

Problem 2829: Determine the Minimum Sum of a k-avoiding Array

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given two integers,

n

and

k

An array of

distinct

positive integers is called a

k -avoiding

array if there does not exist any pair of distinct elements that sum to

k

Return

the

minimum

possible sum of a k-avoiding array of length

n

.

Example 1:

Input:

$n = 5, k = 4$

Output:

18

Explanation:

Consider the k-avoiding array [1,2,4,5,6], which has a sum of 18. It can be proven that there is no k-avoiding array with a sum less than 18.

Example 2:

Input:

$n = 2, k = 6$

Output:

3

Explanation:

We can construct the array [1,2], which has a sum of 3. It can be proven that there is no k-avoiding array with a sum less than 3.

Constraints:

$1 \leq n, k \leq 50$

Code Snippets

C++:

```
class Solution {  
public:  
    int minimumSum(int n, int k) {  
  
    }  
};
```

Java:

```
class Solution {  
public int minimumSum(int n, int k) {  
  
}  
}
```

Python3:

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

Python:

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

JavaScript:

```
/**  
 * @param {number} n  
 * @param {number} k  
 * @return {number}  
 */  
var minimumSum = function(n, k) {  
};
```

TypeScript:

```
function minimumSum(n: number, k: number): number {  
};
```

C#:

```
public class Solution {  
    public int MinimumSum(int n, int k) {  
  
    }  
}
```

C:

```
int minimumSum(int n, int k) {  
  
}
```

Go:

```
func minimumSum(n int, k int) int {  
  
}
```

Kotlin:

```
class Solution {  
    fun minimumSum(n: Int, k: Int): Int {  
  
    }  
}
```

Swift:

```
class Solution {  
    func minimumSum(_ n: Int, _ k: Int) -> Int {  
  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn minimum_sum(n: i32, k: i32) -> i32 {  
  
    }  
}
```

Ruby:

```
# @param {Integer} n  
# @param {Integer} k  
# @return {Integer}  
def minimum_sum(n, k)  
  
end
```

PHP:

```
class Solution {  
  
    /**  
     * @param Integer $n  
     * @param Integer $k  
     * @return Integer  
     */  
    function minimumSum($n, $k) {  
  
    }  
}
```

Dart:

```
class Solution {  
    int minimumSum(int n, int k) {
```

```
}
```

```
}
```

Scala:

```
object Solution {  
    def minimumSum(n: Int, k: Int): Int = {  
  
    }  
    }  
}
```

Elixir:

```
defmodule Solution do  
  @spec minimum_sum(n :: integer, k :: integer) :: integer  
  def minimum_sum(n, k) do  
  
  end  
  end
```

Erlang:

```
-spec minimum_sum(N :: integer(), K :: integer()) -> integer().  
minimum_sum(N, K) ->  
.
```

Racket:

```
(define/contract (minimum-sum n k)  
  (-> exact-integer? exact-integer? exact-integer?)  
)
```

Solutions

C++ Solution:

```
/*  
 * Problem: Determine the Minimum Sum of a k-avoiding Array  
 * Difficulty: Medium
```

```

* Tags: array, greedy, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

```

```

class Solution {
public:
int minimumSum(int n, int k) {

}
};

```

Java Solution:

```

/**
* Problem: Determine the Minimum Sum of a k-avoiding Array
* Difficulty: Medium
* Tags: array, greedy, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

```

```

class Solution {
public int minimumSum(int n, int k) {

}
}

```

Python3 Solution:

```

"""
Problem: Determine the Minimum Sum of a k-avoiding Array
Difficulty: Medium
Tags: array, greedy, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)

```

```
Space Complexity: O(1) to O(n) depending on approach
```

```
"""
```

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

Python Solution:

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

JavaScript Solution:

```
/**  
 * Problem: Determine the Minimum Sum of a k-avoiding Array  
 * Difficulty: Medium  
 * Tags: array, greedy, math  
 *  
 * Approach: Use two pointers or sliding window technique  
 * Time Complexity: O(n) or O(n log n)  
 * Space Complexity: O(1) to O(n) depending on approach  
 */  
  
/**  
 * @param {number} n  
 * @param {number} k  
 * @return {number}  
 */  
var minimumSum = function(n, k) {  
  
};
```

TypeScript Solution:

```

/**
 * Problem: Determine the Minimum Sum of a k-avoiding Array
 * Difficulty: Medium
 * Tags: array, greedy, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

function minimumSum(n: number, k: number): number {

};

```

C# Solution:

```

/*
 * Problem: Determine the Minimum Sum of a k-avoiding Array
 * Difficulty: Medium
 * Tags: array, greedy, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach
 */

public class Solution {
    public int MinimumSum(int n, int k) {

    }
}

```

C Solution:

```

/*
 * Problem: Determine the Minimum Sum of a k-avoiding Array
 * Difficulty: Medium
 * Tags: array, greedy, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(1) to O(n) depending on approach

```

```
*/  
  
int minimumSum(int n, int k) {  
  
}
```

Go Solution:

```
// Problem: Determine the Minimum Sum of a k-avoiding Array  
// Difficulty: Medium  
// Tags: array, greedy, math  
//  
// Approach: Use two pointers or sliding window technique  
// Time Complexity: O(n) or O(n log n)  
// Space Complexity: O(1) to O(n) depending on approach  
  
func minimumSum(n int, k int) int {  
  
}
```

Kotlin Solution:

```
class Solution {  
    fun minimumSum(n: Int, k: Int): Int {  
  
    }  
}
```

Swift Solution:

```
class Solution {  
    func minimumSum(_ n: Int, _ k: Int) -> Int {  
  
    }  
}
```

Rust Solution:

```
// Problem: Determine the Minimum Sum of a k-avoiding Array  
// Difficulty: Medium  
// Tags: array, greedy, math
```

```

// 
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn minimum_sum(n: i32, k: i32) -> i32 {

}
}

```

Ruby Solution:

```

# @param {Integer} n
# @param {Integer} k
# @return {Integer}
def minimum_sum(n, k)

end

```

PHP Solution:

```

class Solution {

/**
 * @param Integer $n
 * @param Integer $k
 * @return Integer
 */
function minimumSum($n, $k) {

}
}

```

Dart Solution:

```

class Solution {
int minimumSum(int n, int k) {

}
}

```

Scala Solution:

```
object Solution {  
    def minimumSum(n: Int, k: Int): Int = {  
  
    }  
}
```

Elixir Solution:

```
defmodule Solution do  
  @spec minimum_sum(n :: integer, k :: integer) :: integer  
  def minimum_sum(n, k) do  
  
  end  
end
```

Erlang Solution:

```
-spec minimum_sum(N :: integer(), K :: integer()) -> integer().  
minimum_sum(N, K) ->  
.
```

Racket Solution:

```
(define/contract (minimum-sum n k)  
  (-> exact-integer? exact-integer? exact-integer?)  
)
```