Problem 375: Guess Number Higher or Lower II

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

We are playing the Guessing Game. The game will work as follows:

I pick a number between

1

and

n

.

You guess a number.

If you guess the right number,

you win the game

.

If you guess the wrong number, then I will tell you whether the number I picked is

higher or lower

, and you will continue guessing.

Every time you guess a wrong number

x

, you will pay

x

dollars. If you run out of money,

you lose the game

.

Given a particular

n

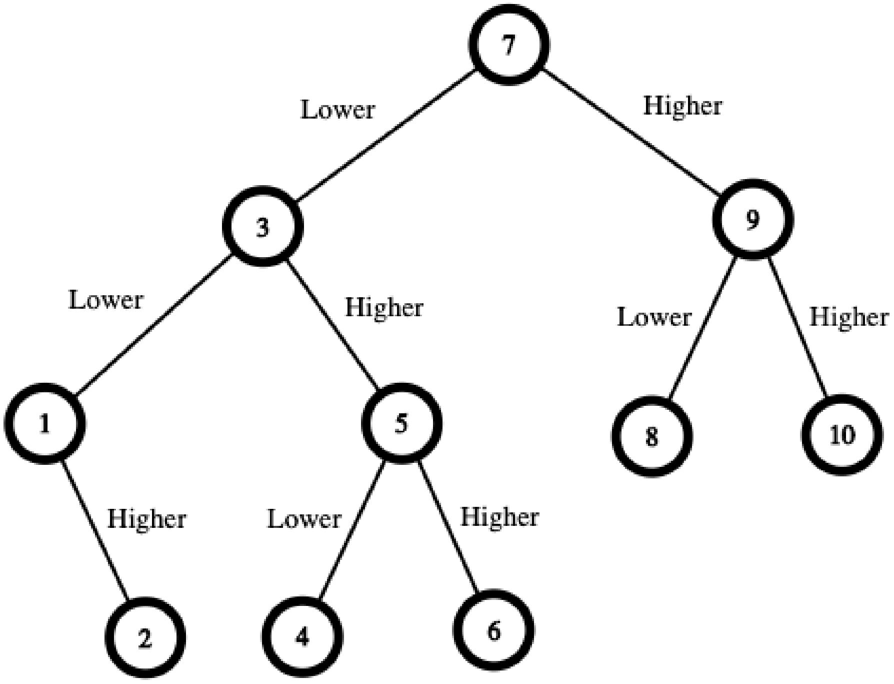
, return

the minimum amount of money you need to

guarantee a win regardless of what number I pick

.

Example 1:



Input:

n = 10

Output:

16

Explanation:

The winning strategy is as follows: - The range is [1,10]. Guess 7.   - If this is my number, your total is $0. Otherwise, you pay $7.   - If my number is higher, the range is [8,10]. Guess 9.   - If this is my number, your total is $7. Otherwise, you pay $9.   - If my number is higher, it must be 10. Guess 10. Your total is $7 + $9 = $16.   - If my number is lower, it must be 8. Guess 8. Your total is $7 + $9 = $16.   - If my number is lower, the range is [1,6]. Guess 3.   - If this is my number, your total is $7. Otherwise, you pay $3.   - If my number is higher, the range is [4,6]. Guess 5.   - If this is my number, your total is $7 + $3 = $10. Otherwise, you pay $5.   - If my number is higher, it must be 6. Guess 6. Your total is $7 + $3 + $5 = $15.   - If my number is lower, it must be 4. Guess 4. Your total is $7 + $3 + $5 = $15.   - If my number is lower, the range is [1,2]. Guess 1.   - If this is my number, your total is $7 + $3 = $10. Otherwise, you pay $1.   - If my number is higher, it must be 2. Guess 2. Your total is $7 + $3 + $1 = $11. The worst case in all these scenarios is that you pay $16. Hence, you only need $16 to guarantee a win.

Example 2:

Input:

n = 1

Output:

0

Explanation:

There is only one possible number, so you can guess 1 and not have to pay anything.

Example 3:

Input:

n = 2

Output:

1

Explanation:

There are two possible numbers, 1 and 2. - Guess 1.   - If this is my number, your total is $0. Otherwise, you pay $1.   - If my number is higher, it must be 2. Guess 2. Your total is $1. The worst case is that you pay $1.

