Problem 1263: Minimum Moves to Move a Box to Their Target Location

Problem Information

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

Problem Description

A storekeeper is a game in which the player pushes boxes around in a warehouse trying to get them to target locations.

The game is represented by an

m x n

grid of characters

grid

where each element is a wall, floor, or box.

Your task is to move the box

'B'

to the target position

'T'

under the following rules:

The character

'S'

represents the player. The player can move up, down, left, right in

grid

if it is a floor (empty cell).

The character

'.'

represents the floor which means a free cell to walk.

The character

'#'

represents the wall which means an obstacle (impossible to walk there).

There is only one box

'B'

and one target cell

'T'

in the

grid

.

The box can be moved to an adjacent free cell by standing next to the box and then moving in the direction of the box. This is a

push

.

The player cannot walk through the box.

Return

the minimum number of

pushes

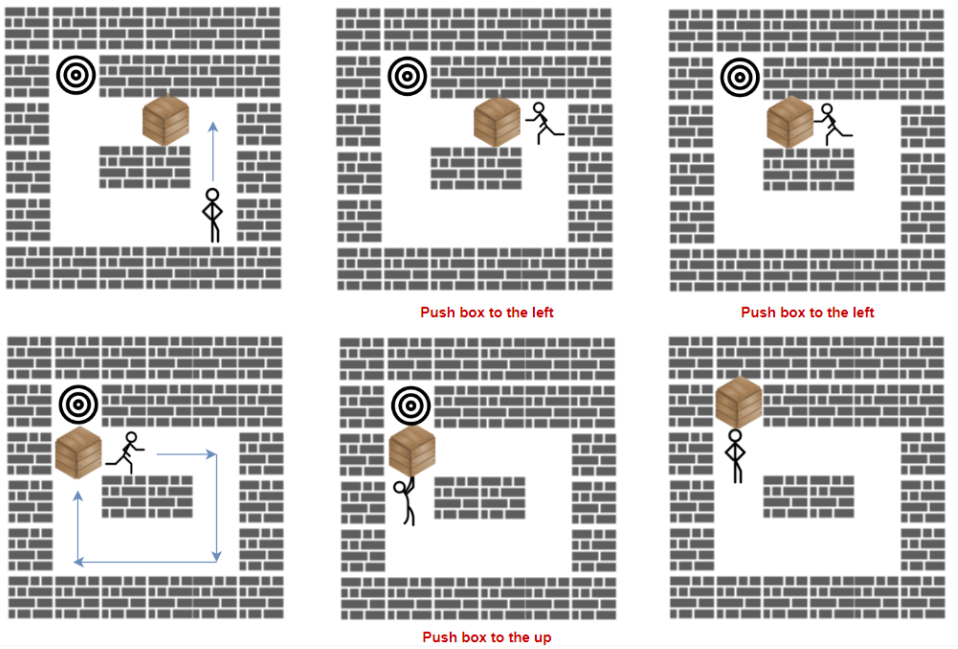
to move the box to the target

. If there is no way to reach the target, return

-1

.

Example 1:



Input:

grid = [["#","#","#","#","#","#"], ["#","T","#","#","#","#"], ["#",".",".","B",".","#"], ["#",".","#","#",".","#"], ["#",".",".",".",".","S","#"], ["#","#","#","#","#","#"]]

Output:

3

Explanation:

We return only the number of times the box is pushed.

Example 2:

Input:

grid = [["#","#","#","#","#","#"], ["#","T","#","#","#","#"], ["#",".",".","B",".","#"],
["#","#","#","#",".","#"], ["#",".",".",".","S","#"], ["#","#","#","#","#","#"]]

Output:

-1

Example 3:

Input:

grid = [["#","#","#","#","#","#"], ["#","T",".",".","#","#"], ["#",".","#","B",".","#"], ["#",".",".",".",".","#"],
["#",".",".",".","S","#"], ["#","#","#","#","#","#"]]

Output:

5

Explanation:

push the box down, left, left, up and up.

Constraints:

m == grid.length

n == grid[i].length

1 <= m, n <= 20

grid

contains only characters

'.'

,

'#'

,

'S'

,

'T'

, or

'B'

.

There is only one character

'S'

,

'B'

, and

'T'

in the

grid

.

