/\*

Find the contiguous subarray within an array (containing at least one number) which has the largest sum.

For example, given the array [-2,1,-3,4,-1,2,1,-5,4],

the contiguous subarray [4,-1,2,1] has the largest sum = 6.

方法一：把连续的正数负数合并，然后再做处理，太麻烦！

方法二：动态规划，找转移状态，

如果前面的sum<0,则sum=nums[i]

否则，sum+=nums[i]

每一步找一次最大

\*/

class Solution {

public:

int maxSubArray(vector<int>& nums)

{

//way-1

/\*

if(nums.size()==0)

return 0;

int max=INT\_MIN;

for(int i=0;i<nums.size();i++)

{

if(nums[i]>max)

max=nums[i];

}

if(max<=0)

return max;

int k=0;

while(k<nums.size())

{

if(nums[k]==0)

nums.erase(nums.begin()+k);

else if(((nums[k]>=0 && nums[k+1]>=0) || (nums[k]<=0 && nums[k+1]<=0)) && k+1<nums.size())

{

nums[k]=nums[k]+nums[k+1];

nums.erase(nums.begin()+k+1);

}

else

k++;

}

while(nums.size()>0)

{

if(nums[0]<0)

nums.erase(nums.begin());

else

break;

}

for(int i=0;i<nums.size();i++)

cout<<nums[i]<<" ";

cout<<endl;

if(nums.size()<=2)//只有一正一负或者全是正

return nums[0];

int count;

for(int i=0;i<nums.size();i=i+2)

{

k=i;

count=0;

for(int j=i;j<nums.size();j=j+2)

{

while(k<=j)

{

count=count+nums[k];

k++;

}

if(count>max)

max=count;

}

}

return max;

\*/

//way-2

int sum=0;

int maxa=INT\_MIN;

for(int i=0;i<nums.size();i++)

{

if(sum>=0)

sum+=nums[i];

else

sum=nums[i];

maxa=max(maxa,sum);

}

return maxa;

}

};