

# 053. Maximum Subarray

## 053 Maximum Subarray

- Divide and Conquer+array
- Dynamic Programming+array

### Description

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`.

[click to show more practice.](#)

Seen this question in a real interview before?

### 1. Thought line

### 2. Divide and Conquer+array

```
1 class Solution {
2 private:
3     int divideAndConquerMaxSubArray(vector<int>& nums, int st, int ed){
4         if (st>ed) return INT_MIN;
5         if (st==ed) return nums[st];
6         int mid = (st+ed)/2;
7         int lf_maxSubArraySum = divideAndConquerMaxSubArray(nums, st, mid-1);
8         int rt_maxSubArraySum = divideAndConquerMaxSubArray(nums, mid+1, ed);
9
10        // Calculate the possible result crossing middle point.
11        int midToLeft = INT_MIN, midToRight = INT_MIN;
12        for (int i = mid, sum=0; i>=st; --i){
13            sum+=nums[i];
14            midToLeft = midToLeft>sum?midToLeft:sum;
15        }
16        for (int i = mid, sum=0; i<=ed; ++i){
17            sum+=nums[i];
18            midToRight = midToRight>sum?midToRight:sum;
19        }
20        int mid_maxSubArraySum = midToLeft+midToRight-nums[mid];
21        return max(lf_maxSubArraySum,max(rt_maxSubArraySum,mid_maxSubArraySum));
22    }
23
24 public:
25     int maxSubArray(vector<int>& nums) {
26         return divideAndConquerMaxSubArray(nums, 0, nums.size()-1);
27     }
28 };
```