Lecture 1
進階圖論 | Advanced Graph Theory

1.1 圖的種類

1.1.1 樹

一個沒有環的連通圖就是一棵樹。相信大家對樹的性質都已經很熟悉，因此這邊介紹一些進階的技巧。

樹的子樹區間

俗稱的「把樹壓平」，其實就是希望對於每一個點 v ，都定出 L_v, R_v，使得如果 u 在 v 的子樹下，有 $[L_u, R_u] \subset [L_v, R_v]$，也就是 $L_v \leq L_u \leq R_u \leq R_v$。

樹的子樹區間： 我們可以用以下方法得到 L_v, R_v，維護一個計數器 $\texttt{cnt}$ ，一開始 $\texttt{cnt} = 0$ 並從根開始 DFS。

- 當第一次到一個點 v 時，就讓 $L_v = \texttt{cnt}$。
- 當離開一個點 u 時，就讓 $R_v = \texttt{cnt}$，然後 $\texttt{cnt} \leftarrow \texttt{cnt} + 1$。

得出的 L_v, R_v 便是一個合理的子樹區間。

這可以用來解決不少區間查尋的問題

例題 1-1 樹上查詢 經典問題

給定一棵 N 個點的有根帶點權樹，接著 Q 個操作，每次操作可能是:

- `add x c`: 把以 x 爲根的子樹的所有節點權重同加上 c
- `change x c`: 把以 x 爲根的子樹的所有節點權重變成 c
- `query x`: 詢問以 x 爲根的子樹的權重最大值。

$1 \leq N, Q \leq 100000$

用上述的方法將樹做子樹區間，原本的題目就變成是在區間上做一些查詢或是修改，用線段樹等等的資料結構處理即可。

樹分治

就是在樹上使用分治法，對於一棵樹，每次選擇 v 一個點拔掉，將原來的樹分成很多個子樹，並對這些子樹遞迴求解。最後再將這些子樹合併回去。複雜度爲

$$T(n) = \sum T(m_i) + f(n) \tag{1.1}$$

其中 m_i 表示各個子樹的大小，$\sum m_i = n - 1$，而 $f(n)$ 是合併所需的時間。而這個複雜度跟 $\max m_i$ 非常相關，
如果 $\max m_i \leq n/2$ ，那通常就有不錯的複雜度，比如 $f(n) = n$ 則 $T(n) = n \log n$。因此 v 的選擇就很重要了，要滿足拔掉 v 後剩下的子樹都不會太大。事實上有以下定理。

定理 所有樹都有**重心**。一個點是重心表示拔掉他之後所有的子樹的大小都不超過原來的樹大小的一半。

Proof. 定義 $c(v)$ 爲 v 到所有其他點的距離的合。則使 $c(u)$ 最小的那個點一定是重心。否則假設本來 (u, v) 相鄰，且拔掉 u 後 v 所屬的子樹的大小大於一半，比較 $c(u), c(v)$ 可以發現有超過一半的點到 u 的距離比到 v 近 1，而不超過一半的點距離遠了 1，
因此 $c(v) < c(u)$ ⇒⇐。 □

例題 1-2 Tree POJ 1741

給一棵樹,樹的邊有權重表示距離,求距離不超過 k 的點對數。($\mathcal{O}\left(n \log^{\mathcal{O}(1)} n\right)$)

用樹分治的想法，我們就只需要考慮如何在線性時間合併子樹即可。對於每個子樹我們可以先做一次 BFS 找出每個點到當前重心 v 的距離，而合併兩個子樹 T_1, T_2 其實就是在問對於所有 $u_1 \in T_1$，有多少 $u_2 \in T_2$ 滿足 $d(u_2, v) \leq k - d(u_1, v)$。這個可用一些如線段樹等的資料結構在 $\mathcal{O}(n \log n)$ 時間完成 (事實上花點巧思，可以用雙指針在線性時間內做到)。總時間複雜度就是 $\mathcal{O}\left(n \log^2 n\right)$。

啓發式合併

樹分治因爲對最大子樹的大小有限制，所以必需額外進行找重心的動作，有點麻煩。不過再某些情況下可以用啓發式合併的方法簡化。

更詳細的說，如果合併兩個子樹 T_1, T_2 的複雜度「幾乎」只和一者的大小有關，如 $\mathcal{O}(T_2), \mathcal{O}(T_2 \log T_1)$ 等等，那麼可以證明每次合併子樹時，找最大的子樹 T_1，把其他的子樹合併進來，就會有不錯的複雜度。

定理 假設合併兩個子樹 T_1, T_2 的複雜度爲 $\mathcal{O}(T_2 f(n))$，如果用上述的方法，則整體的複雜度是 $\mathcal{O}(n \log(n) f(n))$。

Proof. 合併兩個子樹 T_1, T_2 的複雜度爲 $\mathcal{O}(T_2 f(n))$，可以想作把 T_2 裡的點都丟到 T_1 去，且丟一個點的平均複雜度是 $f(n)$。現在一個點被丟到的新的子樹，表示他原來所屬的子樹大小比當前最大的子樹還小，因此合併後他所屬的新子樹會至少是原來的一半，因此每個點最多被丟到新的子樹 $\mathcal{O}(\log n)$ 次，因此總複雜度是 $n \log(n) f(n)$。 □

1.1.2 平面圖

平面圖就是可以畫在平面上的圖，使得任兩個邊只會在點上相交而已。

而平面圖有一個很重要的定理。

定理 (歐拉定理): 對於一個**連通**的平面圖，有

$$E = V + F - 2 \tag{1.2}$$

其中 E, V, F 分別代表圖的邊數、點數還有面數。一個**面**定義爲被邊所切出的一個區域(包含最外面的無限區域)。

歐拉公式給出了平面圖邊和點的關係式。而從公式中也可以看出平面圖邊的個數不會太多! 事實上對於一個簡單的平面圖，有以下定理:

定理: 對於一個**簡單連通**的平面圖，如果 $|V| \geq 3$ 則有

$$|E| \leq 3|V| - 6 \tag{1.3}$$

Proof. 由 (1.2)，$|E| = |V| + |F| - 2$。但簡單圖沒有自環及重邊，一個面一定至少有三條邊，$|F| \geq 3|E|$。代入 (1.2) 得 $|E| \leq 3|V| - 6$。 □

邊很少的這個條件往往是解題的關鍵!

