CS222 Homework 3

Name:王星艺 StudentID:5140309531

1. You are given coins of different denominations and a total amount of money amount.

   Write a function to compute the fewest number of coins that you need to make up that amount.

   If that amount of money cannot be made up by any combination of the coins, return -1.

   Example 1:

   coins = [1, 2, 5], amount = 11

   return 3 (11 = 5 + 5 + 1)

   Example 2:

   coins = [2], amount = 3

   return -1.

   Input:

   int coins[];

   int n: length of coins[];

   int amount;

   Output:

   int num;

Algorithm 1: Solution 1

Input: int coins[]; int n: length of coins[]; int amount;
Output: int num;

```
1  int dp[amount + 1] ⟵ amount + 1;
2  dp[0] ← 0;
3  for i ← 1 to amount do
4  |   for j ← 0 to n − 1 do
5  |   |   if i >= coins[j] then
6  |   |   |   dp[i] ← min(dp[i], dp[i - coins[j]] + 1);
7  |   |   end
8  |   end
9  end
10 if dp[amount] > amount then
11 |   return -1;
12 else
13 |   return dp[amount];
14 end
```

2. Given a string s, partition s such that every substring of the partition is a palindrome.

   Return the minimum cuts needed for a palindrome partitioning of s.

   For example, given s = "aab",

   Return 1 since the palindrome partitioning ["aa","b"] could be produced using 1 cut.

   Input:

   string s;

   Output:

   int cuts;

Algorithm 2: Solution 2

Input: string s;
Output: int cuts;

```
1  int n ⟵ s.length();
2  bool isPal[n][n] ⟵ False;
3  int cut[n];
4  for j ← 0 to n − 1 do
5  |  cut[j]← j;
6  |  for i ← 0 to j do
7  |  |  if s[i] == s[j]&&(j − i <= 1||isPal[i + 1][j − 1]) then
8  |  |  |  isPal[i][j]← True; if i == 0 then
9  |  |  |  |  cut[j]← 0;
10 |  |  |  else
11 |  |  |  |  cut[j]← min(cut[j], cut[i - 1] + 1);
12 |  |  |  end
13 |  |  end
14 |  end
15 end
16 return cut[n - 1];
```

3. Given two arrays of length m and n with digits 0-9 representing two numbers.

   Create the maximum number of length $k \leq m + n$ from digits of the two.

   The relative order of the digits from the same array must be preserved.

   Return an array of the k digits.

   You should try to optimize your time and space complexity.

   Example 1:

   nums1 = [3, 4, 6, 5]

   nums2 = [9, 1, 2, 5, 8, 3]

   k = 5

   return [9, 8, 6, 5, 3]

   Example 2:

nums1 = [6, 7]

nums2 = [6, 0, 4]

k = 5

return [6, 7, 6, 0, 4]

Example 3:

nums1 = [3, 9]

nums2 = [8, 9]

k = 3

return [9, 8, 9]

Input:

int nums1[], int m;

int nums2[], int n;

int k;

Output:

int nums[];

Function Greater(int *nums1, int *nums2, int i, int j)

```
int n1⟵ nums1.size();
int n2⟵ nums2.size();
while i < n1&&j < n2&&nums1[i] == nums2[j] do
    i⟵ i + 1;
    j⟵ j + 1;
end
if j == n2||(i < n1&&nums1[i] > num2[j]) then
    return True;
end
```

Function merge(int *stack1, int *stack2)

```
int p⟵ 0;
int q⟵ 0;
int n⟵ stack1.size() + stack2.size();
int stack[n];
for i ← 0 to n − 1 do
    if Greater(stack1, stack2, p, q) then
        stack[i]⟵ stack1[p];
        p⟵ p + 1;
    else
        stack[i]⟵ stack2[q];
        q⟵ q + 1;
    end
end
```

Function maxSubarray(int *nums, int k)

```
int n⟵ nums.size();
int stack[n];
int r⟵ 0;
for i ← 0 to n − 1 do
    while n − i > k − r&&r > 0&&nums[i] > stack[r − 1] do
        r← r - 1;
    end
    if r < k then
        stack[r]← nums[i];
        r← r + 1;
    end
end
return stack;
```

Algorithm 3: Solution 3

Input: int nums1[], int m; int nums2[], int n; int k;
Output: int nums[];

```
int start⟵ max(0, k - n);
int end⟵ min(m, k);
for i ← start to end do
    int stack1[i];
    int stack2[k-i];
    int nums_tmp[k];
    stack1← maxSubarray(nums1, i);
    stack2← maxSubarray(nums2, k-i);
    nums_tmp← merge(stack1, stack2);
    if Greater(nums_tmp, nums, 0, 0) then
        nums← nums_tmp;
    end
end
return nums;
```