**UVa11475 - Extend to Palindrome**

給你一字串，請新增字元讓這字串變成迴文，但新增字元數量要最少。

迴文：從左邊讀與從右邊讀意思都一樣

**分析：**

如果將題目給的字串顛倒做為 pattern 則 KMP 最後的 j 就會是原本的字串已經有的迴文長度，我們只需要補足就好了！

**UVa526 - String Distance and Transform Process**

給你兩個字串分別是 A,B 你可以用插入、刪除、替換，來將 A 字串轉換成為 B 字串，輸出需要幾次的轉換，也要將步驟輸出。

**UVa11151 - Longest Palindrome**

給你一個字串，詢問在這字串中能夠找到的最長迴文長度為多少。

迴文：從左邊讀與右邊讀相同

**ICPC2019\_Russia\_PJ**

有一張有向圖，有一人遍地此路徑，於是於是她得出每個點有幾種不一樣的路徑，並繪製地圖

你的任務就是將此圖還原成原始地圖，如 a -> b , c -> b

**UVa10603 - Fill**

有 3 個杯子， a , b , c ，以及目標水量 d ， 一開始只有 c 裝滿了水，其他為空，透過倒入杯子>的方式，來讓其他的杯子加滿水，試問能否達到 d ，要是不行，給出最接近數。

輸出要求：輸出最符合 d 的水量與達到 d 的最小倒水量。

**Codeforces 161D - Distance in Tree**

給你一棵樹，保證他們不會形成迴路。詢問這棵樹中有多少 x 的距離。

**UVa1207 – AGTC**

有 a , b 兩個字串求他們的 LCS(最長公共子序列) 最短修改數?

**UVa11747 - Heavy Cycle Edges**

無向圖，之後生成 Minimum Spanning Tree (最小生成樹)

**Codeforces 126B – Password**

給你一字串 X，試問有沒有一個子字串 S，在 X 的頭、尾巴、中間都出現。

，在 X 的 prefix , suffix and Beside prefix and suffix appear。

分析：

我們可以透過 Z-algorithm 找到「最長共同前綴總和 (Longest Common Prefix)」，但題目的要求字串後面也需要前綴呀，有可能共同最長前綴是 abcabcabc，但是這樣會是 6 呀，應該要是 abc、長度 3，才行的！

於是我們需要加一個 if 來找出 3，來防止我們的程式出現瑕疵，這裡用程式碼來解釋會更好解釋，逆向思考有時候會比普通的思考方式來的更好。

z[i] = max(0,min(z[i-x] , y - i + 1));

z[i] == s.length() - i -> 如果 z[i] 等於字串剩下的長度那保證會有後綴，通常 i 都會接近 s.length() 時才會符合maxn >= s.length()-i -> 再從 0 to s.length() 時勢必會找到並經過「最長共同前綴總和 (Longest Common Prefix)」，只要他比後綴還要長或相等 (還需要先滿足第一個條件，才能判斷到此條件)，就肯定代表中間已經也有經過「最長共同前綴總和 (Longest Common Prefix)」，尾巴這個並不會是第一次經過，如果尾巴是那就代表他只有兩次的共同前綴。

if(z[i] == s.length() - i && maxn >= s.length()-i ){

z[i-x] 直接詢問 z[i-x] 共同前綴嘗是多少，如果當前是在目前的共同前綴中，那理所當然現在的 i 必會等於最初的共同前綴 i-x 值，

如果不是，那必定會是 0。

y-i+1 此共同前綴理應只會有 y-i+1 個

如果 z[i-x] 比較小，代表沒有從 i 位置開始的前綴字串，否則 z[i-x] 應該要更大，所以也就表示 z[i] == z[k]。

如果 y-i+1 比較小，代表這次的共同前綴比較小，因為 z[i-x] 更大，也就代表應該有從這裡開始的共同前綴

**UVa12299 - RMQ with Shifts**

給予一個數組，有兩種操作，query l r : 查詢 l 到 r 的最小值、shift x1 , x2 , x3 : 原本的值都往前循環一個位置

**分析：**

透過線段樹來維護最小值查詢，加上 shift 的數字最多只有 30 位，透過單點修改的方式即可完成。

**UVa12538 - Version Controlled IDE**

我們需要版本控制一份文件，類型如下：

type 1 ，插入字串在 p 位置

type 2 ，移除字串從 p 位置開始移除 c 個字元

type 3 ，查詢 v 版本從 p 位置開始輸出 c 個字元

每進行一次 type 1 or 2 的動作，就增加一版本。

我們順便進行加密混淆的動作，每一次只要 type 3 輸出的字元中擁有 ‘c’，則接下來 除了 type 一外的數字都會因為前面 type 3 輸出字元中有多少的 c，則會增加多少數字。

**Uva11987 - Almost Union-Find**

並查集，有三種操作

Union: 把 x , y 加入同一集合

Move: 將 x 移動到 y 集合

Return:將 x 的集合總和，與多少元素

**分析：**

老實講，其實沒有很難，第二點比較特殊，我直接進行改點。

有個地方比較麻煩，算總和跟多少元素，原本是想要每一次都掃描陣列，發現會 TLE (廢話，但是作者太爛不知道阿~)

所以開陣列並在做 Union , Move 操作時，順便紀錄集合總和與內部元素。

**ICPC2019\_RussiaPe**

有一張地圖，有幾個需要參加 ICPC 的城市，我們需要設計一個程式，尋找一個城市可以讓每個要參加 ICPC 的城市都是花相同的距離到達?

**分析：**

先透過 DFS 把參加 ICPC 中的城市兩個最遠距離給求出來。

再將兩個城市中的最長距離 /2 ，再找出那個路經的城市(定義為 A)。

再透過 DFS 從 A 開始到每一個需要參加 ICPC 的城市，如果他們的距離不等於我們在上個 DFS 找出的最長距離 /2，就輸出 “NO” ，如果都可以那就輸出 “YES”