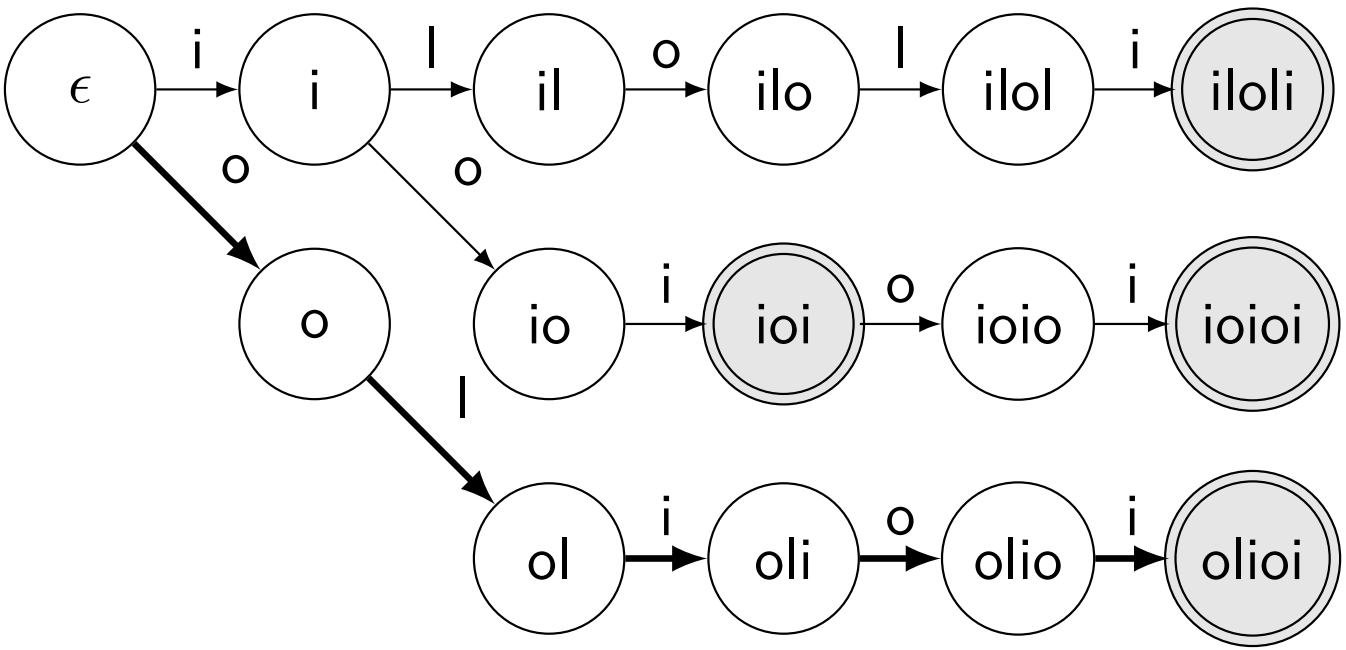
AC 自動機

Definition (AC 自動機)

