**problem a08. b08:**

Given an array of integers, find a consecutive sub-array, including empty sub-array, with maximized sum.

兩題題目相同但a08: n<4000; b08: n<=600000

Analysis: 在一個正整數的array中找連續的一段使得總和最大。

如果資料在a[1..n]，a[0]=0.

是甚麼是所有可能得解? 對於所有的0<=i<=j<n, (a[0]用來對付empty subarray (sum=0)). If we try all possible solution

=>O(n^2) possible solution, each takes O(n) time.

=>O(n^3) time complexity =>TLE

How to save time?

Suppose we first compute the prefix-sum, i.e., letting

pre[i]=a[0]+a[1]+…a[i]

* a[i]+a[i+1]+…a[j]=pre[j]-pre[i-1]
* 每一個連續subarray的和可以在O(1)算出
* time complexity=O(n^2)

但如何計算pre[i]?

pre[0]=a[0]=0

pre[i]=pre[i-1]+a[i]

簡單的DP可以在O(n)算出所有prefix-sum, 但需要O(n)的space

這個技巧是”空間換取時間”, 利用preprocessing算出一些中間結果, 讓後續的計算加快

b08: n可能達到60000, O(n^2)無法在3 sec得到解。

如何加速?

方法不只一種。

以DP的概念，令f(i)是以i為右端點的最大subarray總和，則

f(1)=a[1]，

For i>1, we have

f(i)=(f(i-1)>0)? f(i-1)+a[i]: a[i];

而最佳解是f(i)中最大者或0，

因此先設opt=0，然後只要i由小往大算過去，每次檢查是否更新opt

注意:上述算式，右邊是f(i-1)左邊f(i)，此種遞迴由小往大算不需recursive call，此即是DP

Preprocessing 解法

左端在i的最大區間和=  
left(i)=  
max{sum[j,i]: j<=i}  
=prefix\_sum(i)-min{prefix\_sum(j-1): j<=i}

先求出所有點的prefix sum

在求出所有prefix\_sum的prefix\_min

m=0; // opt

pre[0]=0;

for (i=1;i<=n;i++) pre[i]=pre[i-1]+a[i];

pm[0]=0;

for (i=1;i<=n;i++) pm[i]=(pm[i-1]<pre[i])? pm[i-1]: pre[i];

for (i=1;i<=n;i++) {

k=pre[i]-pm[i-1]; // left(i)

if (k>m) m=k;

}