Day 10

05 December 2022 20:01

Problem Statement

There are N stones, numbered $1, 2, \ldots, N$. For each i ($1 \leq i \leq N$), the height of Stone i is h_i.

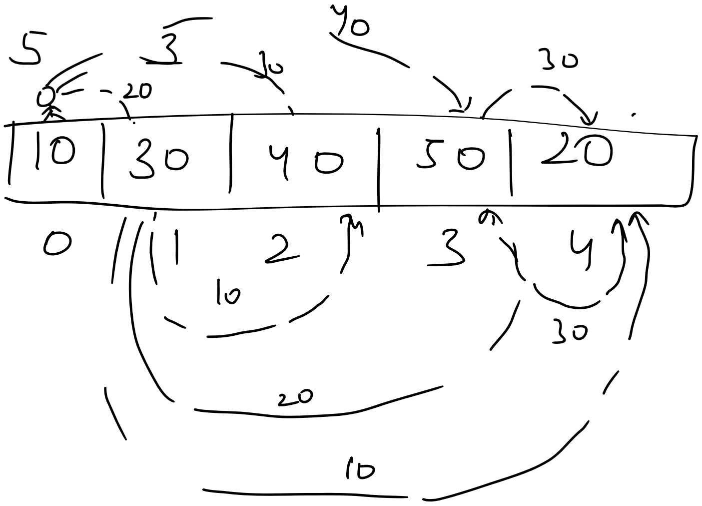
There is a frog who is initially on Stone 1. He will repeat the following action some number of times to reach Stone N:

- If the frog is currently on Stone i, jump to one of the following: Stone $i+1, i+2, \ldots, i+K$. Here, a cost of $|h_i - h_j|$ is incurred, where j is the stone to land on.

Find the minimum possible total cost incurred before the frog reaches Stone N.

Constraints

- All values in input are integers.
- $2 \leq N \leq 10^5$
- $1 \leq K \leq 100$
- $1 \leq h_i \leq 10^4$



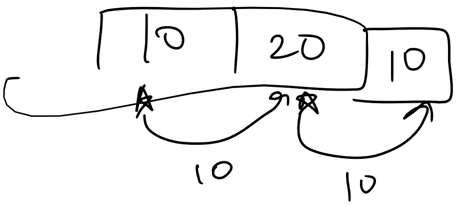
20 + 10

30

$1 \to 2 \to 5$ (30) ☆ || ☆

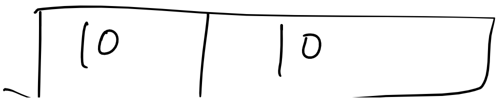
$1 \to 2 \to 4 \to 5$ (70)

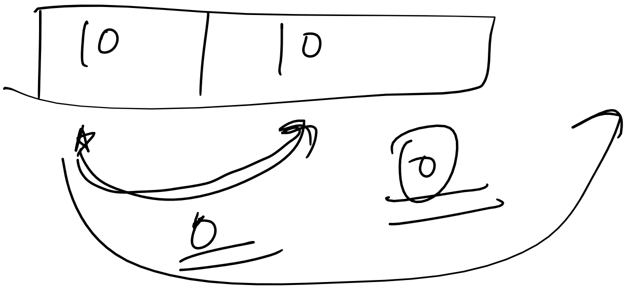
3 1



(20)

2 100






```
int solve(int i, vector<int> &cost,
                               int k)
    if(i==0) return 0;
  // choice   int mini = INT_MAX;
for(int jump=1; jump<=k; jump++){
      if( i-jump >= 0){
          mini = min(mini, solve(i-jump, cost, k)
                           + abs(cost[i]
                              - cost[i-jump]));
      }
}
return mini; ||
}
```

Recursion (80%) top down → Memoize (10%) → Tabulation (10%)

Bottom-top

<#include<bits/stdc++.h>

```cpp
using namespace std;
int solve(int i,int k,vector<int> &cost,vector<int> &dp){
    if(i == 0) return 0;
    if(dp[i] != -1){
        return dp[i];
    }
    int mini = INT_MAX;
    for(int p = 1;p <= k;p++){
        if(i - p >= 0){
            mini = min(mini,solve(i- p,k,cost,dp) + abs(cost[i] - cost[i-p]));
        }
    }
    return dp[i] = mini;
}
int main(){
    int n,k;
    cin >> n >> k;
    vector<int> cost(n);
    for(int i = 0;i < n;i++){
        cin >> cost[i];
    }
    vector<int> dp(n+1,-1);
    cout << solve(n-1,k,cost,dp);
}
```

