Problem 1335: Minimum Difficulty of a Job Schedule

Problem Information

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

Problem Description

You want to schedule a list of jobs in

d

days. Jobs are dependent (i.e To work on the

i

th

job, you have to finish all the jobs

j

where

0 <= j < i

).

You have to finish

at least

one task every day. The difficulty of a job schedule is the sum of difficulties of each day of the d

days. The difficulty of a day is the maximum difficulty of a job done on that day.

You are given an integer array

jobDifficulty

and an integer

d

. The difficulty of the

i

th

job is

jobDifficulty[i]

.

Return

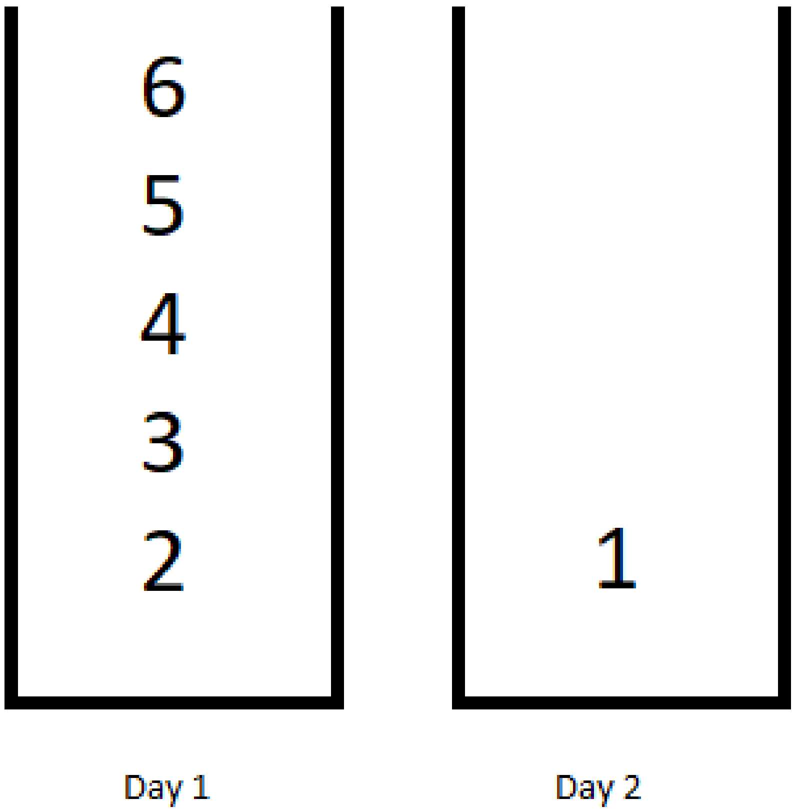
the minimum difficulty of a job schedule

. If you cannot find a schedule for the jobs return

-1

.

Example 1:



Input:

jobDifficulty = [6,5,4,3,2,1], d = 2

Output:

7

Explanation:

First day you can finish the first 5 jobs, total difficulty = 6. Second day you can finish the last job, total difficulty = 1. The difficulty of the schedule = 6 + 1 = 7

Example 2:

Input:

jobDifficulty = [9,9,9], d = 4

Output:

-1

Explanation:

If you finish a job per day you will still have a free day. you cannot find a schedule for the given jobs.

Example 3:

Input:

jobDifficulty = [1,1,1], d = 3

Output:

3

Explanation:

The schedule is one job per day. total difficulty will be 3.

Constraints:

1 <= jobDifficulty.length <= 300

0 <= jobDifficulty[i] <= 1000

1 <= d <= 10

Code Snippets

C++:

```
class Solution {
public:
int minDifficulty(vector<int>& jobDifficulty, int d) {

}
};
```

Java:

```
class Solution {
public int minDifficulty(int[] jobDifficulty, int d) {

}
}
```

Python3:

```
class Solution:
def minDifficulty(self, jobDifficulty: List[int], d: int) -> int:
```

Python:

```
class Solution(object):
def minDifficulty(self, jobDifficulty, d):
"""
:type jobDifficulty: List[int]
:type d: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} jobDifficulty
* @param {number} d
* @return {number}
*/
var minDifficulty = function(jobDifficulty, d) {

};
```

TypeScript:

```
function minDifficulty(jobDifficulty: number[], d: number): number {


};
```