Code Snippets

C++:

```
class Solution {
public:
int minPushBox(vector<vector<char>>& grid) {

}
};
```

Java:

```
class Solution {
public int minPushBox(char[][] grid) {

}
}
```

Python3:

```
class Solution:
def minPushBox(self, grid: List[List[str]]) -> int:
```

Python:

```
class Solution(object):
def minPushBox(self, grid):
"""
:type grid: List[List[str]]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {character[][]} grid
* @return {number}
*/
var minPushBox = function(grid) {

};
```

TypeScript:

```
function minPushBox(grid: string[][]): number {

};
```

C#:

```
public class Solution {
public int MinPushBox(char[][] grid) {

}
}
```

C:

```
int minPushBox(char** grid, int gridSize, int* gridColSize) {

}
```

Go:

```
func minPushBox(grid [][]byte) int {

}
```

Kotlin:

```
class Solution {
fun minPushBox(grid: Array<CharArray>): Int {

}
}
```

Swift:

```
class Solution {
func minPushBox(_ grid: [[Character]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_push_box(grid: Vec<Vec<char>>) -> i32 {

}
}
```

Ruby:

```
# @param {Character[][]} grid
# @return {Integer}
def min_push_box(grid)

end
```

PHP:

```
class Solution {

/**
* @param String[][] $grid
* @return Integer
*/
function minPushBox($grid) {

}
}
```

Dart:

```
class Solution {
int minPushBox(List<List<String>> grid) {

}
}
```

Scala:

```
object Solution {
def minPushBox(grid: Array[Array[Char]]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec min_push_box(grid :: [[char]]) :: integer
def min_push_box(grid) do

end
end
```

Erlang:

```
-spec min_push_box(Grid :: [[char()]]) -> integer().
min_push_box(Grid) ->
.
```

Racket:

```
(define/contract (min-push-box grid)
(-> (listof (listof char?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Moves to Move a Box to Their Target Location
* Difficulty: Hard
* Tags: array, tree, search, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/
```

```
class Solution {
public:
int minPushBox(vector<vector<char>>& grid) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Moves to Move a Box to Their Target Location
* Difficulty: Hard
* Tags: array, tree, search, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public int minPushBox(char[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Moves to Move a Box to Their Target Location
Difficulty: Hard
Tags: array, tree, search, queue, heap

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

class Solution:
def minPushBox(self, grid: List[List[str]]) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

```
class Solution(object):
def minPushBox(self, grid):
"""
:type grid: List[List[str]]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Moves to Move a Box to Their Target Location
* Difficulty: Hard
* Tags: array, tree, search, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* @param {character[][]} grid
* @return {number}
*/
var minPushBox = function(grid) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Moves to Move a Box to Their Target Location
* Difficulty: Hard
* Tags: array, tree, search, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
```

```
*/

function minPushBox(grid: string[][]): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Moves to Move a Box to Their Target Location
* Difficulty: Hard
* Tags: array, tree, search, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class Solution {
public int MinPushBox(char[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Moves to Move a Box to Their Target Location
* Difficulty: Hard
* Tags: array, tree, search, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

int minPushBox(char** grid, int gridSize, int* gridColSize) {

}
```

Go Solution:

```
// Problem: Minimum Moves to Move a Box to Their Target Location
// Difficulty: Hard
// Tags: array, tree, search, queue, heap
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func minPushBox(grid [][]byte) int {

}
```

Kotlin Solution:

```
class Solution {
fun minPushBox(grid: Array<CharArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func minPushBox(_ grid: [[Character]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Moves to Move a Box to Their Target Location
// Difficulty: Hard
// Tags: array, tree, search, queue, heap
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
pub fn min_push_box(grid: Vec<Vec<char>>) -> i32 {

}
```

```
}
```

Ruby Solution:

```
# @param {Character[][]} grid
# @return {Integer}
def min_push_box(grid)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[][] $grid
* @return Integer
*/
function minPushBox($grid) {

}
}
```

Dart Solution:

```
class Solution {
int minPushBox(List<List<String>> grid) {

}
}
```

Scala Solution:

```
object Solution {
def minPushBox(grid: Array[Array[Char]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_push_box(grid :: [[char]]) :: integer
def min_push_box(grid) do

end
end
```

Erlang Solution:

```
-spec min_push_box(Grid :: [[char()]]) -> integer().
min_push_box(Grid) ->
.
```

Racket Solution:

```
(define/contract (min-push-box grid)
(-> (listof (listof char?)) exact-integer?)
)
```