Problem 1643: Kth Smallest Instructions

Problem Information

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

Problem Description

Bob is standing at cell

(0, 0)

, and he wants to reach

destination

:

(row, column)

. He can only travel

right

and

down

. You are going to help Bob by providing

instructions

for him to reach

destination

.

The

instructions

are represented as a string, where each character is either:

'H'

, meaning move horizontally (go

right

), or

'V'

, meaning move vertically (go

down

).

Multiple

instructions

will lead Bob to

destination

. For example, if

destination

is

(2, 3)

, both

"HHHVV"

and

"HVHVH"

are valid

instructions

.

However, Bob is very picky. Bob has a lucky number

k

, and he wants the

k

th

lexicographically smallest instructions

that will lead him to

destination

.

k

is

1-indexed

.

Given an integer array

destination

and an integer

k

, return

the

k

th

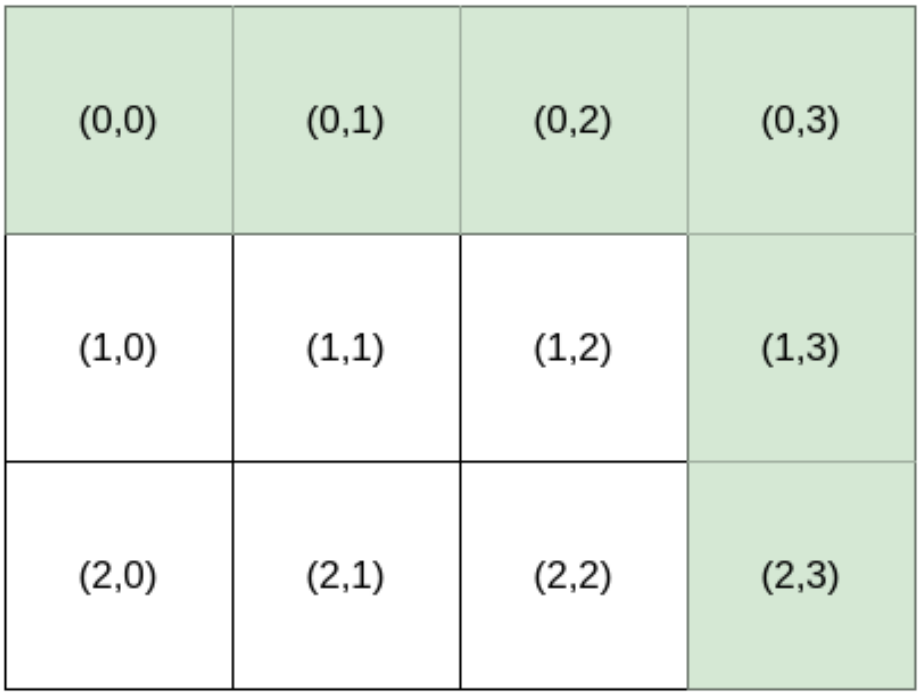
lexicographically smallest instructions

that will take Bob to

destination

.

Example 1:



Input:

destination = [2,3], k = 1

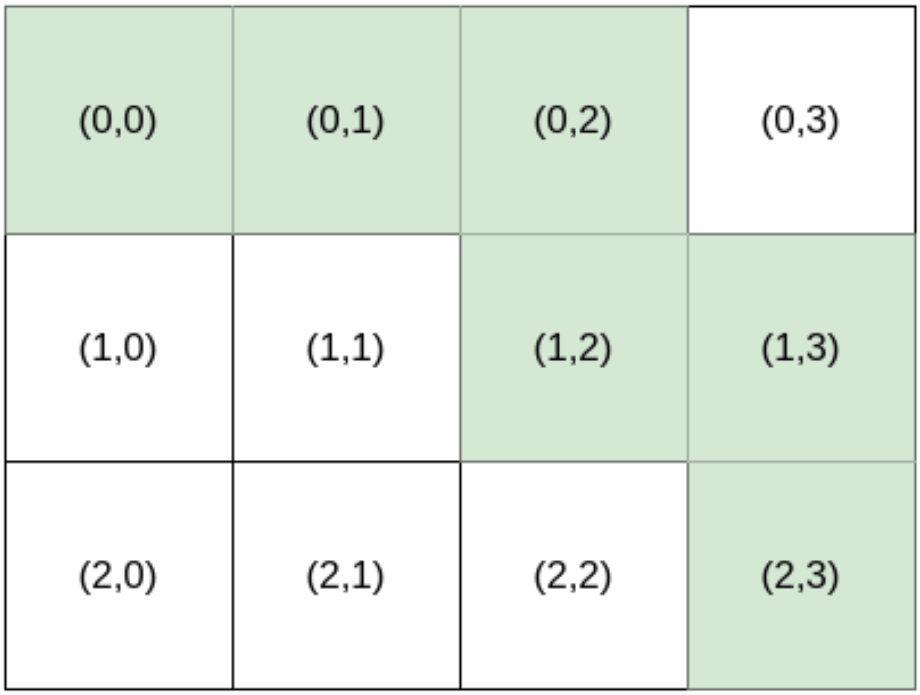
Output:

