HW1

Profit Maximization Problem by recursive and non-recursive version program

B10532011 高靖雅

2019/03/06

Recursive：

程式摘要：

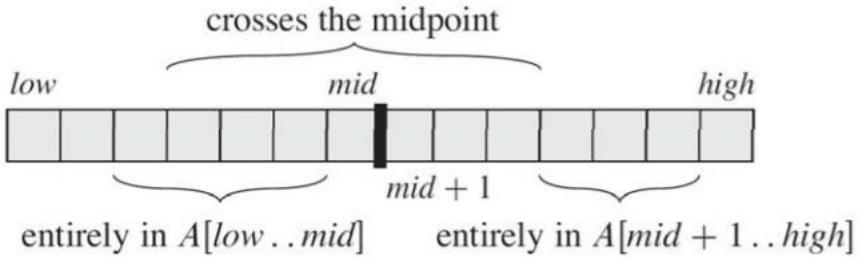
此程式用 C++語言

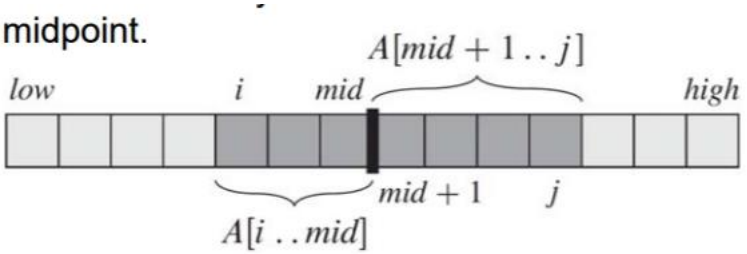
用 struct 存取 Subarray 中的三個值，起始、結束、總和

用 Divide and Conquer 方法做遞迴運算，把原本的 array 分成數個 subarray，再藉由 subarrays 間，互相比較取的的最大區間，從而得出整個 array 的最大區間

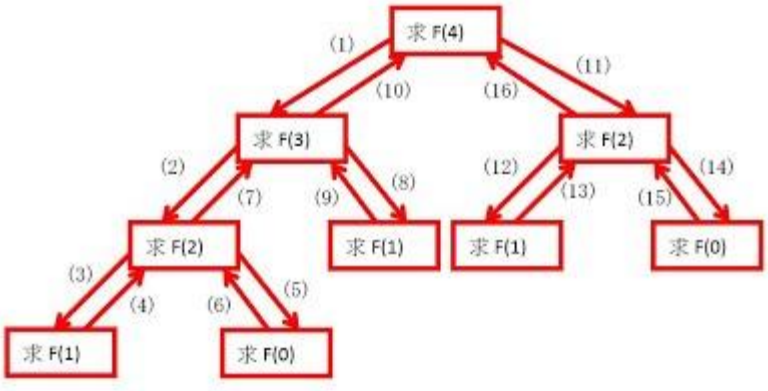
程式內容說明：

圖解:





– We just need to find maximum subarrays of the form *A*[*i* .. *mid*] and *A*[*mid* + 1 .. *j*] and then combine them.



虛擬碼:

```
1 Find_Max_Crossing_Subarray(A, low, mid, high)
2     left_sum = -∞
3     sum = 0
4     for i = mid to low
5         sum = sum + A[i]
6         if sum > left_sum
7             left_sum = sum
8             max_left = i
9     right_sum = -∞
10    sum = 0
11    for j = mid + 1 to high
12        sum = sum + A[j]
13        if sum > right_sum
14            right_sum = sum
15            max_right = j
16    return (max_left, max_right, left_sum + right_sum)

1 Find_Maxium_Subarray( A, low, high)
2     if high == low
3         return (low, high, A[low])
4     else mid = (low + high)/2
5         (left_low, left_high, left_sum) =
              Find_Maxium_Subarray(A, low, mid)
6         (right_low, cross_high, cross_sum) =
              Find_Maxium_Subarray(A, mid+1, high)
7         (cross_low, cross_high, cross_sum) =
              Find_Max_Crossing_Subarray(A, low, mid, high)
8     if left_sum >= right_sum and left_sum >= cross_sum
9         return (left_low, left_high, left_sum)
10    elseif right_sum >= left_sum and right_sum >= cross_sum
11        return(right_low, right_high, right_sum)
12    else return(corss_low, cross_high, cross_sum)
```