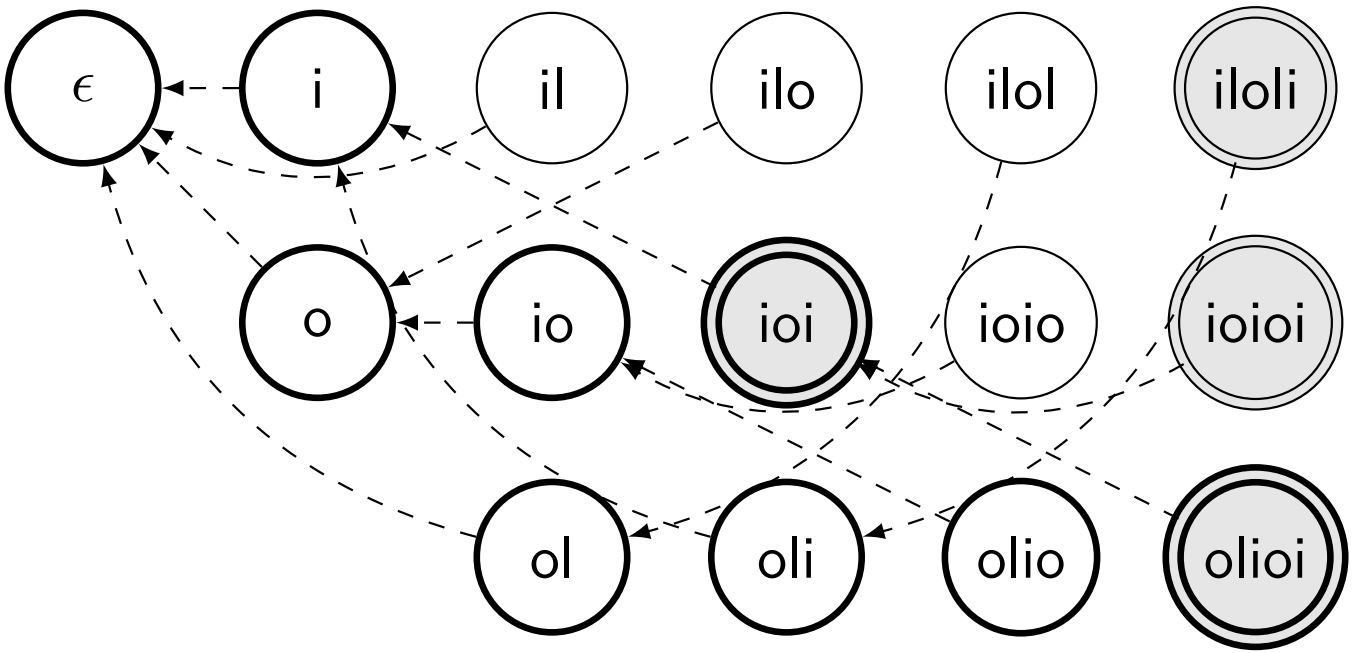
對一個字串集合 X，AC 自動機為 X 的字典樹和失敗指針樹（乾爹樹）。

- 字典樹上每個人的父親是從**後面**開始刪去，第一個在前綴集合 $pre(X)$ 裡的字串
- 失敗樹上每個人的乾爹是從**前面**開始刪去，第一個在前綴集合 $pre(X)$ 裡的字串

字典樹


ϵ
i
i
l
il
o
ilo
l
ilol
i
iloli
o
o
o
io
i
ioi
o
ioio
i
ioioi
l
ol
i
oli
o
olio
i
olioi

失敗指針樹



失敗指針樹構造法

```
1  設樹根的乾爹為空指標，並把字典樹的根放入 queue 中;
2  while queue 不空空 do
3      X ← queue 頭，並 pop queue;
4      for c ∈ Σ 且 X + c ∈ T do
5          Z ← X 的乾爹;
6          while Z 不為空指標且 Z + c ∉ T do
7              Z ← Z 的乾爹;
8          end
9          if Z 為空指標 then
10             X + c 的乾爹 ← 樹根;
11         else
12             X + c 的乾爹 ← Z + c;
13         end
14         把 X + c 放入 queue 中;
15     end
16 end
```

失敗指針樹構造法

原理：
假設某個節點的老爸是 X，他是 Xc。

演算法 6 到 8 行，問 X 的乾爹有沒有 c 的兒子，如果沒有就一直砍前綴（找乾爹的乾爹），直到某位乾爹祖先有 c 的兒子。

假設那個兒子是 Yc，因為 Y 是 X 的後綴，所以 Yc 也是 Xc 的後綴，實際上 Yc 就是 Xc 的乾爹，不過 $|Y|$ 是不是最長的需要證明。

Substring Problem

例題 (Substring Problem)

給一個大字串 A 和 N 個小字串 B_i，如果 B_i 是 A 的子字串，輸出"Y"，否則輸出"N"。$|A| \leq 10^6$, $N < 1000$, $|B_i| \leq 2000$, $\Sigma =$ 字母和數字。

範例：

$A =$ "abghABCDE"

$N = 2$

$B_1 =$ "abAB"

$B_2 =$ "ab"

對 B_1 輸出 "N"

