Algorithm PA2 Report

b0670264 / 會計五 / 林聖硯
EDA Union: Port 40051

一、不同 case 之 runtime 以及記憶體使用量比較

Case	Runtime (ms)	Memory usage (KB)
12.in	3.998	12,500
1000.in	17.997	16,460
10000.in	1,300.8	403,532
100000.in	113,007	39,168,820

二、程式設計概念

1. 在解決這個 dynammic programming 的問題時，我採用的是上課教過的 top-down with memoization 的方法來完成 DP table(程式內部的變數名稱為"M")。從圓上的最後一個點(i.e. 2N-1)對應的弦開始尋找，會遇到以下三個 case。

CaseI: 此弦之"起點"(k)不在現在的 subset(i, j)內

=> 此弦不會算在 subset(i, j)之內，解 M[i][j-1]

CaseII: 此弦之"起點"(k)正好為 i

=> 此弦會算在 subset(i,j)內，解 M[i+1][j-1]並+1

CaseIII: 此弦之"起點"(k)在 subset(i, j)內

=> 需比較以下兩者。

(1) 這個弦"終點"前一個位置的解 (i.e. M[i][j-1])

(2) 以此弦之"起點"為畫分左半 subset 的解(M[k+1][j-1])與右半 subset 的解(M[i][k-1])之合再加一(把這條弦也算進去)，最後取大的為答案。

2. 記憶體空間之節省: 在這邊我使用了能夠自動配置記憶體的 vector library 來記錄我的 DP table(名稱: M)、以及儲存 chord 資訊的陣列(名稱: chords)，而為了不要讓以上兩者在被呼叫時自動複製，我寫了一個名為 MpsSolver 的 class，在傳入其他 object(如 fstream)時也幾乎都是採用 call by reference 的方式來執行，如此可以免除讓函數呼叫 copy constructor 的時間。而在儲存結果的部分，我初始化了一個名為 answer 的 vector(但沒有指定大小)，並不斷 push_back 得到的答案，讓 vector library 自動幫我新增空間，也藉以減少不必要的浪費(如果一開始就先開一個很大的 array 會吃掉太多記憶體)

3. 得到弦的數量以及 DP table 後，找到每個弦之頂點的演算法(函式名稱: findSolutions)也與 1.類似。由於 top-down 的 DP table 只會紀錄最好的解(於三、1.詳述)， 故在 findSolutions 內，我只要找到 M[i][j]與大於 0 與 i>j 的點，並比較

M[i][j]與 M[i][j-1]之大小來決定我要往哪個方向走。

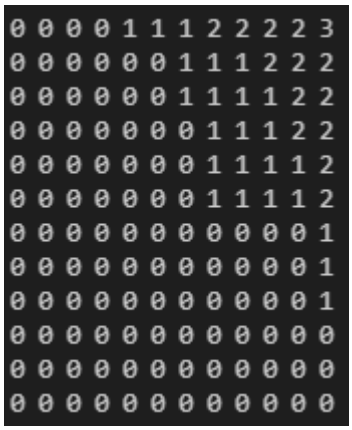
若 M[i][j] > M[i][j-1]則代表在此點上有最佳解發生，紀錄最佳解後，我應該 recursively 地找右半邊和左半邊的 subset；若 M[i][j] <= M[i][j-1]，則代表還沒有走到最佳解的弦上，我應該繼續往 i, j-1 的方向走。

三、觀察與發現

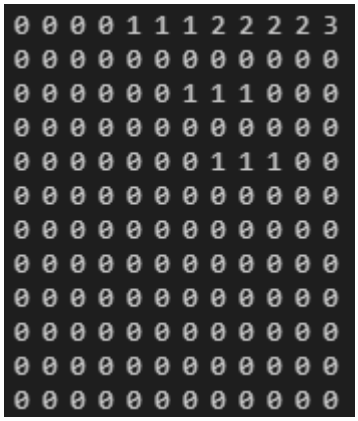
1. 我其實一開始採用的方法為 bottom-up，因為這種設計方法對我來說比較直覺與簡單，但在實作完後發現 bottom-up 的方法連 10000.in 的 case 都沒辦法在十分鐘內跑完。

仔細分析後，觀察到在這個 maximum planar subset 中，其實不需要講所有的子問題都解完，因為我們只是想要知道圓之起點(位置"0")、與圓之終點(位置"2N-1")中涵蓋最大弦的數量，不用找到任意的 M[i][j]。以下印出 12.in 此 Case 中 top-down 與 bottom-up 的 DP table，可以很明顯地發現 top-down 只記錄了最佳解所經過的弦路徑，而 bottom-up 則可以直接看出任意 M[i][j]的解，正好驗證了上課之內容——需要解完所有子問題的任務交給 bottom-up 比較有效率，不需要解完所有子問題的任務則用 top-down 的演算法比較適合。

Bottom-up



Top-Down



2. 我也寫了一個 result checker 來檢查我的答案到底正不正確，由於 output files 的弦端點會由小排到大，所以其實只需檢查"下一條弦"之尾端有沒有超出"這一條弦"之尾端即可。我將檢查程式放在 utilities 資料夾下，並且在 README.md 中說明詳細用法，只要使用 make file 就能自動 compile 出這個檢查程式。