例題 **1-3** Defense Your Country NTUJ 2126

給你一個平面圖，請找一個最大團，也就是最大的一個完全子圖。($|V| \leq 2 \cdot 10^5$)

一看這個題目，不得了，連 $|E|$ 都沒有給，但不要忘了由 (1.3) 邊的數量不會超過 3 倍的 $|V|$。

而最大團是個 NPC 問題，如果我們不善用平面圖的性質，肯定解不出來。注意到**平面圖的子圖仍是平面圖**，而完全圖的邊的數量 $\mathcal{O}(E) = \mathcal{O}(V^2)$，平面圖的是 $\mathcal{O}(E) = \mathcal{O}(V)$。肯定**有個上界 m，K_m 絕對不會出現在平面圖上**。事實上算一下會發現對於 K_5，邊有 10 條，點有 5 個，而 $10 > 3 \cdot 5 - 6 = 9$，所以根據 (1.3)，K_5 肯定不是平面圖! 因此平面圖的最大團一定不超過 4。

不過這題還沒有結束，雖然我們只要檢查 4 個點以下的完全圖，但還是要有一個有效率的方法，否則 $\mathcal{O}(V^4)$ 枚舉肯定要超時。再注意到 (1.3) 其實告訴了

$$\sum \deg(v) = 2E \leq 6V - 12.$$

因此 $\deg(v)$ 的平均值小於 6 ，也就是一定有一個點的度數小於 5，所以我們可以先找一個 v_1 使得 $\deg(v_1) \leq 5$ ，然後 2^6 枚舉他和他的鄰居的最大團就可以了。枚舉完了這個點，將這個點從圖上移除，新的圖仍是一個平面圖，因此我們又可以再找一個 v_2 使得 $\deg(v_2) \leq 5$。一路做下去我們就得到了一個 $\mathcal{O}(V)$ 的做法!

這裡再總結一下平面圖的一些結論。

定理： 定義一個圖的 degeneracy 爲 $\min \deg(v)$，則平面圖的 degeneracy ≤ 5。

定理： 一個圖是平面圖的條件若且唯若 $K_5, K_{3,3}$ 不是他的 minor。G 是 H 的 minor 表示 G 可以由

- 刪掉一條邊
- 刪掉一個點
- 收縮一條邊，也就是把兩個點合併。

得到。

1.1.3 競賽圖

一個競賽圖是一個有向圖，且 $(u, v), (v, u)$ 恰好有一個在 E 中。可以想成是「有向完全圖」。競賽圖也有許多有趣的性質，如以下這個例題。

例題 **1-4** 競賽圖的 Hamilton Path 經典問題

給你一個競賽圖，求他的 Hamilton Path。

如果你把競賽圖想成是「兩兩對局結果」，也就是如果 $u \to v$ 就表示 u 贏 v，那一個 Hamilton path 其實就是一個「幾乎」[1] 正確的強度順序，因此我們不如往排序的方向想。

而衆多排序法，大家第一個想到的應該是最常用的快速排序法。我們可以模仿快速排序法的方法，先隨便挑一個點 v 出來，之後把所有他「贏過」的點 u 挑出來，也就是 $v \to u \in E$ 的點。假說這個集合是 A，那 $\bar{A} \setminus v$ 裡的點 w 由競賽圖的性質可以知道 $w \to v \in E$，因此我們只要遞迴下去做 $A, \bar{A}$，最後用 v 這個點把他們兩個的 Hamilton path 串起來就可以了!

有趣的是這題用「幾乎所有的排序法」都會對，如 Merge sort 或是 Insertion sort 等等。

1.1.4 二分圖

如果一個圖可以被分成兩個點集 U, V 使得任兩個點 u, v 如果在同一個集合的話，他們之間一定沒有邊，即 $(u, v) \notin E$，則我們就稱這個圖是一個二分圖。這也等價於可以被二著色，且相鄰的點不同色。

要判斷一個圖是否是二分圖，一個最簡單的方法就是把這個二分圖遞迴黑白染色，也就是從一個點 u 開始，把 u 先圖成白色，再把其周圍的點圖成黑色，然後把距離 2 的點都塗成白色 ...依此類推，最後再檢查是不是每一個邊的兩端都不同色即可。

另一個做法是用**併查集**，對每一個點 u 都建兩個對應的點，u_0, u_1。可以把 u_0 想做是「你的顏色」，u_1 則是「非你的顏色」。對於每一條邊 (u, v)，我們就把 $(u_0, v_1), (u_1 v_0)$ 做併查集的合併，意思就是「u 的顏色是非 v 的顏色」，而圖是二分圖的條件即是每個點 u 對應到的兩個新點 u_0, u_1 都在不同的併查集裡，如以下這個例題。

例題 **1-5** 分子碰撞實驗 TIOJ 1672

現在有一個未知的序列 $a_1, a_2, \ldots, a_n$，依序給你些個資訊 (s, t, p)，其中 $p \in \{0, 1\}$ 代表 $a_s + a_{s+1} + \cdots + a_t \equiv p \pmod 2$，每次你要判斷給你的資訊會不會和已知的資訊有所矛盾，並且如果有矛盾則捨棄之。

如果我們令 $b_i = \sum_{0}^{i-1} a_j$，可以發現每次的資訊其實就是在告訴你 a_{s-1} 和 a_t 的奇偶性是相同還是相異。因此我們就修改上述的做法就可以了! 假設 $u = a_{s-1}$ 且 $v = a_t$ 並用剛才的記號，如果他們同奇偶則檢查 (u_0, v_1) 是否在不同的集合裡，是的話再合併 $(u_0, v_0), (u_1, v_1)$，反之亦然。

[1] 會說「幾乎」的原因是因爲這個關係不一定有傳遞性，也就是 $(u \to v) \wedge (v \to w) \not\Rightarrow u \to w$。

1.2 有關圖的問題

與圖論有關的問題非常多，幾本上不太可能全部掌握，除了多了解一些結論之外，舉一反三和臨場見招拆招的能力也是很重要的!

1.2.1 計算複雜度理論

「知己知彼、百戰百勝」，要攻克一個題目，我們如果能先知道這個題目有「多難」會是一大幫助。

那要如何區分問題的難度呢? 通常我們說一個問題是「容易的」表示我們已經知道有一個多項式時間的演算法可以求出這個問題的答案。比如排序問題，最短路徑問題等等。

但是有一些問題，人們想破頭了都想不出一個多項式時間的演算法可以解出，比如求一個圖的 Hamilton Path 至今都還沒有一個好的演算法解決他，而且也沒有辦法證明不存在一個多項式時間的演算法解決他。

百般無奈的人們只好先假設我們有一個更加強大的計算體系了，也就是**非確定性圖靈機 (Non-deterministic turing machine)**。

在介紹什麼是非確定性圖靈機之前，我們先定義一個合法的問題是什麼。在這邊我們先只考慮**判定性問題**。什麼是判定性問題呢? 其實就是答案只有 Yes, No 兩種的問題，比如說「這個圖上是不是存在一個 Hamilton path 呢?」或是「這個圖是不是一個平面圖?」等等。至於「從 u 到 v 的最短路徑長度是多少?」就不是判定性問題了。但是我們大可以把他轉成「從 u 到 v 的最短路徑長度是否小於 k?」這個判定性問題，然後用比如說二分搜的方法解出原本的問題。

