Problem 3332: Maximum Points Tourist Can Earn

Problem Information

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

Problem Description

You are given two integers,

n

and

k

, along with two 2D integer arrays,

stayScore

and

travelScore

.

A tourist is visiting a country with

n

cities, where each city is

directly

connected to every other city. The tourist's journey consists of

exactly

k

0-indexed

days, and they can choose

any

city as their starting point.

Each day, the tourist has two choices:

Stay in the current city

: If the tourist stays in their current city

curr

during day

i

, they will earn

stayScore[i][curr]

points.

Move to another city

: If the tourist moves from their current city

curr

to city

dest

, they will earn

travelScore[curr][dest]

points.

Return the

maximum

possible points the tourist can earn.

Example 1:

Input:

n = 2, k = 1, stayScore = [[2,3]], travelScore = [[0,2],[1,0]]

Output:

3

Explanation:

The tourist earns the maximum number of points by starting in city 1 and staying in that city.

Example 2:

Input:

n = 3, k = 2, stayScore = [[3,4,2],[2,1,2]], travelScore = [[0,2,1],[2,0,4],[3,2,0]]

Output:

8

Explanation:

The tourist earns the maximum number of points by starting in city 1, staying in that city on day 0, and traveling to city 2 on day 1.

Constraints:

1 <= n <= 200

1 <= k <= 200

n == travelScore.length == travelScore[i].length == stayScore[i].length

k == stayScore.length

1 <= stayScore[i][j] <= 100

0 <= travelScore[i][j] <= 100

travelScore[i][i] == 0

Code Snippets

C++:

```
class Solution {
public:
int maxScore(int n, int k, vector<vector<int>>& stayScore,
vector<vector<int>>& travelScore) {

}
};
```

Java:

```
class Solution {
public int maxScore(int n, int k, int[][] stayScore, int[][] travelScore) {

}
```

```
}
```

Python3:

```
class Solution:
def maxScore(self, n: int, k: int, stayScore: List[List[int]], travelScore:
List[List[int]]) -> int:
```

Python:

```
class Solution(object):
def maxScore(self, n, k, stayScore, travelScore):
"""
:type n: int
:type k: int
:type stayScore: List[List[int]]
:type travelScore: List[List[int]]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @param {number} k
* @param {number[][]} stayScore
* @param {number[][]} travelScore
* @return {number}
*/
var maxScore = function(n, k, stayScore, travelScore) {

};
```

TypeScript:

```
function maxScore(n: number, k: number, stayScore: number[][], travelScore:
number[][]): number {

};
```

C#:

```
public class Solution {
public int MaxScore(int n, int k, int[][] stayScore, int[][] travelScore) {

}
}
```

C:

```
int maxScore(int n, int k, int** stayScore, int stayScoreSize, int*
stayScoreColSize, int** travelScore, int travelScoreSize, int*
travelScoreColSize) {

}
```

Go:

```
func maxScore(n int, k int, stayScore [][]int, travelScore [][]int) int {

}
```

Kotlin:

```
class Solution {
fun maxScore(n: Int, k: Int, stayScore: Array<IntArray>, travelScore:
Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func maxScore(_ n: Int, _ k: Int, _ stayScore: [[Int]], _ travelScore:
[[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_score(n: i32, k: i32, stay_score: Vec<Vec<i32>>, travel_score:
Vec<Vec<i32>>) -> i32 {
```

```
}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} k
# @param {Integer[][]} stay_score
# @param {Integer[][]} travel_score
# @return {Integer}
def max_score(n, k, stay_score, travel_score)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $k
* @param Integer[][] $stayScore
* @param Integer[][] $travelScore
* @return Integer
*/
function maxScore($n, $k, $stayScore, $travelScore) {

}
}
```

Dart:

```
class Solution {
int maxScore(int n, int k, List<List<int>> stayScore, List<List<int>>
travelScore) {

}
}
```

