JavaScript House Robber II

Challenge

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 a security system connected, and it will automatically contact the police if two adjacent houses were broken into on the same night.

Given an integer array `nums` representing the amount of money of each house, return the maximum amount of money you can rob tonight without alerting the police.

1st Example

```
Input: nums = [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.
```

2nd Example

```
Input: nums = [1,2,3,1]
Output: 4
```

Example continues on next page...

```
Explanation: Rob house 1 (money = 1) and then rob house 3
             (money = 3). Total amount you can rob = 1 + 3 =
             4.
```

3rd Example

```
Input: nums = [1,2,3]
Output: 3
```

Constraints

- `1 <= nums.length <= 100`
- `0 <= nums[i] <= 1000`

Solution

```
const rob = (nums) => {
    let dp = [];

    const helpRob = (start, house) => {
        if (house < start.start) {
            return 0;
        }

        if (dp[house] >= 0) {
            return dp[house];
        }

        dp[house] = Math.max((nums[house] +
                helpRob({start: start.start}, house - 2)),
                helpRob({start: start.start}, house - 1));
```

Solution continues on next page...

```
        return dp[house];
    };

    if (nums.length === 1) {
        return nums[0];
    }

    if (nums.length === 2) {
        return Math.max(nums[0], nums[1]);
    }

    let first = helpRob({start: 0}, nums.length - 2);

    dp = [];

    let second = helpRob({start: 1}, nums.length - 1);

    return Math.max(first, second);
};
```

Explanation

I've created a function called `rob` that takes an array `nums` as input, representing the amount of money in each house. The goal is to determine the maximum amount of money that can be robbed without alerting the police.

Inside the function, there is a helper function called `helpRob` that takes two parameters: `start` (representing the starting index) and `house` (representing the current house being considered).

Within `helpRob`, there are two base cases: if the current house is

less than the starting index, it returns `0`, indicating that all houses have been considered and no money can be robbed. If the maximum amount of money that can be robbed from the current house has already been calculated and stored in `dp`, it returns that value.

If the base cases are not met, the function calculates the maximum amount of money that can be robbed from the current house. It considers two possibilities: robbing the current house and the house two steps back, or not robbing the previous house. It recursively calls `helpRob` with updated parameters to calculate the maximum amount of money and then stores the result in `dp` for future reference.

After defining `helpRob`, the function checks for two special cases: if there is only one house, the maximum amount of money that can be robbed is the amount in that house; if there are only two houses, the maximum amount of money that can be robbed is the larger amount of the two houses.

If neither of the special cases is met, the function proceeds to calculate the maximum amount of money that can be robbed from the first half and the second half of the houses separately. It calls `helpRob` with appropriate parameters and stores the results in `first` and `second`.

Finally, the function returns the maximum amount of money that can be robbed from either the first half or the second half of the houses, whichever is larger.

In summary, the `rob` function solves this challenge by calculating the maximum amount of money that can be robbed from a given array of houses. It uses a dynamic programming approach and a

helper function to determine the optimal strategy for robbing houses without alerting the police.

Author: Trevor Morin