Constraints:

1 <= n <= 200

Code Snippets

C++:

```
class Solution {
public:
int getMoneyAmount(int n) {

}
};
```

Java:

```
class Solution {
public int getMoneyAmount(int n) {

}
}
```

Python3:

```
class Solution:
def getMoneyAmount(self, n: int) -> int:
```

Python:

```
class Solution(object):
def getMoneyAmount(self, n):
"""
:type n: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @return {number}
*/
var getMoneyAmount = function(n) {

};
```

TypeScript:

```
function getMoneyAmount(n: number): number {

};
```

C#:

```
public class Solution {
public int GetMoneyAmount(int n) {

}
}
```

C:

```
int getMoneyAmount(int n) {

}
```

Go:

```
func getMoneyAmount(n int) int {

}
```

Kotlin:

```
class Solution {
fun getMoneyAmount(n: Int): Int {

}
}
```

Swift:

```
class Solution {
func getMoneyAmount(_ n: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn get_money_amount(n: i32) -> i32 {


}
}
```

Ruby:

```
# @param {Integer} n
# @return {Integer}
def get_money_amount(n)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function getMoneyAmount($n) {

}
}
```

Dart:

```
class Solution {
int getMoneyAmount(int n) {

}
}
```

Scala:

```
object Solution {
def getMoneyAmount(n: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec get_money_amount(n :: integer) :: integer
def get_money_amount(n) do

end
end
```

Erlang:

```
-spec get_money_amount(N :: integer()) -> integer().
get_money_amount(N) ->
.
```

Racket:

```
(define/contract (get-money-amount n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Guess Number Higher or Lower II
* Difficulty: Medium
* Tags: graph, dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int getMoneyAmount(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Guess Number Higher or Lower II
* Difficulty: Medium
* Tags: graph, dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int getMoneyAmount(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Guess Number Higher or Lower II
Difficulty: Medium
Tags: graph, dp, math

Approach: Dynamic programming with memoization or tabulation
Time Complexity: O(n * m) where n and m are problem dimensions
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def getMoneyAmount(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def getMoneyAmount(self, n):
"""
:type n: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Guess Number Higher or Lower II
* Difficulty: Medium
* Tags: graph, dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {number} n
* @return {number}
*/
var getMoneyAmount = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Guess Number Higher or Lower II
* Difficulty: Medium
* Tags: graph, dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

function getMoneyAmount(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Guess Number Higher or Lower II
* Difficulty: Medium
* Tags: graph, dp, math
*
```

```
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int GetMoneyAmount(int n) {

}
}
```

C Solution:

```
/*
* Problem: Guess Number Higher or Lower II
* Difficulty: Medium
* Tags: graph, dp, math
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

int getMoneyAmount(int n) {

}
```

Go Solution:

```
// Problem: Guess Number Higher or Lower II
// Difficulty: Medium
// Tags: graph, dp, math
//
// Approach: Dynamic programming with memoization or tabulation
// Time Complexity: O(n * m) where n and m are problem dimensions
// Space Complexity: O(n) or O(n * m) for DP table

func getMoneyAmount(n int) int {

}
```

Kotlin Solution:

```
class Solution {
fun getMoneyAmount(n: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func getMoneyAmount(_ n: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Guess Number Higher or Lower II
// Difficulty: Medium
// Tags: graph, dp, math
//
// Approach: Dynamic programming with memoization or tabulation
// Time Complexity: O(n * m) where n and m are problem dimensions
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn get_money_amount(n: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @return {Integer}
def get_money_amount(n)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function getMoneyAmount($n) {

}
}
```

Dart Solution:

```
class Solution {
int getMoneyAmount(int n) {

}
}
```

Scala Solution:

```
object Solution {
def getMoneyAmount(n: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec get_money_amount(n :: integer) :: integer
def get_money_amount(n) do

end
end
```

Erlang Solution:

```
-spec get_money_amount(N :: integer()) -> integer().
get_money_amount(N) ->
.
```

Racket Solution:

```
(define/contract (get-money-amount n)
(-> exact-integer? exact-integer?)
)
```