Recursion

T.C – $O(K^N)$

S.C – $O(N)$

Memoization

T.C – $O(k \times N)$

S.C – $O(2N)$

N (Recursion) + N (DP)

Tabulation

T.C – $O(K \times N)$

S.C – $O(N)$ (DP)

Buffer overflow

198. House Robber

Medium 15.5K 303

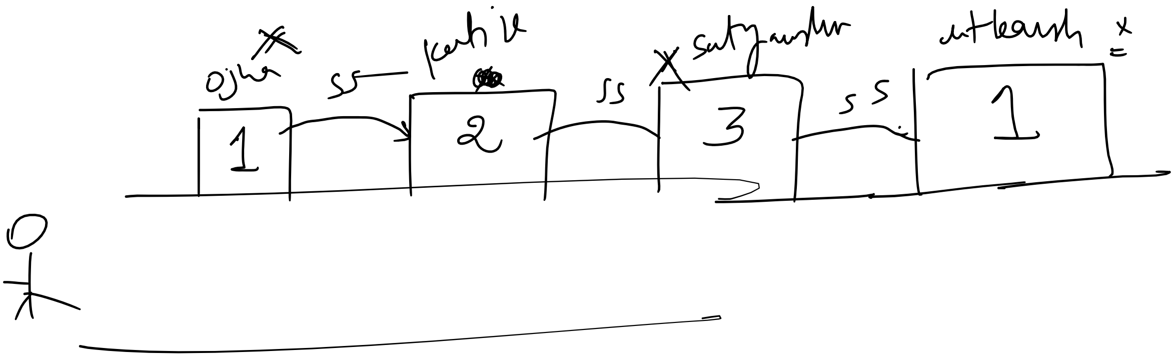
Companies

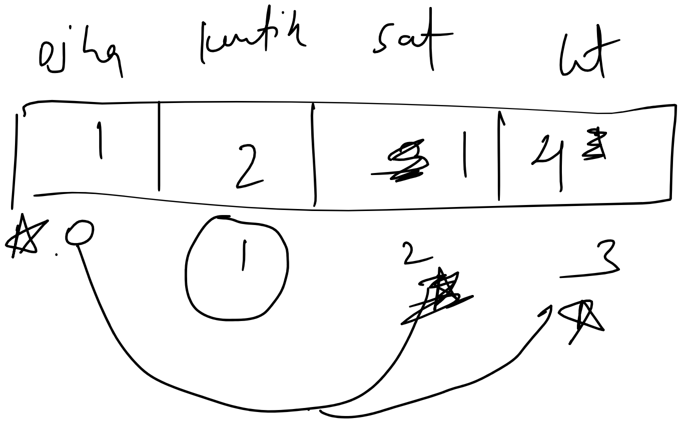
You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and **it will automatically contact the police if two adjacent houses 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***.

Example 1:

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



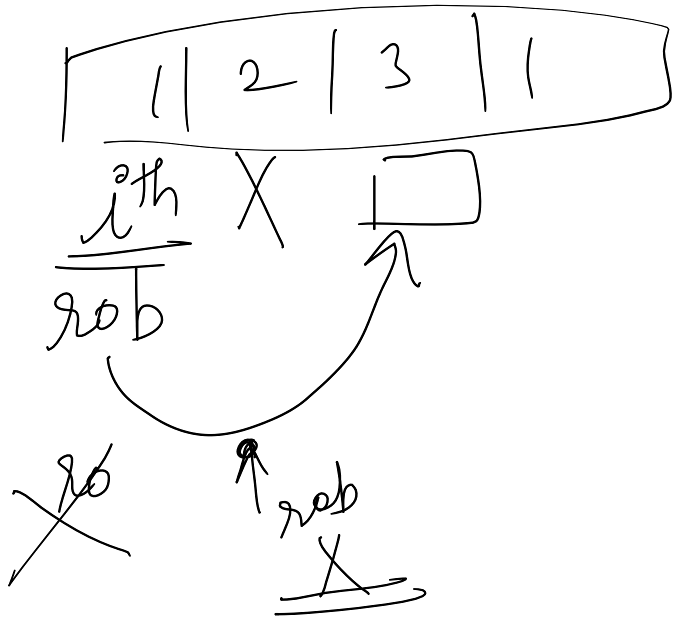


(2)