而非確定性圖靈機強大的地方就在於，有別於我們電腦等等的**確定性圖靈機**的計算體系，在進行搜索時有時後我們要面臨抉擇。比如說在找一個圖的 Hamilton path 時，我們不知道起點應該是 $v_1, v_2, \ldots$ 裡的哪一個，於是只好一個個試過一遍。非確定性圖靈機在面臨這種情況時，如果答案是「有解存在」的，他一定會**很幸運的選到一個最好的選擇**。

於是我們把所有用確定性圖靈機可以在多項式時間內解出的題目的集合稱作 P，把所有用確定性圖靈機可以在多項式時間內解出的題目的集合就稱作 NP。[2] 在經過一番推敲後我們發現**一個判定性問題是 NP 問題若且唯若如果答案是 Yes，他的一個解可以在多項式時間被一個內被一個確定性圖靈機驗證**。比如說 Hamilton Path 是 NP 問題，因爲今天如果有人宣稱 P 是一個 Hamiltom Path，那你只要確認 P 通過所有點恰好一次，而且每一條邊都存在於原本的圖中即可。

[2]所以 NP 並不是 Non-Polynomial 的縮寫，而是 Non-deterministic Polynomial 的縮寫。

定義了 NP 後，人們就開始問，NP 裡最難的問題是什麼呢？我們說一個問題 A 比 B 難表示 B 可以化簡成 A，也就是說**如果我們要解 B 問題，我們可以把 B 在多項式時間內作一些轉化，把他變成 A 問題。** 比如我們可以證明找一個 Hamilton Cycle 比找 Hamilton Path 還難，因爲如果要找 Hamilton Path，我們可以再原圖多加一個點 v ，然後把 v 建邊連到所有點。這樣原圖的一個 Hamilton Path 一定可以頭尾用 v 串起來變成 Hamilton Cycle。而在新的圖如果找到了一個 Hamilton Cycle，只要把 Cycle 中的 v 去掉，就是原本 Hamilton Path 的解了。

也就是說，如果你會解 Hamilton Cycle，那你一定也能解 Hamilton Path。因此 NP 裡「最難」的問題就代表**如果你會解這個問題，那所有 NP 問題都可以在多項式時間化簡為這個問題。**不過「最難」的問題眞的存在嗎？有的！其實以經有許多問題被證明是「最難」的 NP 問題了，如 3-SAT、最大團還有剛剛說的 Hamilton Path 等都是！我們把這些問題叫作 NPC (NP-complete) 問題。

而如果我們有一天找到了一個可以在多項式時間內解出某一個 NPC 問題的演算法，那不得了了，因爲 NPC 已經是 NP 裡最難的問題了，所有 NP 問題都可以在多項式時間內解出！

很遺憾的是這些 NPC 問題我們到目前爲止都找不到一個多項式時間的演算法，也沒有辦法證明這樣的演算法不存在！

這和競賽解題有何關係？

有的！這往往可以幫助你往正確的方向思考。在解題的過程中我們往往會把題目做一些變換，轉成其他的題目來解。而如果你發現你轉成了一個 NPC 問題，除非測資範圍很小，否則通常表示你一定有什麼題目的性質沒有用到，要從新思考思路！比如前面說的「平面圖最大團」問題，一般圖的最大團問題是 NPC，因此關鍵一定在平面圖的性質上！

接下來我們就討論一些圖上常見的問題！

1.2.2 2-SAT

例題 1-6 2-SAT　　經典問題

給你一個關於變數 $x_1, x_2, \ldots, x_n$ 且形式如下的布林代數式

$$(y_1 \vee y_2) \wedge (y_3 \vee y_4) \wedge \cdots \wedge (y_{2m-1} \vee y_{2m})$$

其中 $y_1, y_2, \ldots, y_{2m}$ 都是某個 x_i 或是 $\neg x_i$。也就是每個括號裡都恰有兩個變數 or 起來後再全部 and 起來。問你是否可以給每一個變數 x_i 一個 `true` 或是 `false` 值使得原本的布林代數式的結果爲 `true`？

這個問題有一個很 well known 的解法，對於每個刮號，假設是 $(x_3 \vee \neg x_4)$ 好了，代表不是 $x_3 = \texttt{true}$，就是 $x_4 = \texttt{false}$。也就是

- 如果 $x_3 = \texttt{false}$，那必定有 $x_4 = \texttt{false}$，用邏輯的語言寫就是 $\neg x_3 \to \neg x_4$。
- 如果 $x_4 = \texttt{true}$，那必定有 $x_3 = \texttt{true}$，用邏輯的語言寫就是 $x_3 \to x_4$。

不知道大家有沒有感覺和二分圖判定的時候有點像！只是現在的關係是單向的而已，如上述例子，$x_3 \to x_4$，但沒有強迫 $x_4 \to x_3$。

因此解法就是把每一個變數看成是兩個點 $x_i, \neg x_i$，再如上面的例子所示把所有關係都建一個有向邊出來，基本上 $(x_i, x_j), (\neg x_i, \neg x_j)$ 的建法也類似，讀者可以自己推敲，只有 (x_i, x_i) 這種兩個變數相同比較特別，想一下後可以知道需要把 $\neg x_i \to x_i$。[3] 類似的 $(\neg x_i, \neg x_i)$ 就建 $x_i \to \neg x_i$。

最後可以發現如果在圖上 u 可以走到 v，表示如果「選了」u 就一定要選 v，因此可以想像有解的條件就是對於所有 $x_i, \neg x_i$，要嘛 u 不能走到 v，要嘛 v 不能走到 u，用圖論的說法就是$x_i, \neg x_i$ **不能在同一個強連通元件！** 用 Tarjan 等等的演算法即可以在 $\mathcal{O}(V + E)$ 的時間內做到。

例題 1-7 以前看過的題目 經典問題

給你一個 $R \times C$ 的方格，你有一種特殊的炮彈可以炸一排或一列，不過同一排或一列最多只能炸一次。每一格除了可能是空地外，還有可能是

敵方要塞 你必須炸兩次才能消滅他。
敵方軍團 你必須炸一次以上才能消滅。
住宅區 爲了必免無辜的人受害，最多只能炸一次。
醫院 這個不用說，當然是炸都不能炸。

問你每一排及每一列該選擇炸或是不炸，才能使每一格都符合上述的需求？

我們可以用布林變數 x_i 表示第 i 排要炸還是不炸，用 y_i 表示第 j 排要炸與否，那這些規定其實就是

敵方要塞 $x_i \wedge y_j \equiv (x_i \vee x_i) \wedge (y_j \vee y_j)$
敵方軍團 $x_i \vee y_j$
住宅區 $\neg x_i \vee \neg y_j$
醫院 $\neg x_i \wedge \neg y_j \equiv (\neg x_i \vee \neg x_i) \wedge (\neg y_j \vee \neg y_j)$

轉成 2-SAT 解即可!

