Programming Assignment #2 Report					
姓名	劉又慈	學號	B11901193	系級	電機三

1. Data structure

How do you store the data of chords and/or other supporting information?

使用一個 vector 名稱為 C，C 的大小設定為 vertices 的數量並且每一格的元素初始化為-1，當讀到一個 chord 時，會將 chord 的開頭 vertice 存入 C[chord 的結尾 vertice]當中，chord 的結尾 vertice 存入 C[chord 的開頭 vertice]當中。此做法是為了之後可以快速判斷從 j 連線的 chord 有沒有在[i, j]的範圍裡。

2. Algorithm

- **Find the number of chords**

使用 Top Down Dynamic Programming，recurrence equation 如下：

$$\mathrm{M(i,j)},$$
$$i \le j, \textit{denote the number of chords in the maximum planer subset}$$

$$\mathrm{M(i,j)} = \mathrm{M(i,j-1)}$$
$$\text{if chord kj} \in \mathrm{C}, \mathrm{k} \notin [\mathrm{i,j}]$$

$$\mathrm{M(i,j)} = max\{\mathrm{M(i,j-1)}, \mathrm{M}(i, k-1) + 1 + \mathrm{M}(k+1, j-1)\}$$
$$\text{if chord kj} \in \mathrm{C}, \mathrm{k} \in [\mathrm{i,j}]$$

$$\mathrm{M(i,j)} = \mathrm{M(i+1,j-1)} + 1$$
$$\text{if chord } ij \in \mathrm{C}$$

由於進行 Dynamic Programming 時，如果建立一個 M[vertice][vertice]的二維陣列將只會用到陣列右上方的部分(因為要$i \le j$)，因此在本次程式作業中我使用一維的 vector M[(vertice) * (vertice + 1) / 2]，並建立以下 index 轉換關係式，讓二維陣列中的每一個位置都能對應到 M 的其中一個 index。

```
// Function to convert (i, j) indices to 1D index in upper triangular matrix
long long get1DIndex(int i, int j, int vertice) {
    return static_cast<long long>(i) * vertice - (static_cast<long long>(i) * (i + 1)) / 2 + (j - i);
}
```

藉由以上 recurrence relation 建構 topdown 的判斷關係式以完成尋找 number of chords。

實作程式碼：

```
long long topdown(vector<long long>& M, const vector<int>& C, int i, int j, int vertice) {
    if (i >= j) return 0;

    long long index = get1DIndex(i, j, vertice);
    if (M[index] != 0) return M[index];

    int k = C[j];
    if (k < i || k > j) {
        M[index] = topdown(M, C, i, j - 1, vertice);
    }
    else if (k == i) {
        M[index] = topdown(M, C, i + 1, j - 1, vertice) + 1;
    }
    else {
        M[index] = max(topdown(M, C, i, j - 1, vertice), topdown(M, C, i, k - 1, vertice) + topdown(M, C, k + 1, j, vertice) + 1);
    }

    return M[index];
}
```

- **Find the chords themselves**

藉由 function traceback 逐步回推 maximum planar subset 使用了哪些 chord，traceback 的操作流程：

(1) Base Case（if $i \geq j$ return）：

當i大於或等於j時，表示此範圍無效，無法包含任何 chord，因此直接退出遞迴。

(2) 找出 chord 的端點（int $k = C[j]$）：

$C[j]$表示以 j 為結束點的弦的起始點 k。如果 k 為 -1，代表在 j 位置沒有弦，則跳過此位置。

(3) 判斷條件：

- Case 1：如果$k < i$或$k > -1$，表示$[i,j]$內不包含此$chord\ kj$，因此我們查看 M 的前一個值：$M[i][j-1]$，藉由呼叫 $traceback(M, C, i, j\ -\ 1, result, vertice)$。

- Case 2：如果$chord\ ij$直接連接 i 和 j（$k\ ==\ i$），則將此弦加入 result 中。然後，遞迴呼叫$traceback(M, C, i\ +\ 1, j\ -\ 1, result, vertice)$ 移至內部範圍進行下一次的判斷。

- Case 3：如果k位於$[i,j]$內但不等於i，則需要判斷若$M[index]\ ==\ M[index_j_minus_1]$，則$chord\ kj$不屬於需要選取的弦之一，因此遞迴呼叫$traceback(M, C, i, j\ -\ 1, result, vertice)$。反之，則$chord\ kj$是其中一條要選的弦，將此弦加入 result 中，並呼叫 $traceback(M, C, i, k\ -\ 1, result, vertice)$和 $traceback(M, C, k+1, j, result, vertice)$，繼續判斷由$chord\ kj$分隔的兩個區域中還有選哪些弦。

實作程式碼：

```cpp
// Recursive function to trace back the solution
void traceback(const vector<long long>& M, const vector<int>& C, int i, int j, set<pair<int, int>>& result, int vertice) {
    if (i >= j) return; // Base case: no chords in this range

    int k = C[j];
    long long index = get1DIndex(i, j, vertice);

    if (k < i || k > j) {
        // No chord for `j`, so move to the previous position
        traceback(M, C, i, j - 1, result, vertice);
    }
    else if (k == i) {
        // Chord between i and j is part of the subset
        result.insert({i, j});
        traceback(M, C, i + 1, j - 1, result, vertice); // Move inside
    }
    else {
        // Decision point: choose the maximum subset
        long long index_j_minus_1 = get1DIndex(i, j - 1, vertice);
        if (M[index] == M[index_j_minus_1]) {
            traceback(M, C, i, j - 1, result, vertice);
        }
        else {
            result.insert({k, j});
            traceback(M, C, i, k - 1, result, vertice);
            traceback(M, C, k + 1, j, result, vertice);
        }
    }
}
```

3. Time Complexity Analysis

- Time complexity of dynamic programming $O(n^2)$：M 的大小為 $(2n * (2n + 1))/2 = 2n^2 + n$，因此以 top down 的方式將 M 的格子填滿數字會需要$O(n^2)$。
- Time complexity of traceback $\Theta(n)$：每一個 chord 只會被尋訪一次。

Overall time complexity = $O(n^2)$