程式:

```
// Name: B10532011高婷雅
// Date: March 21, 2018
// Last Update: March 21, 2018
// Problem statement: Profit Maximization Problem by recursive method
// A Divide and Conquer based program for maximum subarray sum problem
#include <stdio.h>
#include <limits.h>
#include <iostream>
using namespace std;

//用struct存答案的三個值，start、end、sum
struct ANS{int low, high, sum;};

// 找出橫跨中間的最大值
ANS maxCrossingSum(int *arr, int l, int m, int h)
{
    // Include elements on left of mid.
    int sum = 0;
    int left_sum = INT_MIN;
    int maxlow=0;
    for (int i = m; i >= l; i--)
    {
        sum = sum + arr[i];
        if (sum > left_sum){
          left_sum = sum;
          maxlow=i;
          }
    }
    // Include elements on right of mid
    sum = 0;
    int right_sum = INT_MIN;
    int maxhigh=0;
    for (int i = m+1; i <= h; i++)
    {
        sum = sum + arr[i];
        if (sum > right_sum){
          right_sum = sum;
          maxhigh=i;
        }
    }
    ANS array;
    array.low=maxlow;
    array.high=maxhigh;
    array.sum=left_sum + right_sum;
    // Return sum of elements on left and right of mid
    return array;
}
```

```cpp
// Returns sum of maxium sum subarray in aa[l..h]
ANS maxSubArraySum(int *arr, int l, int h)
{
    ANS subarrayleft,subarrayright,subarraycross;
    // Base Case: Only one element
    if (l == h){
        subarrayleft.low=l;
        subarrayleft.high=h;
        subarrayleft.sum=arr[l];
        return subarrayleft;
    }else{

    //找出中間值
    int m = (l + h)/2;

    subarrayleft=maxSubArraySum(arr, l, m);
    subarrayright=maxSubArraySum(arr, m+1, h);
    subarraycross=maxCrossingSum(arr, l, m, h);


    if (subarrayleft.sum>=subarrayright.sum&&subarrayleft.sum>=subarraycross.sum)
        return subarrayleft;
    else if(subarrayright.sum>=subarrayleft.sum&&subarrayright.sum>=subarraycross.sum)
        return subarrayright;
    else
        return subarraycross;
    }
}

int main() {
    ANS answer;
    int n; //while迴圈可以不斷輸入
    while (cin >> n) {
        int *arr=new int[n];
        for (int i = 0; i < n; i++) {
            cin >> arr[i];
        }
        //將整個arr代進function，傳回max_sum
        answer= maxSubArraySum(arr, 0, n-1);
        cout<<answer.low<<" "<<answer.high<<" "<<answer.sum<<endl;
        delete[] arr;
    }
    return 0;
}
```

Non-recursive:

程式摘要：

此程式用 C++語言

將所輸入的值做累加的，當有比較大的值則存入 Max

當累加的值突然遠小於之前的值，則歸零，並將 start 位置移到下一個位置

程式內容說明：

圖解:

Ex. N = 8　array = {7, -3, 1, -9, 10, 7, 6, 0}

對整個陣列往下累加，遇到某個加上去的極小值，則歸零

number	TempStart	TempEnd	TempSum	Start	End	Sum
7	0	0	7	0	0	7
-3	0	1	4	0	0	7
1	0	2	5	0	0	7
-9	**0 -> 3+1 (起始位置往後)**	**3**	**-4 -> 0 (歸零)**	**0**	**0**	**7**
10	4	4	10	4	4	10
7	4	5	17	4	5	17
-6	4	6	11	4	5	17
0	4	7	11	4	5	17

虛擬碼:

```
1 maxSubSeqSum(A[], n)
2     for i=0 to n
3         temp = temp + A[i]
4         if temp<0
5             temp = 0
6             tempi=i+1
7         elseif temp>max
8             max = temp
9             maxi = tempi
10            maxj = i
11    return (maxi, maxj, max)
```

程式:

```cpp
// Name: B10532011高靖雅
// Date: March 6, 2019
// Last Update: March 6, 2019
// Problem statement: Profit Maximization Problem by non-recursive method

#include<iostream>
using namespace std;
//宣告一個function找出最大subarray
int maxSubSeqSum(int arr[],int n);

int main() {
    int n; //while迴圈可以不斷輸入
    while (cin >> n) {
        int arr[n];
        for (int i = 0; i < n; i++) {
            cin >> arr[i];
        }
        //將整個arr代進function
        maxSubSeqSum(arr,n);
    }
    return 0;
}
int maxSubSeqSum(int arr[],int n){
        int temp=0, tempi=0;
        int max=0, maxi=0, maxj=0;

        for(int i=0; i < n; i++)
        {   //arr一個一個加上去
            temp=temp+arr[i];
            //當加到某一個數使得整個值連小於之前的值，則歸零
            //(這邊設定成當值變為負的時候來判斷)
            if(temp<0)
            {
                temp=0;
                tempi=i+1;//此時Start位置從for迴圈往後一個
            }
            //當新的值大於之前儲存的最大值，則更新最大值
            //且同時儲存位置
            else if(temp > max)
            {
                max = temp;
                maxi = tempi; //start位置為陣列開始加的位置
                maxj = i; //end位置為for迴圈所加的位置
            }
        }
        cout<<maxi<<" "<<maxj<<" "<<max<<endl;
}
```