report

psuedo code

```
 let M[i][j] denotes the number of chords in the maximum planar subset form by chord ij
let R[i][j] denotes the case of operation on M[i][j] (see more explanation below)

trace_back_M(i, j) {
    if M[i][j] is not empty, return
    if (i >= j) M[i][j] = 0, return
    k = endpoints connect to j // kj is in C
    if k < i or k > j // CASE 1
        trace_back_M(i, j-1)
        M[i][j] = M[i][j-1]
    else if k == i // CASE 2
        trace_back_M(i, j-1)
        M[i][j] = M[i][j-1]+1
        R[i][j] = 1
    else // CASE 3
        trace_back_M(i, j-1)
        trace_back_M(i, k-1)
        trace_back_M(k+1, j-1)
        if M[i][k-1] + M[k+1][j-1] + 1 > M[i][j]
            M[i][j] = M[i][k-1] + M[k+1][j-1] + 1
            R[i][j] = 2
        else M[i][j] = M[i][j-1]
}

trace_back_R(i, j, list) {
    if (i >= j) return
    k = endpoints connect to j // kj is in C
    switch(R[i][j])
        case 0: trace_back_R(i, j-1) // default case
        case 1:
            list.push(i)
            trace_back_R(i, j-1)
        case 2:
            list.push(k)
            trace_back_R(i, k-1)
            trace_back_R(k+1, j-1)

}
```

解題concept

使用top-down的方式 解題分成step1 找出最大弦數並紀錄case，以及step2 根據case找出在最大弦數中弦的集合

1. 找出最大弦數 定義M[i][j]為在ij區間中最大弦數(定義i < j)。透過DP substructure分成三個case (k為j對應的點)。

    - CASE 1：k 不在ij區間，M[i][j] = M[i][j-1]
    - CASE 2：k 等於 i，M[i][j] = M[i][j-1] + 1
    - CASE 3：k 在ij區間內且不等於i，M[i][j] = max((M[i][k-1] + M[k+1][j-1] + 1), M[i][j-1])

    試求M[0][N-1]。利用top-down與memory的方式，若會使用到M[i][j-1]，則先call trace_back_M(i, j-1)已確保M[i][j-1]已經被計算，就算曾經計算過也可以透過M的取值來避免重複計算，接著就可以透過遞迴的方式來快速得到M[0][N-1]。

2. 找出在最大弦數中弦的集合 定義R[i][j]為ij區間在對M[i][j]操作的類別。透過trace back的方式，從R[0][N-1]開始

    - case 0：k 不在ij區間，沒有增加弦，繼續trace back R[i][j-1]
    - case 1：k = i，在list中增加chord ij，並繼續trace back R[i][j-1]
    - case 2：是step 1中case 3的其中一個可能，M[i][k-1] + M[k+1][j-1] + 1 > M[i][j-1]，在list中增加chord kj，並trace back R[i][k-1] 以及 R[k+1][j-1]

    整體來說很像是step 1的三個case，只差在step 1 case 3可能會對應到step 2 case 1或是case 3. P.S. implement時我只會紀錄小的點，接著在輸出前將list排序，印出時才找出其對應的點。

資料結構

- 由於每個點都是數字，利用陣列的index與value兩兩對應的關係來儲存chord。 例如chord(1, 10)，chord_set[1] = 10, chord_set[10] = 1，這樣就可以在O(1)的時

間找到對應點。

- M與R使用new來宣告動態array，會比使用vector快，因為會直接allocate適當大小給process，若使用vector則會慢慢double capacity直到夠用。由於已知最大的case是100000，M使用short，R使用u_int8