Recursion

Choice + Decision

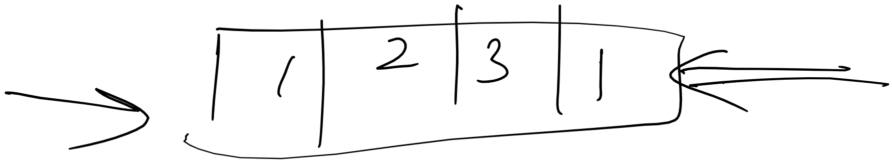
Choice + Decision



Choice Diagram

$i^{th} \rightarrow$ rob $\rightarrow$ cost[i] + (i+2)

$i^{th} \rightarrow$ not rob + (i+1)




```
int solve(vector<int>& paisa,
          int n)
```

if (n==0) return 0;

// rob int maxi = 0;

maxi = max(maxi, solve(paisa, n-2) + paisa[n-1]);

// rob X

maxi = max(maxi, solve(paisa, n-1));

return maxi;

}

Recursive (TLE)

```cpp
class Solution {
public:
    int solve(vector<int>& paisa,int n){
        if(n <= 0) return 0;
        int maxi = 0;
        // rob
        maxi = max(maxi,solve(paisa,n-2) + paisa[n-1]);
        // rob nahi krta
        maxi = max(maxi,solve(paisa,n-1));
        return maxi;
    }
    int rob(vector<int>& nums) {
        return solve(nums,nums.size());
    }
};
```

T.C — O(2^N)

S.C — O(N)

Memoization

```
class Solution {
public:
    int solve(vector<int>& paisa,int n,vector<int> &dp){
        if(n <= 0) return 0;
        if(dp[n] != -1) return dp[n];
        int maxi = 0;
        // rob
        maxi = max(maxi,solve(paisa,n-2,dp) + paisa[n-1]);
        // rob nahi krta
        maxi = max(maxi,solve(paisa,n-1,dp));
        return dp[n] = maxi;
    }
    int rob(vector<int>& nums) {
        vector<int> dp(nums.size() + 1,-1);
        return solve(nums,nums.size(),dp);
    }
};
```

T.C — O(N)

S.C — O(2N) → N ⇒ Recursion, N ⇒ DP Array

```
int rob(vector<int>& nums) {
    vector<int> dp(nums.size() + 1,0);
    // base case
    dp[0] = 0;
    int n = nums.size();
    for(int i = 1;i <= n;i++){
        int maxi = 0;
        // rob
        maxi = max(maxi,(i-2 >= 0?dp[i-2] : 0) + nums[i-1]);
        // rob nahi krta
        maxi = max(maxi,dp[i-1]);
        dp[i] = maxi;
    }
    return dp[n];
}
```

T.C — O(N)

S.C — O(N) → dp

dp[i-2] ⇒ dp[i]

i=3 dp[3-2] ⇒ dp[1]
dp[3-1] ⇒ dp[2]

i=2 dp[2-2] ⇒ dp[0]
dp[2-1] ⇒ dp[1]

i=1 dp[0]

i=4 dp[4-2] ⇒ dp[2]
dp[4-1] ⇒ dp[3]

i=n

dp[n-2]
dp[n-1]

int prev = -1

int curr = 0

dp[0] = 0

```
for(int i = 1;i <= n;i++){
            int maxi = 0;
            // rob                                -1
            maxi = max(maxi,(i-2 >= 0?prev : 0)
+ nums[i-1]);
            // rob nahi krta
            maxi = max(maxi,curr);
            Prev = curr; (0)
            curr = maxi;
        }
```

o(1)

```
        curr = maxi;
    }
```

$dp[i-2] \Longleftrightarrow prev$

$dp[i-1] \Longleftrightarrow curr$

O(1)

~~dp[i]~~

prev = curr;

curr = maxi;

prev curr curr

1	2	3	1

T.C — O(N)

S.C — O(1)

```
int rob(vector<int>& nums) {

    // base case
    int curr = 0; // dp[i-1]
    int prev = -1;  // dp[i-2]
     int n = nums.size();
    for(int i = 1;i <= n;i++){
        int maxi = 0;
        // rob
        maxi = max(maxi,(i-2 >= 0?prev : 0) + nums[i-1]);
        // rob nahi krta
        maxi = max(maxi,curr);
        prev = curr;
        curr = maxi;
    }
    return curr;
}
```