Scala:

```
object Solution {
def maxScore(n: Int, k: Int, stayScore: Array[Array[Int]], travelScore:
Array[Array[Int]]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_score(n :: integer, k :: integer, stay_score :: [[integer]],
travel_score :: [[integer]]) :: integer
def max_score(n, k, stay_score, travel_score) do

end
end
```

Erlang:

```
-spec max_score(N :: integer(), K :: integer(), StayScore :: [[integer()]],
TravelScore :: [[integer()]]) -> integer().
max_score(N, K, StayScore, TravelScore) ->
.
```

Racket:

```
(define/contract (max-score n k stayScore travelScore)
(-> exact-integer? exact-integer? (listof (listof exact-integer?)) (listof
(listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Points Tourist Can Earn
* Difficulty: Medium
* 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 maxScore(int n, int k, vector<vector<int>>& stayScore,
vector<vector<int>>& travelScore) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Points Tourist Can Earn
* Difficulty: Medium
* 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 maxScore(int n, int k, int[][] stayScore, int[][] travelScore) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Points Tourist Can Earn
Difficulty: Medium
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 maxScore(self, n: int, k: int, stayScore: List[List[int]], travelScore:
List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maxScore(self, n, k, stayScore, travelScore):
"""
:type n: int
:type k: int
:type stayScore: List[List[int]]
:type travelScore: List[List[int]]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Maximum Points Tourist Can Earn
* Difficulty: Medium
* 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} n
* @param {number} k
* @param {number[][]} stayScore
* @param {number[][]} travelScore
* @return {number}
*/
var maxScore = function(n, k, stayScore, travelScore) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Points Tourist Can Earn
* Difficulty: Medium
* 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 maxScore(n: number, k: number, stayScore: number[][], travelScore:
number[][]): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Points Tourist Can Earn
* Difficulty: Medium
* 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 MaxScore(int n, int k, int[][] stayScore, int[][] travelScore) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Points Tourist Can Earn
* Difficulty: Medium
* 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 maxScore(int n, int k, int** stayScore, int stayScoreSize, int*
stayScoreColSize, int** travelScore, int travelScoreSize, int*
travelScoreColSize) {

}
```

Go Solution:

```
// Problem: Maximum Points Tourist Can Earn
// Difficulty: Medium
// 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 maxScore(n int, k int, stayScore [][]int, travelScore [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxScore(n: Int, k: Int, stayScore: Array<IntArray>, travelScore:
Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func maxScore(_ n: Int, _ k: Int, _ stayScore: [[Int]], _ travelScore:
[[Int]]) -> Int {

}
```

```
}
```

Rust Solution:

```
// Problem: Maximum Points Tourist Can Earn
// Difficulty: Medium
// 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 max_score(n: i32, k: i32, stay_score: Vec<Vec<i32>>, travel_score:
Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} k
# @param {Integer[][]} stay_score
# @param {Integer[][]} travel_score
# @return {Integer}
def max_score(n, k, stay_score, travel_score)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $k
* @param Integer[][] $stayScore
* @param Integer[][] $travelScore
* @return Integer
*/
```

```
function maxScore($n, $k, $stayScore, $travelScore) {

}
}
```

Dart Solution:

```
class Solution {
int maxScore(int n, int k, List<List<int>> stayScore, List<List<int>>
travelScore) {

}
}
```

Scala Solution:

```
object Solution {
def maxScore(n: Int, k: Int, stayScore: Array[Array[Int]], travelScore:
Array[Array[Int]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_score(n :: integer, k :: integer, stay_score :: [[integer]],
travel_score :: [[integer]]) :: integer
def max_score(n, k, stay_score, travel_score) do

end
end
```

Erlang Solution:

```
-spec max_score(N :: integer(), K :: integer(), StayScore :: [[integer()]],
TravelScore :: [[integer()]]) -> integer().
max_score(N, K, StayScore, TravelScore) ->
.
```

Racket Solution:

```
(define/contract (max-score n k stayScore travelScore)
(-> exact-integer? exact-integer? (listof (listof exact-integer?)) (listof
(listof exact-integer?)) exact-integer?)
)
```