Programming homework 2

Wan-Cyuan Fan

- Data structure

```
private:
   int _num;               //store the number of the chords

   int _num_point;

   map<int_pair,int> _result_tab;

   map<int,int> _chord_tab;

   vector<int> _k_array;         //store the j,k endpoints
};
```

利用map結構儲存：

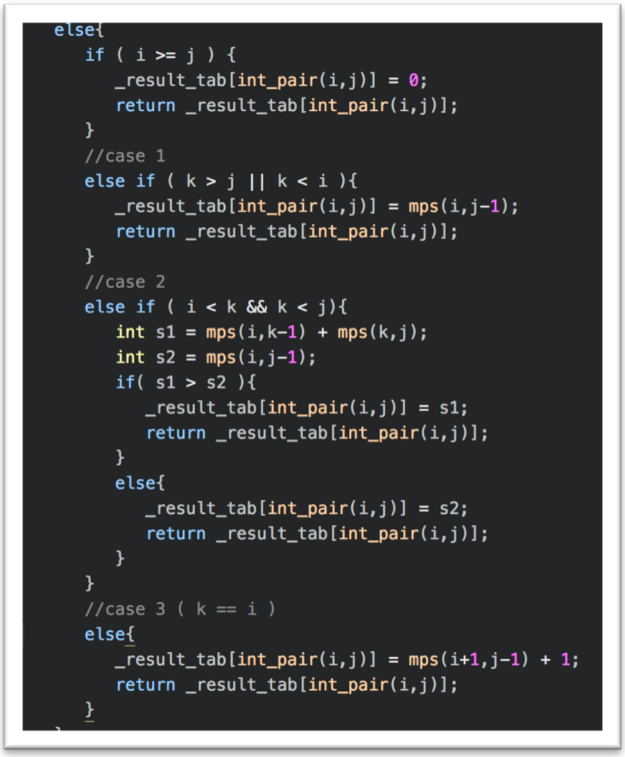
1. 先將所有input給的弦組合，放到_chord_tab中，做map的一對一對應。
2. result_tab，先typedef int_pair用來存取<int,int>的pair，再將pair和int做mapping，存取在i,j為範圍的弧中，最大的不相交弦數量。
3. _*k*_array用來存取最大不相交弦的弦端點

- Algorithm

Part 1:先建立好_result_tab：

```
int
mpsMgr::mps( int i , int j)
{
   int k = 0;
   vector<int> tem_k_array,tem_k_array_s1,tem_k_array_s2;
   k = _chord_tab[j];
   map<int_pair,int>::iterator it;
   it = _result_tab.find(int_pair(i,j));
   if( it != _result_tab.end()){
      return _result_tab[int_pair(i,j)];
   }
```

1. 先存取k值，然後在地回的過程中，尋找i,j對應到的範圍值，是不是已經在_result_tab中可以找到，如果可以找到就直接取值就好，避免重複計算。



```
else{
    if ( i >= j ) {
        _result_tab[int_pair(i,j)] = 0;
        return _result_tab[int_pair(i,j)];
    }
    //case 1
    else if ( k > j || k < i ){
        _result_tab[int_pair(i,j)] = mps(i,j-1);
        return _result_tab[int_pair(i,j)];
    }
    //case 2
    else if ( i < k && k < j){
        int s1 = mps(i,k-1) + mps(k,j);
        int s2 = mps(i,j-1);
        if( s1 > s2 ){
            _result_tab[int_pair(i,j)] = s1;
            return _result_tab[int_pair(i,j)];
        }
        else{
            _result_tab[int_pair(i,j)] = s2;
            return _result_tab[int_pair(i,j)];
        }
    }
    //case 3 ( k == i )
    else{
        _result_tab[int_pair(i,j)] = mps(i+1,j-1) + 1;
        return _result_tab[int_pair(i,j)];
    }
```

2. 如果找不到才做遞迴，這裡分成三種case:(trivial 情況直接回傳0)
    a. case 1: k > j || k < i , 直接計算mps(i,j-1)
    b. case 2: i < k && k < j , 這種情況要分開計算，看mps(i,k-1) + mps(k,j)還有mps(i,j-1)哪一個比較大，我們紀錄較大的那個。
    c. case 3: k == i , 遞迴mps(i+1,j-1) + 1

3. 用以上演算方法找出_result_tab之對應關係。

Part 2: 開始建構_k_array:

```
mpsMgr::mps_k( int i , int j)
{
   int k = 0;
   //cout << " i = " << i << ", j = "<< j << endl;
   k = _chord_tab[j];
   //cout << " k = " << k << endl;
   if ( j > i) {
      //case 1
      if ( k > j || k < i ){
         mps_k(i,j-1);
      }
      //case 2
      else if ( i < k && k < j){
         int s1 = _result_tab[int_pair(i,k-1)] + _result_tab[int_pair(k,j)];
         int s2 = _result_tab[int_pair(i,j-1)];
         if( s1 > s2 ){
            mps_k(i,k-1);
            mps_k(k,j);
         }
         else{
            mps_k(i,j-1);
         }
      }
      //case 3 ( k == i )
      else{
         _k_array.push_back(k);
         mps_k(i+1,j-1);
      }
```

1. 建構方法和part 1基本上一樣，只是這次遞迴就是查表，每一次都找最大的那個可能i,j對應，然後存入k值。
2. 只有在第三種情況發生的時候，我們存入k值。

- Discussion

計算演算法複雜度：

演算法的兩個部分，利用mps function建構_result_tab，以及利用mps_k function來找出對應個弦端點。

1. 在建構_result_tab的過程中，利用遞迴找尋，每次都會記錄對應的int_pair使其Top_down的方法不會重複計算，因此最worst的情況就是將整個int_pair由i=0,j=0一直填滿到i=0,j=2N-1。

   For(j = 0 , j < 2N-1 ,j++)

   For(i = 0, i <= j ,i++)

   故時間複雜度為O(N^2)

2. 而在建立mps_k的時候，因為_result_tab已經填寫完畢，我們只需要取直就好，故將0~2N-1的所有end point走過一遍即可，時間複雜度為O(N)。