3-SAT

很不幸的 3-SAT 問題，也就是每個刮號可能有三個變數，雖然好像只差一點卻是個 NPC 問題。不過並不是 NPC 問題就不值得研究或是優化，有好的搜索方法還是會差很多！現在有些 SAT-solver 在一般沒有設計過的測資 (如電路等等) 似乎跑到 10^4 個變數都不是問題，好像也是目前在解如電路等價等等一個很不錯的解法。

[3] 用邏輯的解釋，因 $a \to b \equiv \neg a \vee b$，$\neg x_i \to x_i \equiv (\neg\neg x_i) \vee x_i \equiv x_i$

1.2.3 著色問題

點著色

點著色就是要把圖上的點都塗上顏色，使得相鄰的點都要不同色，並要求用的顏色越少越好。如果最少用 k 個顏色就足夠了，我們就稱這個圖的著色數爲 k。點著色是 NPC 問題，但是在特殊的圖可能會有特殊的方法。比如大家都熟悉的四色定理

四色定理： 任意一個平面圖一定可以 4 著色。

我們來看一個例題。

例題 **1-8** K Graph Oddity — NERC 2010-2011 pK

給你一個連通圖 G 滿足 G 有奇數個點，且存在一個奇數 k 使得所有點的度數都不超過 k。請你把這張圖點 k 著色。

其實畫一畫會發現，因爲度數最高是 k，要把一個圖 k 著色其實非常容易。如果一個點的度數小於 k，也就是最多 $k-1$，那我們跟本不用考慮這個點，等其它點都塗色了以後，再找一個顏色不在其 $k-1$ 個鄰居中就可以了。

那會不會所有點的 degree 都是 k 呢？不會！否則所有點的度數合是 $k|V|$，這個等於 $2|E|$，但 $k, |V|$ 都是奇數，乘出來居然是偶數，矛盾！因此一定有一個點的度數不超過 $k-1$，那因爲這個點更本不需考慮，甘脆把他拔掉吧！而又因爲這個點連通，所以拔掉這個點之後，一定又有一個點的度數變少了，也就是不超過 $k-1$。一路做下去，所有的點都被我們拔掉了！

那題目就很簡單了，把最後一個拔掉的點塗隨便一個顏色，再把倒數第二個點塗上任一個合法的顏色...一路做回來，我們上面說明了因爲每一次的 degree 都不超過 $k-1$，一定能找到一個合法的顏色塗！

邊著色

類似地，邊著色就是要把圖上的邊都塗上顏色，使得任一個點連出去的所有邊都不同色。邊著色仍然是一個 NPC 問題！但有一個很神奇的結論。

Vizing's theorem: 任意一個簡單圖的邊著色數一定是 $d, d+1$ 中的一個，其中 $d = \max_{v \in V} \deg(v)$。

顯然邊著色數至少要 d，但 $d+1$ 一定可以著色就沒有那麼顯然了。Vizing's theorem 的證明中就給了一個 $d+1$ 著色的演算法，但是滿複雜的，有興趣可以到 Wiki 上看看。

1.2.4 歐拉路徑/回路

給你一張圖，一個歐拉路徑 (Euler path) 是一條連續的路徑 (點可重複) 恰好經過所有的邊。如果終點恰好回到起點，就稱作是一個歐拉回路。

一筆劃問題就是要問圖上存不存在一條歐拉回路。觀察一下可以發現一個路徑 $P = (v_1, v_2, \ldots, v_n)$，除了起點和終點以外，每個點都會從一個邊進去，再從另一條邊出來，總共會「吃掉」這個點 2 個度數。但因點可能重複出現在路徑 P 上，因此我們可以知道除了起點/終點外，每個點的度數都要是偶數! 事實上對於連通圖，這不僅是必要條件，還是充分條件。

定理 定義一個**奇點**是度數爲奇數的點。如果圖連通，有

- 一個無向圖有歐拉路徑的條件是其奇點的個數是 0 或 2。
- 一個無向圖有歐拉迴路的條件是其奇點的個數是 0。
- 一個有向圖有歐拉路徑的條件是所有點都滿足 $\deg^+(v) = \deg^-(v)$，或是除此之外有一個點滿足 $\deg^+(v) = \deg^-(v) + 1$，另一個滿足 $\deg^+(v) = \deg^-(v) - 1$。
- 一個有向圖有歐拉迴路的條件是所有點都滿足 $\deg^+(v) = \deg^-(v)$，

例題 1-9 構造字串 經典問題

問你存不存在一個字串 S 使得子字串 'ab' 恰好出現過 $c_{a,b}$ 次，其中 $a, b \in$ '0123456789'。

如 $c_{0,0} = 2, c_{0,1} = 1, c_{1,1} = 1, c_{1,0} = 2$，有字串 '1100010'。

如果把字串，如 '13413' 想像做從 $v_1 \to v_3 \to v_4 \to v_1 \to v_3$ 的一條路徑，其實原本的題目就是要問你有沒有一條歐拉路徑，計算每一個點的度數即可。

Hamiltonian path

一個 Hamiltonian path 是一個簡單路徑通過所有的點恰好一次。雖然看起來跟 Euler path 很像，但卻是一個 NPC 問題。通常的做法是用狀態壓縮 DP ，複雜度是 $\mathcal{O}\left(2^V V^2\right)$。

1.2.5 匹配問題

一個圖的匹配是一個邊集 E'，滿足 $E' \subseteq E$ 且每個在圖上的點至多只跟 E' 裡面的一條邊相鄰。一個最大匹配就是 $|E'|$ 最大的一個匹配。

二分圖上的匹配問題

二分圖上的最大匹配可以直接轉成最大流問題求解 [4]。我們這邊用另一種看法來看。首先我們定義何謂交錯路徑。

定義： 一個**交錯路徑**是一個路徑 $P = (x_1, y_1, x_2, y_2, \ldots, x_n, y_n)$，滿足

- 路徑上的每一條邊都在原圖的邊上。
- x_1, y_n 都還沒有被匹配。
- (y_k, x_{k+1}) 都是已經在匹配中的邊。
- (x_k, y_k) 都是還沒有在匹配中的邊。

