B07901103 電機三 陳孟宏

1. main.cpp

- 先設定一個help_message(), 做讀檔失敗的error handling
- 開始讀檔, 每行getline (因為測資最多好像到18萬, 所以我每行只開15個buffer空間)
- 初始 1-D array叫做chord_array, 按照input檔建立j連到k, k連到j
- 執行函數MaximumPlanarSubset()找到最大弦數量M
- 用which_chord(), 去找出是哪幾條弦構成最大弦數量
- 按照弦jk的node順序去sort
- 按照output格式輸出

2. maxPlanarsubset.cpp

- MaximumPlanarSubset(int*, int):
  - int*: 吃進chord_array, 已經建好的jk弦資訊
  - int: 吃進node_num, 也就是這個圓上有幾個點
  - 首先先建立一個double array M (按照手寫作業2方法), 初始 M[i][i] = 0
  - 開始跑recursion, (bottom up方法), 分成3種情況:
    - k不在[i, j]中, 則往內縮M[i][j] = M[i][j-1]
    - k在(i, j)中, 有兩種情況要取max, M[i][j-1] < (M[i][k-1]+1+M[k+1][j-1]) 或是 M[i][j-1] >= (M[i][k-1]+1+M[k+1][j-1]), 其中前者會確定一條弦jk一定構成最大弦數量, 所以先把他mark起來等等需要用, 因為 i <= j, 所以M[j][i]空間不會用到, 我令M[j][i]此時等於2 (case2)
    -
    - k就是i, M[i][j] = M[i+1][j-1]+1, 這時候可以確定有一條弦構成最大弦數量, 也就是ij, 所以先把他mark起來等等需要用, 因為i <= j, 所以M[j][i]空間不會用到, 我令M[j][i]此時等於3 (case3)
- which_chord(int i, int j, int**M, int* chord_array, vector<int>& optimal_chord)
  - optimal_chord: 拿來存構成最大弦數量的弦
  - 在main檔, 所求為M[0][N-1], 所以這裡i=0, j=N-1
  - 能夠繼續執行條件: j還沒小於i的弦都可能是一組解
  - 接續上個函數有mark的弦 (確定是一組解), case3比較簡單, 把ij弦加進optimal_chord後, 繼續recursion往內找

    - case2因為k在(i, j)中, 所以分為左半和右半各自recursion來分別找出有哪些弦
    - 剩下case既然還無法確定是哪些弦就繼續往內recursion

3. maxPlanarSubset.h

- 建構一個class叫做"MPS", construct並定義了兩個maxPlanarSubset函數, MaximumPlanarSubset(int*, int), 以及which_chord(int i, int j, int**M, int* chord_array, vector<int>& optimal_chord)

4. Findings

- 到大筆測資常常會被kill掉, 代表我有地方一直吃太多空間, 所以能省地方該省, 像是我原本用2-D vector (vector<vector<int> >)存M, 但爆掉, 所以改用空間較小的int **M, 還有時間跑太久, 我原本在which_chord找哪些弦時, 判斷式用 if (M[i][j-1] < (M[i][k-1]+1+M[k+1][j-1])) 來代表case2, 但這樣又要重跑recursion耗時間, 所以我乾脆在MaximumPlanarSubset()直接mark