C#:

```
public class Solution {
public int MinDifficulty(int[] jobDifficulty, int d) {


}
}
```

C:

```
int minDifficulty(int* jobDifficulty, int jobDifficultySize, int d) {


}
```

Go:

```
func minDifficulty(jobDifficulty []int, d int) int {


}
```

Kotlin:

```
class Solution {
fun minDifficulty(jobDifficulty: IntArray, d: Int): Int {


}
}
```

Swift:

```
class Solution {
func minDifficulty(_ jobDifficulty: [Int], _ d: Int) -> Int {


}
}
```

Rust:

```
impl Solution {
pub fn min_difficulty(job_difficulty: Vec<i32>, d: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} job_difficulty
# @param {Integer} d
# @return {Integer}
def min_difficulty(job_difficulty, d)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $jobDifficulty
* @param Integer $d
* @return Integer
*/
function minDifficulty($jobDifficulty, $d) {

}
}
```

Dart:

```
class Solution {
int minDifficulty(List<int> jobDifficulty, int d) {

}
}
```

Scala:

```
object Solution {
def minDifficulty(jobDifficulty: Array[Int], d: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec min_difficulty(job_difficulty :: [integer], d :: integer) :: integer
def min_difficulty(job_difficulty, d) do

end
end
```

Erlang:

```
-spec min_difficulty(JobDifficulty :: [integer()], D :: integer()) ->
integer().
min_difficulty(JobDifficulty, D) ->
.
```

Racket:

```
(define/contract (min-difficulty jobDifficulty d)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Difficulty of a Job Schedule
* Difficulty: Hard
* Tags: array, dp
*
* 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:
```

```
int minDifficulty(vector<int>& jobDifficulty, int d) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Difficulty of a Job Schedule
* Difficulty: Hard
* Tags: array, dp
*
* 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 int minDifficulty(int[] jobDifficulty, int d) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Difficulty of a Job Schedule
Difficulty: Hard
Tags: array, dp

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 minDifficulty(self, jobDifficulty: List[int], d: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def minDifficulty(self, jobDifficulty, d):
"""
:type jobDifficulty: List[int]
:type d: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Difficulty of a Job Schedule
* Difficulty: Hard
* Tags: array, dp
*
* 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[]} jobDifficulty
* @param {number} d
* @return {number}
*/
var minDifficulty = function(jobDifficulty, d) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Difficulty of a Job Schedule
* Difficulty: Hard
* Tags: array, dp
*
* 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 minDifficulty(jobDifficulty: number[], d: number): number {
};
```

C# Solution:

```
/*
* Problem: Minimum Difficulty of a Job Schedule
* Difficulty: Hard
* Tags: array, dp
*
* 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 int MinDifficulty(int[] jobDifficulty, int d) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Difficulty of a Job Schedule
* Difficulty: Hard
* Tags: array, dp
*
* 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
*/

int minDifficulty(int* jobDifficulty, int jobDifficultySize, int d) {

}
```

Go Solution:

```
// Problem: Minimum Difficulty of a Job Schedule
// Difficulty: Hard
// Tags: array, dp
//
// 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 minDifficulty(jobDifficulty []int, d int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minDifficulty(jobDifficulty: IntArray, d: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func minDifficulty(_ jobDifficulty: [Int], _ d: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Difficulty of a Job Schedule
// Difficulty: Hard
// Tags: array, dp
//
// 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 min_difficulty(job_difficulty: Vec<i32>, d: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} job_difficulty
# @param {Integer} d
# @return {Integer}
def min_difficulty(job_difficulty, d)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $jobDifficulty
* @param Integer $d
* @return Integer
*/
function minDifficulty($jobDifficulty, $d) {

}
}
```

Dart Solution:

```
class Solution {
int minDifficulty(List<int> jobDifficulty, int d) {

}
}
```

Scala Solution:

```
object Solution {
def minDifficulty(jobDifficulty: Array[Int], d: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_difficulty(job_difficulty :: [integer], d :: integer) :: integer
def min_difficulty(job_difficulty, d) do

end
end
```

Erlang Solution:

```
-spec min_difficulty(JobDifficulty :: [integer()], D :: integer()) ->
integer().
min_difficulty(JobDifficulty, D) ->
.
```

Racket Solution:

```
(define/contract (min-difficulty jobDifficulty d)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```