找到一條交錯路徑後，把 (y_k, x_{k+1}) 共 $n-1$ 個從匹配中移除，並把 (x_k, y_k) 加進匹配中，就是一個大小多 1 的新匹配。

可以證明一個匹配是最大匹配的條件就是圖上已經不存在交錯路徑，因此我們可以不斷的建一棵**交錯樹**，找交錯路徑增加匹配大小，直到不存在交錯路徑爲止。詳細的證明和方法用 flow 的角度來看非常清礎，我們就不在細講。

一般圖上的匹配問題

和二分圖相似，不斷的找交錯路徑擴充。不過不同的是二分圖的交錯路徑 $P = (x_1, y_1, x_2, y_2, \ldots, x_n, y_n)$ 剛好有 $x_i \in X, y_i \in Y$，其中 X, Y 是二分圖的一個二分子集 (也就是 X, Y 各自裡面都沒有邊)。一般圖就沒有這種好事了，會造成交錯樹沒有那麼好建，會有**花 (Blossom)** 的存在。

這時候就要修改演算法成**縮花演算法 (Blossom Algorithm)**，不過縮花演算法蠻難寫的，最好是準備一份放在 Codebook [5] 裡。

當點數如果不大，可以直接用 $\mathcal{O}\left(2^V V^2\right)$ 狀態壓縮 DP 來解。

帶權匹配問題

有時後邊上會有權重，你要找一組邊權總合最大的匹配。

在二分圖可以直接轉成 Cost-flow 來做，也有從 Dual problem 下手的演算法如 Hungarian algorithm 。

在一般圖上也有多項式時間的演算法，不過出名的難寫。有一種其妙的假解是基於以下觀察

[4] 可以參考「最大流」那一章

[5] 或許有人不知道，ACM-ICPC 比賽每個隊伍都可以帶約 25 頁單面的 code。

定義： 一個最大帶權匹配中的負環定義爲一個交錯環
$P = (x_1, y_1, x_2, y_2, \ldots, x_n, y_n, x_1)$，滿足

- $(y_k, x_{k+1})\ \forall k$ 和 (y_n, x_1) 都是已經在匹配中的邊，把這些邊的權重和叫作 C_1。
- (x_k, y_k) 都是還沒有在匹配中的邊，把這些邊的權重和叫作 C_2。
- $C_2 > C_1$

找到一個負環後，把上面 n 個已匹配邊從匹配中移除，其它 n 個加到匹配裡，那新的匹配的權重和更大。

發揮創意不斷的找負環直到沒有，就是一個最大帶權匹配了。

例題 1-10 中國郵差問題 經典問題

給你一個圖，邊上有權重代表路徑長，你要找一個路徑，經過所有的邊至少一次(可以重複走)，最後回到出發點，問最短的路徑長。

這個有點像歐拉迴路，可是每個邊可以重複經過。想法是想辦法把圖上的奇點都「補」成偶點。如果補兩個奇點 (u, v)，那這兩個點的度數都加 1 變成偶點，而當然是補一條 $u \rightsquigarrow v$ 的最短路徑最賺。因此其實我們就是要把所有奇點兩兩配對，(u, v) 配對的花費就是他們之間的最短路。

總結以上，要做的事情就是把所有奇點對 (u, v) 的最短路徑求出，建成一個完全圖，然後做最小帶權匹配，可以把邊權取負號，就變成最大帶權匹配。

一些相關的問題

我們這邊列了一些相關的問題。

最大獨立點集 一個最大的點集 V' 使得裡面的點都不相鄰。其大小記做 $I(G)$。
最大匹配數 前面定義過了。其大小記做 $M(G)$。
最小點覆蓋 最小的一個點集，使得所有的邊都至少與點集裡的一個點相鄰。
其大小記做 $C_v(G)$。
最小邊覆蓋 最小的一個邊集，使得所有的點都至少與邊集裡的一個邊相鄰。
其大小記做 $C_e(G)$。

這些問題是很有相關性的，有以下定理:

定理 對於連通圖，有

- $I(G) + C_v(G) = |V|.$ (1.4)
- $M(G) + C_e(G) = |V|.$ (1.5)
- 對連通二分圖，有 $M(G) = C_v(G), I(G) = C_e(G).$ (1.6)

Proof. 對 (1.4) 我們證明一個獨立點集 U 的補集 U^c 一定是一個點覆蓋。對於所有邊 (u,v)，可知 u,v 至少有一個不在 U，也就是在 U^c，因此 U^c 覆蓋所有邊。這也同時證明了一個點覆蓋 U 的補集 U^c 一定是一個獨立點集。因此 $I(G)+C_v(G)=|V|$。

對 (1.5)，我們先證明 $C_e(G)\le |V|-M(G)$。先找一個最大匹配 M，注意到 M 恰好蓋住了 $2|M|=2M(G)$ 個點，又因圖連通，可在找 $|V|-2M(G)$ 個邊就把所有點都蓋住了，因此 $C_e(G)\le |V|-2M(G)+M(G)=|V|-M(G)$。

我們再證明 $C_e(G)\ge |V|-M(G)$，也就是 $M(G)\ge |V|-C_e(G)$。先找一個最小邊覆蓋。這些邊 (加上端點) 會形成許多連通塊，而對於任一個連通塊，不會有環，否則隨便拔掉一條環上的邊仍可覆蓋所有點，也就是每個連通塊都是一棵樹。而每一個連通塊任選一條邊，這些選出來的邊都不相鄰，所以是一個匹配。假設連通塊分別爲 $S_1,S_2,\ldots,S_n$，有 $V(S_1)=E(S_1)+1$。因此

$$M(G)\ge n=\sum V(S_i)-E(S_i)=\sum V(S_i)-\sum E(S_i)=|V|-C_e(G)$$

最後證 (1.6)。首先對於一個匹配 M 中的任一個邊，點覆蓋一定要蓋住其中一點，因此 $C_v(G)\ge M(G)$。現在如果有一個圖 G 滿足 $C_v(G)>M(G)$，找 $|V|,|E|$ 最小的一個反例。容易證明 G 不是一條路徑或是一個環，因此有一個點 u 的度數至少是 3。假設 u 和 v_1,v_2,v_3 相鄰，如果 $G_1=G\setminus v_1$ 的最大匹配數 $M(G_1)<M(G)$，則因 $C_v(G_1)$ 至少是 $C_v(G)-1$，有 $M(G_1)\le M(G)-1<C_v(G_1)$，與反例的最小性矛盾。因此存在一個最大匹配 M 使得 $v_1\notin M$。而 $(u,v_2),(u,v_3)$ 至少有一個不在 M 中，假設是 $e=(u,v_2)$，那考慮 $G_2=G\setminus e$ 的最小覆蓋 C，由反例的最小性有 $M(G)=M(G_2)=C_v(G_2)$，且因爲對於一個匹配 M 中的任一個邊，點覆蓋一定要蓋住其中一點，$v_2\notin C$，從而 $u\in C$ 所以 C 也是 G 的覆蓋，$M(G)=C_v(G)$，矛盾。因此 (1.6) 成立。 □

