headFile.h

typedef struct node

{

Int serialNumber; // serial number, 即大學的序列號

Int studentNumber; // serial number correspond student number, 搜索結果數量

}StudentNumberNode

定義了一個結構體數組，存放學校序列號以及該學校搜索结果統計数量

int AdjustMinimumHeap(StudentNumberNode \*, int, int); // size is the array size, i represents the position in the array 最小堆單個子樹調整函數

int AdjustMaximumHeap(StudentNumberNode \*, int, int); // size is the array size, i represents the position in the array 最大堆單個子樹调整函数

void AdjustDoubleEndedHeap(StudentNumberNode \*, int, int); // i represents the position in the array, size means the double ended heap actually contains node number

// Binomial Tree單個子樹調整函數

void BuildDoubleEndedHeap(StudentNumberNode \*, int); // build double ended heap function, 、、// 利用現有數組構建 Binomial Tree

int InsertDoubleEndedHeap(StudentNumberNode \*, StudentNumberNode, int); // size is the array[] actually size // 向當前Binomial Tree中插入新節點

int AdjustDoubleEndedHeap(StudentNumberNode \*array, int i, int size) // i represents the position in the array, size means the double ended heap actually contains node number

傳入的參數：

StudentNumberNode \* ：元素為 StudentNumberNode 的數組首地址

Int i ：當前需要調整的節點序號

Int size ：當前Heap中節點數量，從 1 開始計數

關鍵數學公式

int layer = (int)( log(i + 1) / log(2) ); // caculate which layer of the node is currently operating, round down，計算當前待排序節點處在binomial tree中第幾層，從0層開始計數

int distanceOfBrothers = (int)pow(2, layer - 1); // 計算當前待排序節點與其兄弟節點位置上的距離

int boundryOfMinimum\_MaximumHeap = (int)3 \* pow(2, layer - 1) - 2 ; // to judge i position in minimum heap of maximum heap // 計算當前待排序節點所在層中，min-max heap的分界位置，此為當前層min-heap最後一個元素位置

之後對Binomial tree的處理過程和老師說的一樣。只是一些邏輯處理，沒有太多數學思維。

void BuildDoubleEndedHeap(StudentNumberNode \*array, int size) // build double ended heap function

傳入的參數

StudentNumberNode \* ：元素為 StudentNumberNode 的數組首地址

Int size ：當前Heap中節點數量，從 1 開始計數

從bottom節點開始，直到root node of min-heap，迴圈調用binomial tree調整函數AdjustDoubleEndedHeap(), 即可構建一棵 Binomial tree

void InsertDoubleEndedHeap(StudentNumberNode \*array, StudentNumberNode insertElement, int size) // size is the array[] actually size

傳入的參數

StudentNumberNode \* ：元素為 StudentNumberNode 的數組首地址

StudentNumberNode insertElement：待插入的新節點

Int size ：當前Heap中節點數量，從 1 開始計數

因為插入前已經是一棵Binomial tree，所以只要調整新插入的節點，調整完成後就是新的一棵Binomial tree。 調整這個步驟只需要做一次，即 AdjustDoubleEndedHeap()函數只用調用一次就好。

AdjustMaximumHeap( StudentNumberNode \*array, int I, int size);

參數含義

StudentNumberNode \* ：元素為 StudentNumberNode 的數組首地址

Int i ：當前需要調整的節點序號

Int size ：當前Heap中節點數量，從 1 開始計數

Int max; 父節點，左右孩子節點三者中，所存儲的值最大的節點序號

該函數是建 MaximumHeap 最核心的函數。我的 MaximumHeap 是從 0 開始計數的，即 ：

樹根為序號 0，

左孩子序號 = 父節點序號 \* 2 + 1，

右孩子序號 = 父節點序號 \* 2 + 2，

父節點序號 = ( 孩子節點序號 – 1 ) / 2。

函數邏輯

1. 判斷當前需要調整的節點是否為非葉子節點，由於葉子節點沒有左右孩子，所以當前需要调整的節點是葉子節點的話，則不進行任何操作。函數直接完成
2. 判斷當前需要調整的節點是否有左孩子，即左孩子節點序號是否大於當前Heap 最後一個元素序號。同時判斷左孩子節點中存儲的值是否大於序號為 max 節點中存儲的值，若大於，則將左孩子節點的序號賦給 max，若不，則不進行操作。
3. 判斷當前需要調整的節點是否有右孩子，即右孩子節點序號是否大於當前Heap 最後一個元素序號。同時判斷右孩子節點中存儲的值是否大於序號為 max 節點中存儲的值，若大於，則將右孩子節點的序號賦給 max，若不，則不進行操作。
   1. 若父節點，左右孩子節點三者中，所存储的最大的節點序號 max 不等於父節點序號時，即左右孩子中至少有一個孩子節點中存储的值大於父節點中存儲的值，將左右孩子節點中最大的節點，與父節點進行交換。交換完成後，為了避免以新的孩子節點（之前為父節點）為父節點的子樹不滿足最大堆的需求，所以重複調用 AdjustHeap() 函數若父節點，使得以新的孩子節點（之前為父節點）為父節點的子樹同樣滿足最大堆的需求。
   2. 若以左右孩子節點三者中，所存储的最大的節點序號 max 等於父節點序號時，不進行任何操作。

AdjustMinimumHeap( StudentNumberNode \*array, int I, int size);

參數含義

StudentNumberNode \* ：元素為 StudentNumberNode 的數組首地址

Int i ：當前需要調整的節點序號

Int size ：當前Heap中節點數量，從 1 開始計數

Int min; 父節點，左右孩子節點三者中，所存儲的值最大的節點序號

該函數是建 MinimumHeap 最核心的函數。我的 MinimumHeap 是從 0 開始計數的，即 ：

樹根為序號 0，

左孩子序號 = 父節點序號 \* 2 + 1，

右孩子序號 = 父節點序號 \* 2 + 2，

父節點序號 = ( 孩子節點序號 – 1 ) / 2。

函數邏輯

1. 判斷當前需要調整的節點是否為非葉子節點，由於葉子節點沒有左右孩子，所以當前需要调整的節點是葉子節點的話，則不進行任何操作。函數直接完成
2. 判斷當前需要調整的節點是否有左孩子，即左孩子節點序號是否小於當前Heap 最後一個元素序號。同時判斷左孩子節點中存儲的值是否小於序號為 min 節點中存儲的值，若小於，則將左孩子節點的序號賦給 min，若不，則不進行操作。
3. 判斷當前需要調整的節點是否有右孩子，即右孩子節點序號是否小於當前Heap 最後一個元素序號。同時判斷右孩子節點中存儲的值是否小於序號為 min 節點中存儲的值，若小於，則將右孩子節點的序號賦給 min，若不，則不進行操作。
   1. 若父節點，左右孩子節點三者中，所存储的小於的節點序號 min 不等於父節點序號時，即左右孩子中至少有一個孩子節點中存储的值小於父節點中存儲的值，將左右孩子節點中最大的節點，與父節點進行交換。交換完成後，為了避免以新的孩子節點（之前為父節點）為父節點的子樹不滿足最大堆的需求，所以重複調用 AdjustMinimumHeap() 函數若父節點，使得以新的孩子節點（之前為父節點）為父節點的子樹同樣滿足最大堆的需求。
   2. 若以左右孩子節點三者中，所存储的最大的節點序號 min 等於父節點序號時，不進行任何操作。