T.C — O(N)

S.C — O(1)

213. House Robber II

Hint

Medium 7.3K 108

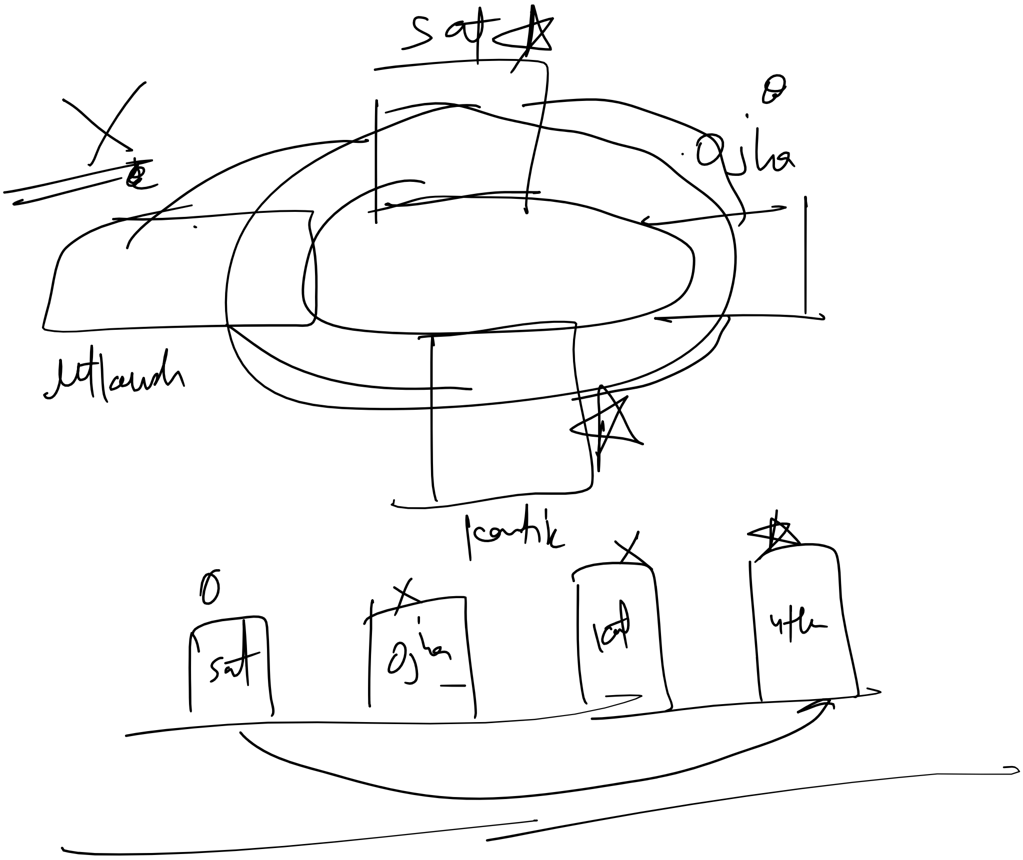
Companies

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***.

Example 1:

```
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.
```



```cpp
int simple_rob(vector<int>& nums) {

    // base case
    int curr = 0; // dp[i-1]
    int prev = -1;  // dp[i-2]
     int n = nums.size();
    for(int i = 1;i <= n;i++){
        int maxi = 0;
        // rob
        maxi = max(maxi,(i-2 >= 0?prev : 0) + nums[i-1]);
        // rob nahi krta
        maxi = max(maxi,curr);
        prev = curr;
        curr = maxi;
    }
    return curr;

nt rob(vector<int>& nums) {
    if(nums.size() == 1) return nums[0];
    if(nums.size() == 0) return 0;
    vector<int> first_skip,last_skip;
    int n = nums.size();
    for(int i= 0;i < n;i++)
    {
        if(i!= 0){
            first_skip.push_back(nums[i]);
        }
        if(i!= n-1){
             last_skip.push_back(nums[i]);
        }
    }
    return max(simple_rob(first_skip),simple_rob(last_skip));
```

T.C − $O(2N) \approx O(N)$

S.C − $O(2N) \approx O(N)$