(1.6) 的證明其實沒有給出一個構造點覆蓋的方法，實際的構造方法有些複雜，可以參考 Wiki。

1.2.6 圖的計數問題

有時候也有一些圖的計數問題出現，比如說

例題 1-11 二元樹的數量 經典問題

求 n 個點可以形成多少種不同的有根二元樹。

不少計數問題可以用遞迴的方式想，比如這題，不妨先想想何謂一個二元樹？其實就是「根節點有兩個二元子樹的樹」。因此我們可以寫出遞迴式，令 $f(n)$ 表示 n 個點的二元樹個數，有

$$f(n)=\sum_{k=0}^{n-1} f(n-k-1)f(k),\quad f(0)=1$$

對組合熟悉的讀者應該一眼就可以看出這個就是著名的**卡特蘭數**。

以下也是一個類似的題目

例題 1-12 樹的數量 經典問題

求 n 個有編號的點可以形成多少種不同的樹。

這個是非常經典的問題，答案是 $n^{n-2}, n = |V|$，證明如下:

Proof. 我們現在對於一棵樹，都把他轉成一個序列，過程如下。

每次找編號最大的一個樹葉 (也就是 degree 是 1 的點) u，假設他連出去的邊是 (u, v) ，就記下 v 的編號。重複 $|V| - 1$ 次後就只剩一個點了。

假設得出的序列是 $(a_1, a_2, \cdots, a_{n-1}), n = |V|$，注意到 a_{n-1} 必定是 1，所以我們不妨只看 $(a_1, a_2, \cdots, a_{n-2})$，其中 a_i 可以是 1 到 n 的任何數字。我們只要能證明一個樹可以一對一到一個這樣的序列，一個這樣的序列也一定可以還原成一棵樹即可。

注意到如果 $\deg(v) = k$，那 v 就會在序列中出現 $k - 1$ 次。當兩個樹 T_1, T_2 對映到的序列 A_1, A_2 相同，如果 $|T_1| = 2$，那這兩個樹肯定相同。接下來我們用數學規納法，假設這兩個樹的大小爲 k，現在考慮他們編號最小的葉子 u_1, u_2，一定有 $u_1 = u_2$，否則假設 $u_1 < u_2$ ，那 u_1 不會出現在 a_1 裡但不會出現在 a_2 理。接著因爲序列的第一項相同，u_1, u_2 也要接到相同的點，拔掉這個點後由規納假設，兩棵樹仍相同，因此 $T_1 = T_2$，也就是不同的樹會對映到不同的序列。

還原的方法也很簡單，如果 $n = 2$ 那只有一種可能的樹，就是 $1, 2$ 接在一起。假設我們要還原 $n = k$，也就是序列 A 長度是 $k - 2$，我們找最小的編號 u 使得 u 沒有出現在 A 裡，先遞迴建出 $A' = (a_2, a_3, \ldots, a_n)$ 的樹，把 u 接上 a_1 即可。

因此一個樹可以一對一且映成對到這樣的序列，容易計算這樣的序列有 n^{n-2} 種! □

再看一個比較不一樣的題目

例題 1-13 三角形的數量 經典問題

給一個圖，求有多少點對 (v_1, v_2, v_3) 滿足 $(v_1, v_2), (v_2, v_3), (v_3, v_1) \in E$。

如果直接枚舉 v_1, v_2, v_3 是個 $\mathcal{O}(V^3)$ 的做法，但聰明一點的做法是枚舉所有邊 $(v_1, v_2) \in E$，再枚舉所有點 v_3，是 $\mathcal{O}(VE)$。

更甚著，可以發現不需要枚舉所有點 v_3，只要枚舉比如 v_2 所有的鄰居就可以了，我們挑 $k = \sqrt{|E|}$，再枚舉三角形的時候如果 (v_1, v_2, v_3) 的度數都比 k 大，我們就直接用第一種方法枚舉，注意到這種點只會有 $\mathcal{O}\left(\sqrt{E}\right)$ 個，所以複雜度是 $\mathcal{O}\left(E\sqrt{E}\right)$。接下來用第二種方法枚舉，對一個邊 (v_1, v_2)，如果 $\deg(v_1) > k, (v_2) > k$ 就捨棄，如果有一個，假設是

$v_2, \deg(v_2) < k$，我們就在枚舉 v_2 的鄰居，雜度是 $\mathcal{O}\left(E\sqrt{E}\right)$。

因此三角形有一個點的度數比 k 小，一定會在上面被枚舉到，全部都比 k 大我們第一種方法也枚舉到了，經過適當的排容，總複雜度是 $\mathcal{O}\left(E\sqrt{E}\right)$。

去年的講義資料分治 (Square root method) 有更詳細的說明。

1.2.7 其它的問題

前面講了那麼多種圖論的問題，可以發現圖論問題眞的是非常豐富！在程式競賽中更可能會出現其它奇奇怪怪的問題，這時候就只能見招拆招了！如下面這一題

例題 1-14 怪圖怪題 經典問題

一個 Union-Join graph 定義爲

- 一個點 v 是一個 Union-Join graph.
- 如果 G, H 都是 Union-Join graph，那他們的 Union

$$G \cup H = (V(G) \cup V(H), E(G) \cup E(H)),$$

也是 Union-Join graph.

- 如果 G, H 都是 Union-Join graph，那他們的 Join

$$G \oplus H = (V(G) \cup V(H), E(G) \cup E(H) \cup E'),$$

其中 $E' = \{(u, v) : \forall u \in G, v \in H\}$ 也是 Union-Join graph.