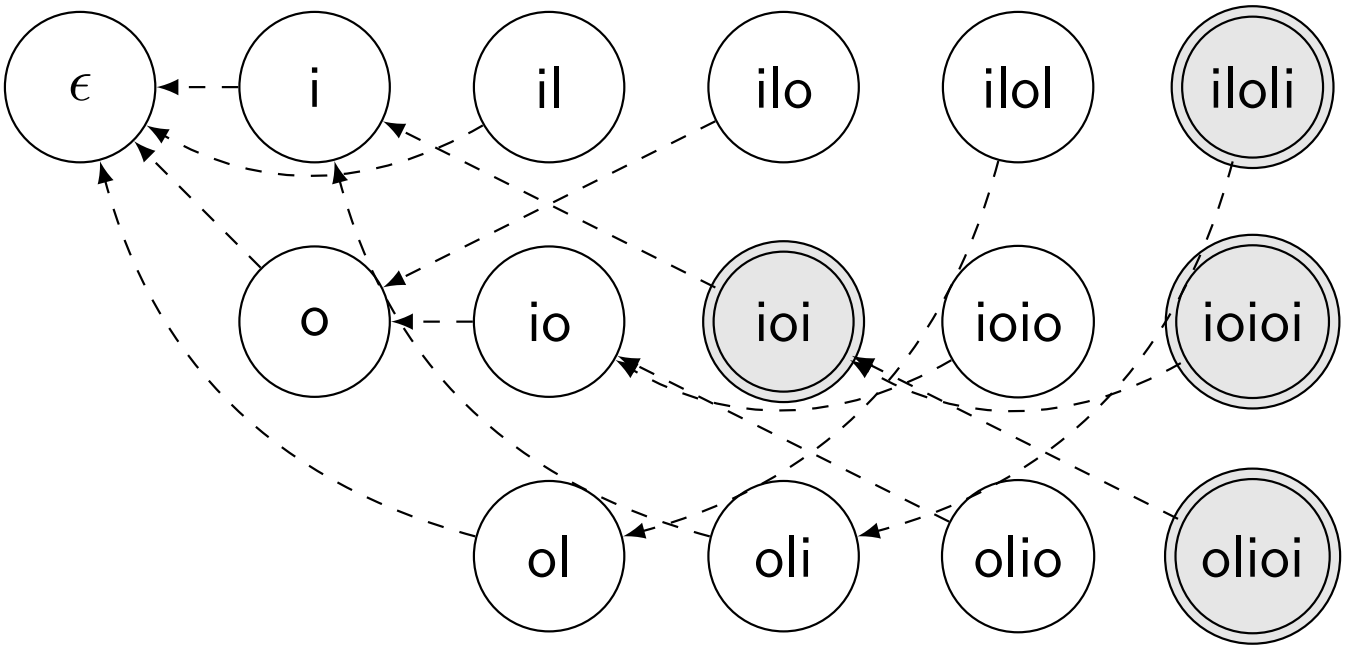
對 B_2 輸出 "Y"

AC 自動機計數方法

Data: 字典樹 T、T 的失敗指針、字串 $A = a_1 a_2 \cdots a_n$、T 的 BFS 順序 $q_1, q_2, \cdots, q_m$
Result: cnt[i] 代表點 i 在 A 中出現幾次

```
1  Z ← T 的樹根;
2  for c ← {a_1, a_2, ··· , a_n} do
3      while Z 不為空指標且 Z + c ∉ T do
4          Z ← Z 的乾爹;
5      end
6      if Z 為空指標 then
7          Z ← 樹根;
8      else
9          Z ← Z + c;
10     end
11     cnt[Z]←cnt[Z]+1;
12 end
13 for X ← {q_m, ··· , q_2, q_1} do
14     cnt[X 的乾爹]←cnt[X 的乾爹]+cnt[X];
15 end
```

AC 自動機計數方法



假設題目中 $A =$ "olioioi"，N 個小字串的集合 $= L$。

AC 自動機計數方法中

$c = a_i$	0	1	2	3	4	5	6	7
Z	ϵ	o	ol	oli	olio	olioi	ioio	ioioi

AC 自動機計數方法

- 如果你有手動跑完演算法你會發現 `"ioi"` 的計數器在 12 行以前沒被加到，因為實際上我們要把所有走過的點在失敗指針樹往上所有乾爹祖先都加一。
- 13 到 15 行就是在做等價於" 往上所有乾爹祖先都加一" 的 *dp*。15 行執行完後 `"ioi"` 的計數器就會變成 2。

DNA Sequence

例題 (DNA Sequence)

給你 m 個有病 DNA 串，求長度 n 的 DNA 串中不包含任意一個有病串作為子字串的有幾種 (對 100000 取模)。
$m \leq 10$, $n \leq 2 \times 10^9$，*每個 DNA 串長度最長* 10，*字元集* $= \{A, T, C, G\}$。

