Lecture 1
進階圖論 | Advanced Graph Theory

1.1 什麼是字串？

字串顧名思義，就是一個字元的串列。比如 “Wrong answer at test case #3”，“你今天穿的眞是夠勁” 之類的。字串是人類傳遞資訊的單位，因此也衍生出了許許多多的問題，當然也成爲競賽領域中相當重要的一門！因此，本章將帶領大家領略字串的許多基本精神，替各位打下更上層樓的基礎。

首先我們定義什麼是字串。

定義： 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

$$A = a_0 a_1 \cdots a_{n-1}$$

其中 $a_i \in \Sigma$。我們把 n 叫作字串的長度。

舉個例子，如果 $\Sigma = \{\texttt{"A"}, \texttt{"C"}, \texttt{"G"}, \texttt{"T"}\}$，$A$ 可能表示你的 DNA 序列，或者如果 $\Sigma = \{\texttt{"0"}, \texttt{"1"}, \cdots \texttt{"9"}\}$，那麼 A 可能代表客戶的電話號碼。這時候就會有許多有趣的問題了，比如:

- 假設另一段 DNA 字串 B 代表一個可能的致癌基因，如何快速的判斷一個人是否帶有這種基因?
- 現在有幾種不同的生物的 DNA 序列，如何判斷這些字串間的相似成度，以了解演化順序?

對於這些問題，數據的大小可以是非常大的，我們勢必會需要一些好的演算法來解決它們！不過在這之前我們必需先說清楚一些定義。

定義： 給定一個字串 $A = a_0a_1 \cdots a_{n-1}$

- 一個子字串是其連續的一段 $a_ia_{i+1}a_{i+2} \cdots a_j$ 記作 $A[i, j]$。
- 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 $0 \leq q_1 < q_2 < q_3 < \cdots < q_{m-1} < q_m < n$ 。
- 一個 A 的前綴是 A 的一個子字串 $a_0a_1a_2 \cdots a_h$，其中 $0 \leq h < n$，記作 $P_A(h)$。
- 一個 A 的後綴是 A 的一個子字串 $a_ka_{k+1}a_{k+2} \cdots a_n$，其中 $0 \leq k < n$，我們特別記作 $S_A(k)$。並讓所有後綴的集合稱作 $\sigma(A)$。

舉例來說，如果 $A =$ "abcbbab"，那"bcb" 是他的子字串，"acb" 是他的子序列，而"bbab" 是他的一個後綴。

1.2 字串的儲存

通常最基本的儲存方式就是用一個陣列依序將字串的每一個字元存下來。不過當我們要同時儲存許多字串時，可能就要花點巧思了。而這邊要介紹一個可以同時儲存多個字串的資料結構—**字典樹 Trie**。

Trie 的道理非常簡單，其實就是用一棵樹來儲存字串。在這棵樹上，每個點 (除了根節點之外) 上都有一個字元，而從根節點一路走到某個節點，依序經過的字元串起來就是那個點代表的字串。最後我們再記錄哪些點是一個字串的最後一個字元即可！ 如 Figure 1.1就是一個儲存 $\{A_1 =$ "abc", $A_2 =$ "abde", $A_3 =$ "bc", $A_4 =$ "bcd"$\}$ 的 trie

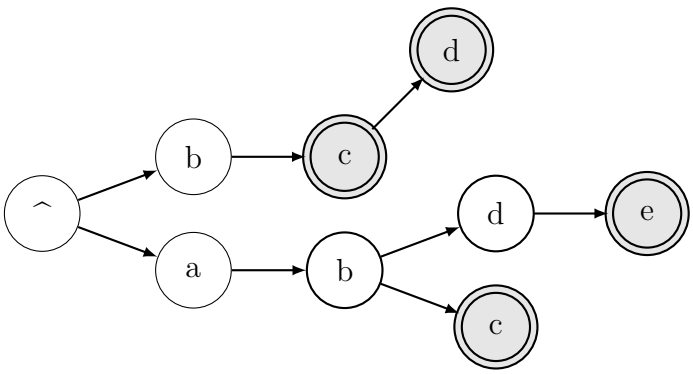

Figure 1.1: An example of trie

定義： 給定一個字典樹 Trie $T = \{V, E\}$，我們定義 $P_T(v)$ 爲從根節點走到 v 所得出的字串。

而 Trie 的基本操作也都很簡單，如要新增一個字串 A，我們就從根節點開始，依照字串 A 的第 $0, 1, 2, \cdots, n-1$ 個字元，如果此字元在當前節點的子節點中就繼續走下去，否則就新增一個節點。