Problem 1230: Toss Strange Coins

Problem Information

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

Problem Description

You have some coins. The

i

-th coin has a probability

prob[i]

of facing heads when tossed.

Return the probability that the number of coins facing heads equals

target

if you toss every coin exactly once.

Example 1:

Input:

prob = [0.4], target = 1

Output:

0.40000

Example 2:

Input:

prob = [0.5,0.5,0.5,0.5,0.5], target = 0

Output:

0.03125

Constraints:

1 <= prob.length <= 1000

0 <= prob[i] <= 1

0 <= target

<= prob.length

Answers will be accepted as correct if they are within

10^-5

of the correct answer.

Code Snippets

C++:

```
class Solution {
public:
double probabilityOfHeads(vector<double>& prob, int target) {

}
};
```

Java:

```
class Solution {
public double probabilityOfHeads(double[] prob, int target) {

}
}
```

Python3:

```
class Solution:
def probabilityOfHeads(self, prob: List[float], target: int) -> float:
```

Python:

```
class Solution(object):
def probabilityOfHeads(self, prob, target):
"""
:type prob: List[float]
:type target: int
:rtype: float
"""
```

JavaScript:

```
/**
* @param {number[]} prob
* @param {number} target
* @return {number}
*/
var probabilityOfHeads = function(prob, target) {

};
```

TypeScript:

```
function probabilityOfHeads(prob: number[], target: number): number {

};
```

C#:

```
public class Solution {
public double ProbabilityOfHeads(double[] prob, int target) {

}
}
```

C:

```
double probabilityOfHeads(double* prob, int probSize, int target){

}
```

Go:

```
func probabilityOfHeads(prob []float64, target int) float64 {

}
```

Kotlin:

```
class Solution {
fun probabilityOfHeads(prob: DoubleArray, target: Int): Double {

}
}
```

Swift:

```
class Solution {
func probabilityOfHeads(_ prob: [Double], _ target: Int) -> Double {

}
}
```

Rust:

```
impl Solution {
pub fn probability_of_heads(prob: Vec<f64>, target: i32) -> f64 {

}
}
```

Ruby:

```
# @param {Float[]} prob
# @param {Integer} target
# @return {Float}
def probability_of_heads(prob, target)

end
```

PHP:

```
class Solution {

/**
* @param Float[] $prob
* @param Integer $target
* @return Float
*/
function probabilityOfHeads($prob, $target) {

}
}
```

Dart:

```
class Solution {
double probabilityOfHeads(List<double> prob, int target) {

}
}
```

Scala:

```
object Solution {
def probabilityOfHeads(prob: Array[Double], target: Int): Double = {

}
}
```

Elixir:

```
defmodule Solution do
@spec probability_of_heads(prob :: [float], target :: integer) :: float
def probability_of_heads(prob, target) do
```

```
end
end
```

Erlang:

```
-spec probability_of_heads(Prob :: [float()], Target :: integer()) ->
float().
probability_of_heads(Prob, Target) ->
.
```

Racket:

```
(define/contract (probability-of-heads prob target)
(-> (listof flonum?) exact-integer? flonum?)

)
```

Solutions

C++ Solution:

```
/*
* Problem: Toss Strange Coins
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
double probabilityOfHeads(vector<double>& prob, int target) {

}
};
```

Java Solution:

```
/**
* Problem: Toss Strange Coins
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public double probabilityOfHeads(double[] prob, int target) {

}
}
```

Python3 Solution:

```
"""
Problem: Toss Strange Coins
Difficulty: Medium
Tags: array, dp, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def probabilityOfHeads(self, prob: List[float], target: int) -> float:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def probabilityOfHeads(self, prob, target):
"""
:type prob: List[float]
:type target: int
:rtype: float
"""
```

JavaScript Solution:

```
/**
* Problem: Toss Strange Coins
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {number[]} prob
* @param {number} target
* @return {number}
*/
var probabilityOfHeads = function(prob, target) {

};
```

TypeScript Solution:

```
/**
* Problem: Toss Strange Coins
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function probabilityOfHeads(prob: number[], target: number): number {

};
```

C# Solution:

```
/*
* Problem: Toss Strange Coins
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public double ProbabilityOfHeads(double[] prob, int target) {

}
}
```

C Solution:

```
/*
* Problem: Toss Strange Coins
* Difficulty: Medium
* Tags: array, dp, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

double probabilityOfHeads(double* prob, int probSize, int target){

}
```

Go Solution:

```
// Problem: Toss Strange Coins
// Difficulty: Medium
// Tags: array, dp, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func probabilityOfHeads(prob []float64, target int) float64 {

}
```

Kotlin Solution:

```
class Solution {
fun probabilityOfHeads(prob: DoubleArray, target: Int): Double {

}
}
```

Swift Solution:

```
class Solution {
func probabilityOfHeads(_ prob: [Double], _ target: Int) -> Double {

}
}
```

Rust Solution:

```
// Problem: Toss Strange Coins
// Difficulty: Medium
// Tags: array, dp, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn probability_of_heads(prob: Vec<f64>, target: i32) -> f64 {

}
}
```

Ruby Solution:

```
# @param {Float[]} prob
# @param {Integer} target
# @return {Float}
def probability_of_heads(prob, target)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param Float[] $prob
* @param Integer $target
* @return Float
*/
function probabilityOfHeads($prob, $target) {

}
}
```

Dart Solution:

```
class Solution {
double probabilityOfHeads(List<double> prob, int target) {

}
}
```

Scala Solution:

```
object Solution {
def probabilityOfHeads(prob: Array[Double], target: Int): Double = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec probability_of_heads(prob :: [float], target :: integer) :: float
def probability_of_heads(prob, target) do

end
end
```

Erlang Solution:

```
-spec probability_of_heads(Prob :: [float()], Target :: integer()) ->
float().
probability_of_heads(Prob, Target) ->
.
```

Racket Solution:

```
(define/contract (probability-of-heads prob target)
(-> (listof flonum?) exact-integer? flonum?)

)
```