現在給你一個 Union-Join graph，你要把每一個點填上一個數字 $c(v)$，使得如果兩個點 u, v 的數字相同，那任何 $u \rightsquigarrow v$ 的路徑上，都要有一個點 w 使得 $c(w) < c(u)$。問你至少要用幾個不同的數字。

乍看之下題目無比複雜，但越複雜的題目可能反而是越簡單的題目！首先觀察到這個圖是**被遞迴**的定義出來的，因此我們不妨就猜測，假設最少需要的數字數量是 $\tau(G)$，如果 $G = G_1 \cup G_2$，那 $\tau(G) = \tau_\cup(G_1, G_2)$，且如果 $G = G_1 \oplus G_2$，那 $\tau(G) = \tau_\oplus(G_1, G_2)$。我們只需要求出 $\tau_\cup, \tau_\oplus$ 這兩個函數是怎麼樣的函數就可以了。

顯然 $\tau_\cup(G_1, G_2) = \max(\tau(G_1), \tau(G_2))$，因爲 Union 其實就只是把兩張圖擺在一起而已，彼此互不影響。至於 $\tau_\oplus$ 比較複雜，觀察後可以發現

1. 兩邊的點不能有相同的數字，不然 union 就把兩個相同的數字連起來了。
2. Union 後只有一邊裡的點可以有相同的數字，另一邊內的點一定要全不相同。不然如果 G_1 中 u_1, u_2 相同，G_2 中 v_1, v_2 相同，如果 $c(u_1) < c(v_1)$，union 後會有路徑 $u_1 \to v_1 \to u_2$ 不符合要求，反之亦然。

因此由上面討論我們猜想 $\tau_\oplus(G_1, G_2) = \min(\tau(G_1) + |V(G_2)|, \tau(G_2) + |V(G_1)|)$。由 1. 2.

知道這是下界，而一邊的點 (假設是 G_1) 選 $1, 2, \cdots, |V(G_1)|$，一邊令 $c(v) = c'(v) + V(G_1)$ 其中 $c'(v)$ 是原先的答案，顯然是一組解，因此這個答案是正確的。

1.3 習題

習題 1-1 樹上模 m 的最長路徑 經典問題

給一個邊有權的樹，求一個路徑使得路徑上權值和模 m 下最大。[$|V| \leq 10^5$]

習題 1-2 Planar Graph Codeforces 223E

給一個連通的平面圖，每次會給你圖上的一個環 C_i，問你環內有幾個點。
[$|V| \leq 10^5, \sum |C_i| \leq 10^5$]

習題 1-3 Conntected Graph POJ 1737

問你 n 個有編號的點可以形成多少不同的連通圖。[$n \leq 50$]

習題 1-4 Edge coloring of bipartite graph Codeforces 600F

給你一個二分圖，求他最小的一個邊著色。[$|V| \leq 2000, |E| \leq 10^5$]

習題 1-5 Four Colors ASC 47 pF

現在有一棵樹，A, B 兩個人玩一個遊戲，A 先開始輪流選一個還沒有塗色的點塗成 4 種顏色的其中一種，但相鄰的兩個點不能塗同一種顏色。如果一個人不能行動遊戲就結束。遊戲結束時所有的點都被塗色的話 A 贏，否則 B 贏。問 A 的必勝策略？

習題 1-6 Halin Graph NTUJ 2061

一個 Halin graph 就是一個畫在平面上的樹，然後把所有葉子用順時針依序連起來，最後一個葉子在連到第一個葉子。一個 3-regular graph 就是每個點的 degree 都是 3。給你一個邊有權的 3-regular Halin Graph，求一個權值最大的 Hamiltonian graph [$|V| \leq 10^5, |E| \leq 1.5 \cdot 10^5$]

習題 1-7 Tax PA 2012

給你一個圖 G ，每個點上有權重 $c(v)$，你要從 s 走到 t，每走一條邊 (u, v) 你就要付 $\max(c(u), c(v))$ 元，問你最小花費。[$|V| \leq 1000, |E| \leq 10^5$

習題 1-8 Flights ASC 45 pF

給你一個圖，其中 v_1 與所有其它點相鄰。你要給每個邊一個 $1, 2, \ldots, |E|$ 的數字，滿足令 $f(v)$ 表示所有與 v 相鄰的邊的權重和，你要讓所有 $f(v)$ 都不同。[$|V| \leq 10^5, |E| \leq 2 \cdot 10^5$

習題 1-9 同色三色形 經典問題

給你一個完全圖，圖的邊不是黑色就是白色，問你有幾個同色三角形。

習題 1-10 競賽圖 經典問題

對於一個競賽圖，請證明

1. 一定存在一個點 v，使得對於其它點 v，要嘛 $u \to v$ 有邊，要嘛存在一個點 w，使得 $u \to w \to v$ 可以通。
2. 如果不存在一個點 v 使得他贏過所有人，也就是不是所有 u，$v \to u$ 都有邊，並也不存在一個點輸給所有其它人，則存在 Hamiltonian cycle.

習題 1-11 Hamiltonian Path 經典問題

請證明如果圖中所有點的度數都比 $\lceil |V|/2 \rceil$ 大，則存在 Hamiltonian path.

習題 1-12 Strygwyr, the Bloodseeker NTUJ 2045

給你一個競賽圖，你要找一個 Hamiltonian path，使得假設這一個 path 爲 $v_1, v_2, \ldots, v_n, n = |V|$，對於所有 $i < j$，都存在一個長度不超過 2 的路徑 $v_i \rightsquigarrow v_j$。

習題 1-13 Drawing Hell NTUJ 2539

給你一個平面圖，兩個人玩一個遊戲，輪流連一條邊，但邊不可以和已經再圖上的邊相交 (只能在點上)。問先手還是後手贏。

習題 1-14 Massacre at Camp Happy TIOJ 1725

定義兩個字串 A, B 是 k-幾乎相同如果把 A 的前 k 個字元搬到最後面，那兩者恰相差一個字元，給你 A, B ，求出所有的 k 使得他們是 k-幾乎相同。[$|A| = |B| \leq 10^6$]

習題 1-15 最長回文子字串 經典問題

給一個字串 A ，求出他最長的一個回文子字串。[$\mathcal{O}(|A|)$]

這題可以用類似 Z-algorithm 的方法。我們先考慮如何求最長的**奇數**回文子字串。令 $\mathcal{Z}(i)$ 表示以 i 爲中心最長的回文字串爲 $A[i - \mathcal{Z}(i), i + \mathcal{Z}(i)]$。現在假設 $(L, R) = (i - \mathcal{Z}(m), i + \mathcal{Z}(m))$, 即 $A[L, R]$ 是一個以 $a[m]$ 爲中心的回文。如果 $j \in [m + 1, R]$，考慮 $j' = 2m - j$，即 j 對 m 的反射點。現在與 Z-algorithm 類似，令 $L' = j - \mathcal{Z}(j')$，分別考慮 $L' > L, L' = L, L' < L$ 三種情況。

但此方法僅能求得奇數的回文長度。因此我們把原字串相鄰的兩個字元都插入一個不在字元集 Σ 中的字元，如 $A =$ "abbab" $\to A' =$ "abbab"，這樣所有回文的長度都變成奇數了。

習題 1-16 Anti-hash text Codeforces Zlobober's blog

在 Hashing 實作上，因爲 C++ 的模運算 % 是非常花時間的，因此有一種 Lazy hash 的方法是直接 $\bmod 2^{32}$ 或是 $\bmod 2^{64}$，這樣直接在 `int`, `long long` 下做運算就可以了。但這樣做有其風險在，以下說明其原因。

