213. House Robber II

Medium,
Dynamic Programming.

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed. All houses at this place are arranged in a circle. That means the first house is the neighbor of the last one. Meanwhile, adjacent houses have security system connected and it will automatically contact the police if two adjacent houses were broken into on the same night.

Given a list of non-negative integers representing the amount of money of each house, determine the maximum amount of money you can rob tonight without alerting the police.

Example 1:

```
Input: [2,3,2]
Output: 3
Explanation: You cannot rob house 1 (money = 2) and then rob house 3 (money = 2),
             because they are adjacent houses.
```

Example 2:

```
Input: [1,2,3,1]
Output: 4
Explanation: Rob house 1 (money = 1) and then rob house 3 (money = 3).
             Total amount you can rob = 1 + 3 = 4.
```

解法

这道题和House Robber I不同的地方是这道题是为环形。
第一个和最后一个房子可以认为是邻居。

所以需要分为三种情况。
第一种是选第一家，不选最后一家。

第二种是选最后一家，不选第一家。

第三种是都不选，直选中间的部分。

这三种情况的值的最大值为答案。

Java

```java
class Solution {
    public int rob(int[] nums) {
        int len = nums.length;
        int res = 0;
        if(len==0) return 0;
        else if(len==1 || len==2 || len==3){
            for(Integer num:nums) res=Math.max(res,num);
            return res;
        }
        int[] dp = new int[len+2];
        for(int i=0; i<len+2; i++) dp[0]=0;
        int first=0;
        int last=0;
        //选择第一个
        for(int i=2; i<len-1;i++){
            dp[i+2]=Math.max(dp[i+1],dp[i]+nums[i]);
        }
        first = nums[0]+dp[len];
        //初始化操作
        for(int i=0; i<len+2; i++) dp[0]=0;
        //选择最后一个
        for(int i=1; i<=len-3; i++){
            dp[i+2]=Math.max(dp[i+1],dp[i]+nums[i]);
        }
        last = nums[len-1]+dp[len-1];
        for(int i=0; i<len+2; i++) dp[0]=0;
        //第一个和最后一个都不选
        for(int i=1; i<len-1; i++){
            dp[i+2]=Math.max(dp[i+1],dp[i]+nums[i]);
        }
        res = dp[len];
        return Math.max(res,Math.max(first,last));
```

```
    }
}
```