演算法 PA2

電機二　B12901075 賴禹衡

一、資料結構

此次作業我在手寫 HW2 時就有想過，一開始儲存 chord 可以用一維 vector<int>來儲存，使得 chord ij 的資訊能同時存在 data[i] = j、data[j] = i 之中，接著是主要的 chord 數的資訊，我存在一個叫做 M 的 vector<vector<int>>之中，也是與 HW2 最後一題的思路一樣，接著是要怎麼叫出 parent 點，這部分我下面的演算法再詳細解釋，原本使用一個 vector<int>來紀錄 parent 的相關資訊，但因為我最後放棄使用 vector<int>來紀錄 parent 的資訊，所以在程式中沒有看到。

二、演算法

最一開始我使用的方法非常直接，就是 bottom up 的同時記錄 M 與 C（如圖一），其中 C 代表了選擇的 chord，但是在 compile 測試時，發現這樣做會使用過多的 memory 來儲存，因此我把 C 改成 vector<vector<char>>來儲存遇到的 case ，並在最後使用遞迴來處理選出來的 chord，（如圖二、三），然而在上傳到 eda union 後，針對 100000.in 一直跑不出結果（被中斷連線），我推測是耗時或記憶體過久，因此我上 FB(NTUEE algorithm)社團查詢看有沒有

人也遇到一樣的問題，發現前幾年的助教有給過一個提示，是關於 bottom up 跟 top down 哪個比較節省的提示，因此我就想到可以用跟紀錄 parent 一樣的方法來處理 M，因此便把演算法部分也改寫成 top down 形式，來試試看是否會成功，最後總算能成功跑出結果，最後 M 的處理就改成如圖四那樣。

<table>
<tr>
<td>

```
for (int l = 1; l < n; ++l) {
        for (int i = 0; i + l < n; ++i) {
            int j = i + l;
            int k = data[j];
            if (k < i || k > j) {
                M[i][j] = M[i][j-1];
                C[i][j] = C[i][j-1];
            }
            else if (k == i) {
                if (i + 1 <= j - 1) {
                    M[i][j] = M[i+1][j-1] + 1;
                    C[i][j] = C[i+1][j-1];
                } else {
                    M[i][j] = 1;
                }
                C[i][j].push_back({i, j});
            }

            else {
                int a = M[i][j-1];
                int b = M[i][k-1] + 1 + M[k+1][j-1];
                }
                if (b > a) {
                    M[i][j] = with_j;
                    C[i][j] = C[i][k-1];
                    C[i][j].push_back({k, j});
                } else {
                    M[i][j] = a;
                    C[i][j] = C[i][j-1];
                }
            }
        }
    }
```

</td>
<td>

```
vector<vector<char>> parent(n, vector<char>(n, 1));
for(int l =1; l<=n-1; l++){
    for (int i =0; i<=n-1-l; i++) {
        int j = i+l;
        int k = data[j];

        if( (k<i || k>j)){
            M[i][j] = M[i][j-1];
            parent[i][j] = 1;

        }
        else if (k == i) {
            if (i + 1 <= j - 1) {
                M[i][j] = M[i+1][j-1] + 1;
            }
            else {
               M[i][j] = 1;
            }
            parent[i][j] = 2;
        }

        else {
            int a = M[i][j-1];
            int b = M[i][k - 1] + M[k + 1][j - 1] + 1;
            if(k + 1 <= j - 1) {
                b += M[k + 1][j - 1];
            }
            if (b > a) {
                M[i][j] = b;
                parent[i][j] = 3;
            }
            else {
                M[i][j] = a;
                parent[i][j] = 1;
            }
        }

        }
    }
```

</td>
</tr>
<tr>
<td>圖一</td>
<td>圖二</td>
</tr>
</table>

```
void buildResult(int i, int j,const vector<int>& data
    if (i > j) return;

    char dpcase = parent[i][j];
    if (dpcase == 1) {
        buildResult(i, j - 1, data, parent, result);
    }
    else if (dpcase == 2) {
        result.emplace_back(i, j);
        buildResult(i + 1, j - 1, data, parent, resul
    }
    else if (dpcase == 3) {
        int k = data[j];
        result.emplace_back(k, j);
        buildResult(i, k - 1, data, parent, result);
        buildResult(k + 1, j - 1, data, parent, resul
    }
}
```

圖三

```
int buildM(int i, int j, vector<int>& data,vector<vector<int>>& M){
    if(i > j) return 0;
    if(M[i][j] != -1) return M[i][j];
    int k = data[j];
    if(k < i || k > j)
    {
        M[i][j] = buildM(i, j-1, data, M);
    }else if(k == i)
    {
        M[i][j] = buildM(i+1, j-1, data, M) + 1;
    }else //case 2
    {
        if(buildM(i, j-1, data , M) < buildM(i, k-1, data, M) + buildM(k+1, j-1, data, M) + 1)
        {
            M[i][j] = buildM(i, k-1,data, M) + buildM(k+1, j-1, data, M) + 1;
        }else
        {
            M[i][j] = buildM(i, j-1, data, M);
        }
    }
    return M[i][j];
}
```

圖四

三、time & memory cost

最後附上我在 EDA Union 上的測試結果

	time	memory
12.in	0.153ms	6088KB
1000.in	16.226ms	10048KB
10000.in	764.075ms	397272KB
100000.in	152506ms	39162772KB

四、複雜度分析

程式主要部分是 buildM 的填表，可以得知最差情況下為 O(n^2)，BuildResult 的部分也是一樣，整體遞回跟 M 差不多也是 O(n^2)，剩下大多都是 O(n)或 O(nlgn)，因此此演算法的整體時間複雜度為 O(n^2)