NTUEE Algorithm Programming Assignment #2
B10901151 電機三 林祐群

Data Structure & Basic Algorithm：

->本次 PA 最終選擇使用 Stack 這種 Abstract Data Type 來實作，取其 First In Last Out 之特性，因為在利用 Dynamic Programming 並記錄於表格後，若需要重新讀取每個弦的資料，會從表格最右上開始回溯，但輸出要求從小到大，股可以使用 SDL 的函式庫內建的功能進行實作。

->準備一個儲存輸入資料的自訂結構 chord，每個 chord 有兩個端點，分別是 a 和 b，並用 num_node 作為紀錄端點個數的媒介。

->建立長度為 num_node 的 vector，並設定初始值為 1

->比較兩個弦的"大小"，若某一弦的兩個端點之 index 值均較另一弦為小，則可做出在這個題目下的大小判斷。實作上利用一弦較大之 index 值和另一弦較小一端之 index 值做為比較基準，可以增加比較效率，即上課提及之"團結力量大"的心法。

->用兩個 int 分別記錄總共最多可以建立出之不重疊弦的個數與其 index

->建立 stack，將上述之最大值填入，並逐步回溯取值放入 stack 中，符合條件(index last 的值和目前 index 的值差 1 且符合弦的大小關係)者 push 進 stack 同時更動 index last 和目前 index 的值。

->上述環節使用 while 迴圈進行實作，有機會達到 big-O 而非 big-Θ的 time complexity，因當 dp[i]的值觸及 1 時便可以填入最後一筆資訊並終止迴圈。

->輸出階段，先輸出代表最大可能結果之 mps_size 值，後以 while 迴圈用 stack 的 empty check 作為條件，逐步 pop，同時進行比較以滿足輸出條件(依大小輸出弦的端點)。

Findings：

本次作業我原先是利用教授上課提及的三個 subproblem 來試圖利用動態規劃來解題，但當數據一大的時候卻會無法輸出正確，試圖尋找判斷式或是遞迴上的盲點無果，故決定從資料結構層面出發，簡化問題。另外在撰寫程式時，像是撰寫過程中確認結果之輸出和最終寫入檔案等等輸出輸入的過程在資料量大時會消耗大量時間，也導致在 EDA UNION 上運行時若沒有設計好會等待很久甚至超時，這讓我想到高中初次接觸程式設計時的學長給我的知識，也就是利用：

```
ios_base::sync_with_stdio(0);
cin.tie(0);
```

進行輸入優化，可以使 cin、cout 不要在緩衝區消耗太多時間，而降低效率。