"HHHVV"

Explanation:

All the instructions that reach (2, 3) in lexicographic order are as follows: ["HHHVV", "HHVHV", "HHVVH", "HVHHV", "HVHVH", "HVVHH", "VHHHV", "VHHVH", "VHVHH", "VVHHH"].

Example 2:



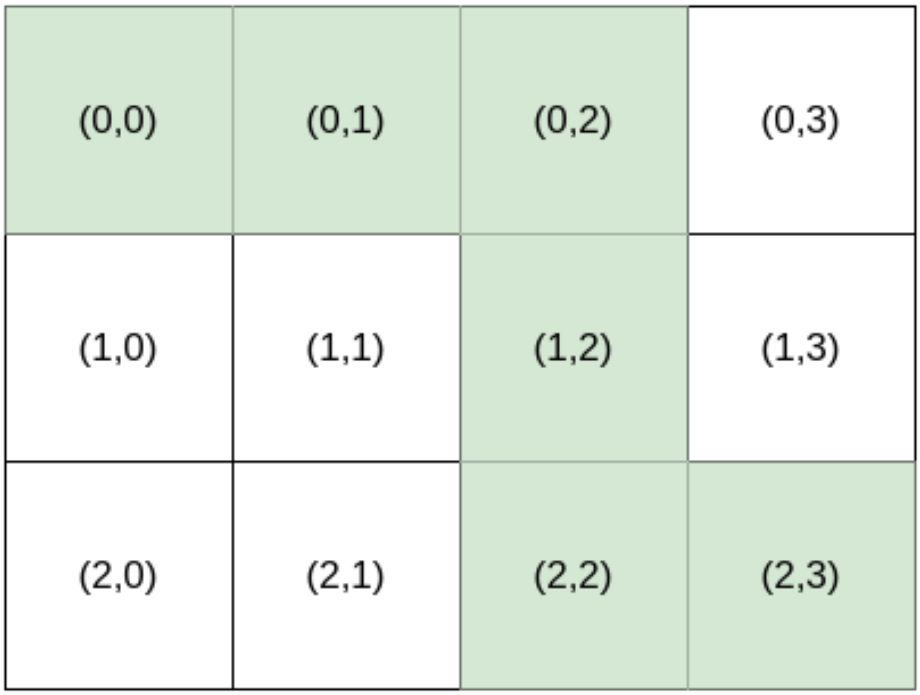
Input:

destination = [2,3], k = 2

Output:

"HHVHV"

Example 3:



Input:

destination = [2,3], k = 3

Output:

"HHVVH"

Constraints:

destination.length == 2

1 <= row, column <= 15

1 <= k <= nCr(row + column, row)

, where

nCr(a, b)

denotes

a

choose

b

.

Code Snippets

C++:

```
class Solution {
public:
string kthSmallestPath(vector<int>& destination, int k) {

}
};
```

Java:

```
class Solution {
public String kthSmallestPath(int[] destination, int k) {

}
}
```

Python3:

```
class Solution:
def kthSmallestPath(self, destination: List[int], k: int) -> str:
```

Python:

```
class Solution(object):
def kthSmallestPath(self, destination, k):
"""
:type destination: List[int]
:type k: int
:rtype: str
"""
```

JavaScript:

```
/**
* @param {number[]} destination
* @param {number} k
* @return {string}
*/
var kthSmallestPath = function(destination, k) {

};
```

TypeScript:

```
function kthSmallestPath(destination: number[], k: number): string {

};
```

C#:

```
public class Solution {
public string KthSmallestPath(int[] destination, int k) {

}
}
```