不妨假設此 Hash function 爲

$$f(A) = \sum_{i=0}^{n-1} a_i p^{n-1-i} \bmod q$$

其中 p 爲一奇數 (否則此 Hash function 無意義)，$q = 2^m$，並且 "a", "b" 轉換成數字後分別爲 $\alpha, \alpha+1$ 。此時令

$$A_0 = \text{"a"}, \quad A_j = \widetilde{A}_{j-1} A_{j-1}$$

其中 $\widetilde{A}$ 表示把字串 A 中的 "a" 換成 "b"，"b" 換成 "a"。如:

$$A_1 = \text{"ba"}, \quad A_2 = \text{"abba"}, \quad A_3 = \text{"baababba"}, \quad \cdots$$

現在考慮 $A_k, \widetilde{A}_k$，注意這兩個字串恰好是 A_{k+1} 的前後半且 $A_k \neq \widetilde{A}_k$。請證明

$$\begin{aligned} f(\widetilde{A}_k) - f(A_k) &= p^0 - p^1 - p^2 + p^3 - p^4 + p^5 + \cdots + (-1)^{k-1} p^{2^k-1} \\ &= \sum_{i=0}^{2^k-1} b_i p^i \quad = S \end{aligned}$$

其中 $b_i \in \{1, -1\}$，且如果 i 的二進位表示有偶數個 1 ，則 $b_i = 1$，否則 $b_i = 0$。並且有

$$\sum_{i=0}^{2^k-1} b_i p^i = (p-1)(p^2-1)(p^4-1)\cdots(p^{2^{k-1}}-1)$$

注意到 $(p^{2^r}-1)$ 可以被 2^{r+1} 整除 (Why?) ，所以 S 可以被 $2^{1+2+\cdots+k} = 2^{k(k+1)/2}$ 整除。因此只要 $2^{k(k+1)/2}$ 被 2^m 整除，也就是 $\frac{k(k+1)}{2} > m$，A_k 和 $\widetilde{A}_k$ 的 Hash value 就會相同。因此可以用一個長度爲 $2^{\mathcal{O}(x)}$ 的字串構造出一個使 $q = 2^{\mathcal{O}(x^2)}$ 的 Hash function 失效的反例，並且與 p 的選擇無關。

習題 1-17 二維匹配 經典問題

給你一個 $R_A \times C_A$ 的二維字串 A ，問一個 $R_B \times C_B$ 的二維字串 B 有沒有出現在其中? [$\mathcal{O}(R_A C_A + R_B C_B)$]。

一個巧妙的做法可參考 Baker-Bird Algorithm，令外也有用 Hash 的方法，令 $c = C_B$, $a_i[j] = f(A[i][j, j+c-1])$，原題等價於問 $f(B[0]), f(B[1]), \cdots f(B[R_B-1])$ 是不是等於
$a_i[j], a_{i+1}[j], \cdots, a_{i+R_B-1}[j]$。這是一個一維的匹配問題。詳細可參考去年的講義。

習題 1-18 sed ACM ICPC 2013-2014 NEERC Northern Subregional

給你 A, B，求出字串 X, Y 使得將 A 中的 X 取代爲 Y 後會變成 B，並且 $|X| + |Y|$ 最短。取代的定義爲找 X 出現在 A 的第一個位置，假設 $A[i, j] = X$ ，立刻將 $A[i, j]$ 變成 Y ，並從 $j+1$ 繼續找下一個 X 直到沒有爲止。[$|A|, |B| \leq 5000$]。

1.3.1 Suffix structure

習題 1-19 最小的後綴 經典問題

給一個字串 A，求出他字典序最小的後綴。[$\mathcal{O}(|A|)$]

有不需要建出 Suffix Array 的方法。[6]

這有一些應用，如最小循環表示法。有時候我們會把一個狀態或物體用一個字串表示，並且可能旋轉需視爲相同的，也就是把字串前 k 個字元接到後面視爲相同。這時候如果要比較兩個字串，可以先求他們的最小循環表示在進行比較。

習題 1-20 重複子字串 經典問題

給一個字串 A，求出他最長的一個重複的子字串，重複的部分可以重疊。比如說 $A =$ "cababababc"，最長的重複子字串爲 "aba"。[$\mathcal{O}(\mathcal{SA})$] [7]
如果改成不能重複呢？[$\mathcal{O}(|A|\log|A|)$]

習題 1-21 連續重複的子字串 經典問題

給一個字串 A，求出他的連續重複次數最多的子字串。比如說 $A =$ "cababababc"，"ab" 在子字串 "abababab" 中連續重複了 4 次。[$\mathcal{O}(|A|\log|A|)$]

習題 1-22 Binary Suffix Array ASC 40

給定一個 n 個整數的序列 $a_1, a_2, \cdots, a_n$，找出一個 01 字串 A 使得 A 的後綴數組 $\mathcal{SA}$ 恰好是給定的序列，注意到有可能無解。[$|A| \leq 3 \cdot 10^5$]

習題 1-23 Lexicographical Substring Search SPOJ 7258

給一個字串 A，現在把 A 的所有子字串列出，將重複的刪去後排序。對於 Q 比詢問，輸出字典序第 k 小的子字串。[$|A| \leq 90000, Q \leq 500$]

習題 1-24 Beautiful Substring NTUOJ 2200

給一個字串 A，把他所有不同的子字串找出來，令這個集合爲 S，接著每次會給一個字串 B_i，假設他所有不同的子字串的集合爲 T_i，就令 $S \leftarrow S \setminus T_i$，並輸出此時的 $|S|$，也就是 S 裡還剩多少字串。[$|A| \leq 2 \times 10^5$, $\sum|B_i| \leq 10^6$]

習題 1-25 Incomparable Suffixes ASC 42

給一個字串 A，我們說兩個字串 s, t 是**可分辨的** 若且爲若存在一個字串 z 使得 sz, tz 中恰好只有一個是 A 的後綴，說兩個字串 s, t 是**無可比較的** 若且爲若對所有 s 的前綴 x，t 的前綴 y，x, y 都是可分辨的。現在請找出 $a_1, a_2, \cdots, a_n$ 滿足:

1. 所有 a_i 都是 A 的後綴。
2. n 越大越好。

[6] Hint: 雙指針
[7] $\mathcal{O}(\mathcal{SA})$ 表示建出後綴數組所需的時間。

3. 在 n 是所有可能的最大值下，$\sum|a_i|$ 越大越好。

[$|A| \leq 10^5$]

1.4 Special Thanks

此分講義部分參考自去年及前年的講義，特別感謝以前的編輯者。