Algorithms

Programming Assignment #2 Report

Student: B07901113 廖甜雅

Collaborator: None

我的程式簡介：

採用 top-down dynamic programming，以從 i 到 j 的 mps 作為 subproblem，分為以下 4 種 case：

1. i < j ，且 j 所在的弦的另一端點 k 不在 ij 之間：結果相當於 i 到 j-1 的 mps。
2. i<j ，且 j 所在的弦的另一端點即為 i：結果相當於 i 到 j-1 的 mps 加上弦 ij。
3. i < j ，且 j 所在的弦的另一端點 k 不在 ij 之間：計算
   A. i 到 j-1 的 mps
   B. i 到 k-1 的 mps 加上弦 kj 加上 k+1 到 j-1 的 mps
   結果為 A 與 B 中較大者。
4. i ≥ j，為方便計算將結果設為 0。（base case）

DP 過程中以 M（2n*2n 大小的 vector< vector<int> >）紀錄從 i 到 j 中 mps 的 chords 數。

$$M[i][j] = \begin{cases} 0, & i \geq j; \\ M[i][j-1], & i < j, k < i \ or \ k > j; \\ M[i][j-1] + 1, & i < j, k == i; \\ \max\left(M[i][j-1], M[i][k-1] + 1 + M[k+1][j-1]\right), & i < k < j \end{cases}$$

並以 S（2n*2n 大小的 vector< vector<int> >）紀錄 case 2 與 case 3-B 的發生處。

$$S[i][j] = \begin{cases} -1, & i \geq j; \\ S[i][j-1], & i < j, k < i \ or \ k > j; \\ S[i][j-1] - 2n - 1, & i < j, k == i; \\ S[i][j-1], & i < k < j, M[i][j-1] > M[i][k-1] + 1 + M[k+1][j-1] \\ j, & i < k < j, M[i][j-1] \leq M[i][k-1] + 1 + M[k+1][j-1] \end{cases}$$

除了 Case 3-B，遞迴過程中 i 並不會發生變化。因此如果 S[i][j] ≥ 0，Case 3-B 即發生在(i, S[i][j])。如果 S[i][j] == -1，則為 base case。如果 S[i][j] < -1，代表 Case 2 發生於(i, i 所在弦的另一端點)，Case 3-B 發生在(i, S[i][j]+2n+1)。

- main(): 讀入 input，並存成一個長度為端點總數=2n 的 vector ep，ep 中 key 對應到的 value 就是該 key 所在 chord 的另一個端點。如此可以快速取得 k。呼叫 get_mps()取得 output 後，再按格式輸出。
- get_mps(): 呼叫 fill_table 以 top-down 的方式填寫 M 與 S，並呼叫 load_index() 根據 S 得出 maximum planner subset 的較小端點給 main()。
- fill_table(): 以 recursive 方式填寫 M 與 S，詳見上方描述。回傳 M[i][j]。

- write(): 將指定值填寫在 M[i][j]與 S[i][j]。回傳 M[i][j]。
- load_index(): 以 recursive 根據 S 得出 maximum planner subset 的較小端點，存在 mps_index。詳見上方描述。

心得：

- 先想清楚需要的演算法，留意這個演算法需要對資料做什麼事，就比較容易找到輸入輸出適合的資料結構。
- 對於 input 資料量比較大、且 subproblems 只佔所有較小 case 的小部分的情況下，使用 top-down 會比 botton-up 節省很多時間。