C:

```
char* kthSmallestPath(int* destination, int destinationSize, int k) {

}
```

Go:

```
func kthSmallestPath(destination []int, k int) string {

}
```

Kotlin:

```
class Solution {
fun kthSmallestPath(destination: IntArray, k: Int): String {

}
```

```
}
```

Swift:

```
class Solution {
func kthSmallestPath(_ destination: [Int], _ k: Int) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn kth_smallest_path(destination: Vec<i32>, k: i32) -> String {

}
}
```

Ruby:

```
# @param {Integer[]} destination
# @param {Integer} k
# @return {String}
def kth_smallest_path(destination, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $destination
* @param Integer $k
* @return String
*/
function kthSmallestPath($destination, $k) {

}
}
```

Dart:

```
class Solution {
String kthSmallestPath(List<int> destination, int k) {

}
}
```

Scala:

```
object Solution {
def kthSmallestPath(destination: Array[Int], k: Int): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec kth_smallest_path(destination :: [integer], k :: integer) :: String.t
def kth_smallest_path(destination, k) do

end
end
```

Erlang:

```
-spec kth_smallest_path(Destination :: [integer()], K :: integer()) ->
unicode:unicode_binary().
kth_smallest_path(Destination, K) ->
.
```

Racket:

```
(define/contract (kth-smallest-path destination k)
(-> (listof exact-integer?) exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Kth Smallest Instructions
* Difficulty: Hard
* Tags: array, string, graph, 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:
string kthSmallestPath(vector<int>& destination, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Kth Smallest Instructions
* Difficulty: Hard
* Tags: array, string, graph, 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 String kthSmallestPath(int[] destination, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Kth Smallest Instructions
Difficulty: Hard
Tags: array, string, graph, 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 kthSmallestPath(self, destination: List[int], k: int) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def kthSmallestPath(self, destination, k):
"""
:type destination: List[int]
:type k: int
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Kth Smallest Instructions
* Difficulty: Hard
* Tags: array, string, graph, 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[]} destination
* @param {number} k
* @return {string}
*/
var kthSmallestPath = function(destination, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Kth Smallest Instructions
* Difficulty: Hard
* Tags: array, string, graph, 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 kthSmallestPath(destination: number[], k: number): string {

};
```

C# Solution:

```
/*
* Problem: Kth Smallest Instructions
* Difficulty: Hard
* Tags: array, string, graph, 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 string KthSmallestPath(int[] destination, int k) {

}
}
```

C Solution:

```
/*
* Problem: Kth Smallest Instructions
* Difficulty: Hard
* Tags: array, string, graph, 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
*/

char* kthSmallestPath(int* destination, int destinationSize, int k) {

}
```

Go Solution:

```
// Problem: Kth Smallest Instructions
// Difficulty: Hard
// Tags: array, string, graph, 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 kthSmallestPath(destination []int, k int) string {

}
```

Kotlin Solution:

```
class Solution {
fun kthSmallestPath(destination: IntArray, k: Int): String {

}
}
```

Swift Solution:

```
class Solution {
func kthSmallestPath(_ destination: [Int], _ k: Int) -> String {

}
}
```

Rust Solution:

```
// Problem: Kth Smallest Instructions
// Difficulty: Hard
```

```
// Tags: array, string, graph, 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 kth_smallest_path(destination: Vec<i32>, k: i32) -> String {

}
}
```

Ruby Solution:

```
# @param {Integer[]} destination
# @param {Integer} k
# @return {String}
def kth_smallest_path(destination, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $destination
* @param Integer $k
* @return String
*/
function kthSmallestPath($destination, $k) {

}
}
```

Dart Solution:

```
class Solution {
String kthSmallestPath(List<int> destination, int k) {

}
```

```
}
```

Scala Solution:

```
object Solution {
def kthSmallestPath(destination: Array[Int], k: Int): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec kth_smallest_path(destination :: [integer], k :: integer) :: String.t
def kth_smallest_path(destination, k) do

end
end
```

Erlang Solution:

```
-spec kth_smallest_path(Destination :: [integer()], K :: integer()) ->
unicode:unicode_binary().
kth_smallest_path(Destination, K) ->
.
```

Racket Solution:

```
(define/contract (kth-smallest-path destination k)
(-> (listof exact-integer?) exact-integer? string?)
)
```