

[Algorithms] Programming Assignment #2

Data Structure

- chords_map: 用 map 存所有連線相對應的 endpoint，例如 (2,5) 為一連線的兩端，chords_map = { {2, 5}, {5, 2} }，因為 map 能做一對一的 mapping，如此一來用 chords_map[2] 就可以知道 2 跟誰相連。（原本一開始思考想說 map 可以空間用量變少，後來發現其實還是跟 vector 的儲存方式一樣）
- mps_matrix: $2n \times 2n$ 大小的矩陣（用 vector<vector<int>> 去實現），用來存不同 sub problem 的 optimal solution，這樣遇到 overlapping problem 可以直接拿答案
- bool_matrix: $2n \times 2n$ 大小的矩陣，bool_matrix[i, j] 紀錄在計算 i 到 j 之間的 max planar subset 時，(i, j) 是不是其中一組解，用來回推最後的答案
- subset: 用來存解出來的 subset

Algorithms

- Dynamic programming for mps: 我使用了 top-down 的方式去做 mps（會比 bottom-up 快），分成 3 個 case 跟 1 個 base case。（k 是跟 j 相連的點）
 - Base case (if $i > j$): $M(i, j) = 0$
 - Case 1 ($k > j$): $M(i, j) = M(i, j - 1)$
 - Case 2 ($i < k < j$):
$$M(i, j) = \max(M(i, j - 1), M(i, k - 1) + 1 + M(k + 1, j - 1))$$
 - Case 3 ($k = i$): $M(i, j) = M(i + 1, j - 1) + 1$
- Get the subset list by divide and conquer: case 基本上跟 mps 一樣，配合 bool_matrix 把 [i, j] 一點一點切割去找是不是解
 - Case 1 ($k > j$): recursive (i, j-1)
 - Case 2 ($i < k < j$): push {k, j} to subset，然後接著 recursive (i, k-1) 跟 (k+1, j-1)
 - Case 3 ($k = i$): push {k, j} to subset，然後接著 recursive (i+1, j-1)