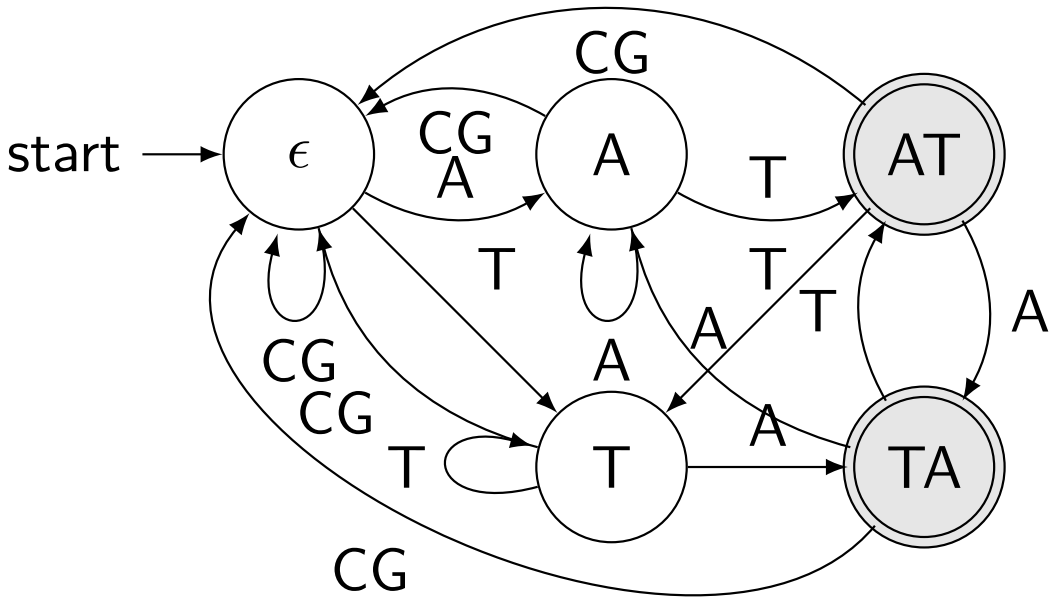
範例：
$m = 4, n = 3$，有病串為 AT,AC,AG,AA。
用邏輯推得前兩個字元不能有 A 但第三個可以，答案 $= 3 \times 3 \times 4 = 36$。

DNA Sequence

用 L 的例子，我們發現 `"xxxxio"` 和 `"abcdio"` 對 $pre(L)$ 來說其實是一樣的。
因為他倆在 **AC 自動機計數方法**的迴圈中 Z 都是 `"io"`，也就是說他倆後面接 Σ 中任一字元之後的 Z 都一樣。
這樣的意義是，我們把世界上所有字串依照 $pre(L)$ 分類了。

DNA Sequence

我們可以預先建出每一類多加一個字元會變成哪一類，就會形成一張圖。



{"AT", "TA"} 形成的圖。

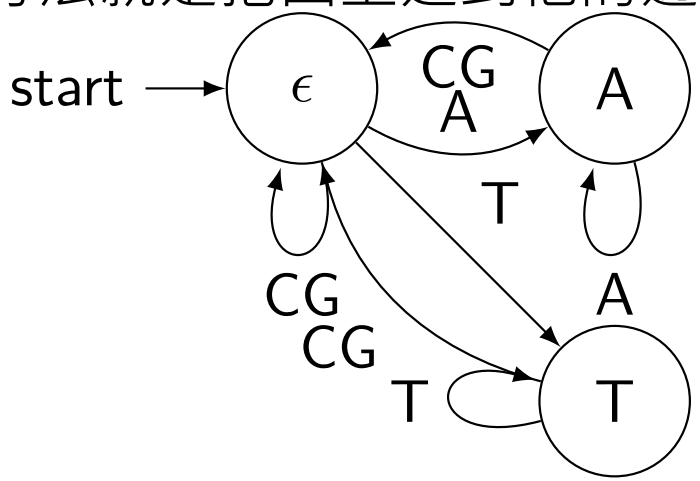
DNA Sequence

- 每個種類的長度恰 1 的字串個數會等於 ϵ 走 1 步可以到達各點的方法數
- 每個種類的長度恰 2 的字串個數會等於 ϵ 走 2 步可以到達各點的方法數
- $\cdots$
- 每個種類的長度恰 n 的字串個數會等於 ϵ 走 n 步可以到達各點的方法數

圖上走 n 步到達每個點的方法數可以 $\mathcal{O}\left(n(V+E)\right)$，或者透過矩陣乘法做到 $\mathcal{O}\left(V^3 \log n\right)$。

DNA Sequence

現在要處理的是怎麼透過 AC 自動機的分類排除有病串，可以發現如果有個種類就是有病串那我們就要把經過這個點的方法去除，或是，那個種類的**直系乾爹們中存在有病串**也要把經過他的方法去除。去除的方法就是把圖上連到他的邊消去。



去除有病串 {"AT", "TA"} 後形成的圖。

結語

- 希望大家對字串都有進一步的認識。
- 根據我本人參賽經驗字串題出現在比賽的機率不高。
- 多寫動態規劃、數學、樹、計算幾何、組合賽局、捲積啊，CP 值較高。

The End