CS222 Homework 2

Name:王星艺 StudentID:5140309531

1. There are two sorted arrays nums1 and nums2 of size m and n respectively.

   Find the median of the two sorted arrays. The overall run time complexity should be O(log (m+n)).

   Example 1:

   nums1 = [1, 3]

   nums2 = [2]

   The median is 2.0

   Example 2:

   nums1 = [1, 2]

   nums2 = [3, 4]

   The median is (2 + 3)/2 = 2.5

   Input:

   int nums1[]; int m;

   int nums2[]; int n;

   Output:

   double median.

Function FindKth(int *nums1, int *nums2, int m, int n, int k)

```
1  if m > n then
2  |  return FindKth(nums2, nums1, n, m, k);
3  end
4  if m == 0 then
5  |  return nums2[k - 1];
6  end
7  if k == 1 then
8  |  return min(nums1[0], nums2[0]);
9  end
10 int p ⟵ min(m, k / 2);
11 int q ⟵ k - p;
12 if nums1[p - 1] < nums2[q - 1] then
13 |  return FindKth(nums1 + p, nums2, m - p, n, k - p);
14 else if nums1[p - 1] > nums2[q - 1] then
15 |  return FindKth(nums1, nums2 + q, m, n - q, k - q);
16 else
17 |  return nums1[p - 1];
18 end
```

Algorithm 1: Solution 1

Input: int nums1[]; int m; int nums2[]; int n;
Output: double median.

```
1 int k ⟵ (m + n) / 2;
2 if (m + n) % 2 == 0 then
3 |   return (FindKth(nums1, nums2, m, n, k) + FindKth(nums1, nums2, m, n, k + 1))
  |   / 2;
4 else
5 |   return FindKth(nums1, nums2, m, n, k+1);
6 end
```

2. 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.

   Input:

   int A[]: the input array.

   int N: length of A.

   Output:

   return the largest sum.

Algorithm 2: Solution 2

Input: int A[]: the input array; int N: length of A
Output: return the largest sum

```
1 int tmp ⟵ A[0];
2 int result ⟵ tmp;
3 for i ← 1 to N − 1 do
4 |   if tmp > 0 then
5 |   |   tmp ← tmp + A[i];
6 |   else
7 |   |   tmp ← A[i];
8 |   end
9 |   result ← max(result, tmp);
10 end
11 return result;
```

3. Given a non-empty array containing only positive integers, find if the array can be partitioned into two subsets such that the sum of elements in both subsets is equal.

   Note:

   Each of the array element will not exceed 100.

   The array size will not exceed 200.

   Example 1:

   Input: [1, 5, 11, 5]

   Output: true

   Explanation: The array can be partitioned as [1, 5, 5] and [11].

   Example 2:

   Input: [1, 2, 3, 5]

   Output: false

   Explanation: The array cannot be partitioned into equal sum subsets.

   Input:

   int A[]: the input array.

   int N: length of A.

   Output:

   return true or false.

Algorithm 3: Solution 3

Input: int A[]: the input array; int N: length of A
Output: return true or false

```
1  if the sum of A is odd then
2  |  return false;
3  end
4  int dp[sum / 2 + 1] ⟵ 0;
5  dp[0] ← 1;
6  for i ← 0 to N − 1 do
7  |  for j ← sum/2 to A[i] do
8  |  |  dp[j] ← dp[j] || dp[j-A[i]];
9  |  end
10 end
11 if dp[sum / 2] == 1 then
12 |  return true;
13 else
14 |  return false;
15 end
```