Problem 3143: Maximum Points Inside the Square

Problem Information

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

Problem Description

You are given a 2D

array

points

and a string

s

where,

points[i]

represents the coordinates of point

i

, and

s[i]

represents the

tag

of point

i

.

A

valid

square is a square centered at the origin

(0, 0)

, has edges parallel to the axes, and

does not

contain two points with the same tag.

Return the

maximum

number of points contained in a

valid

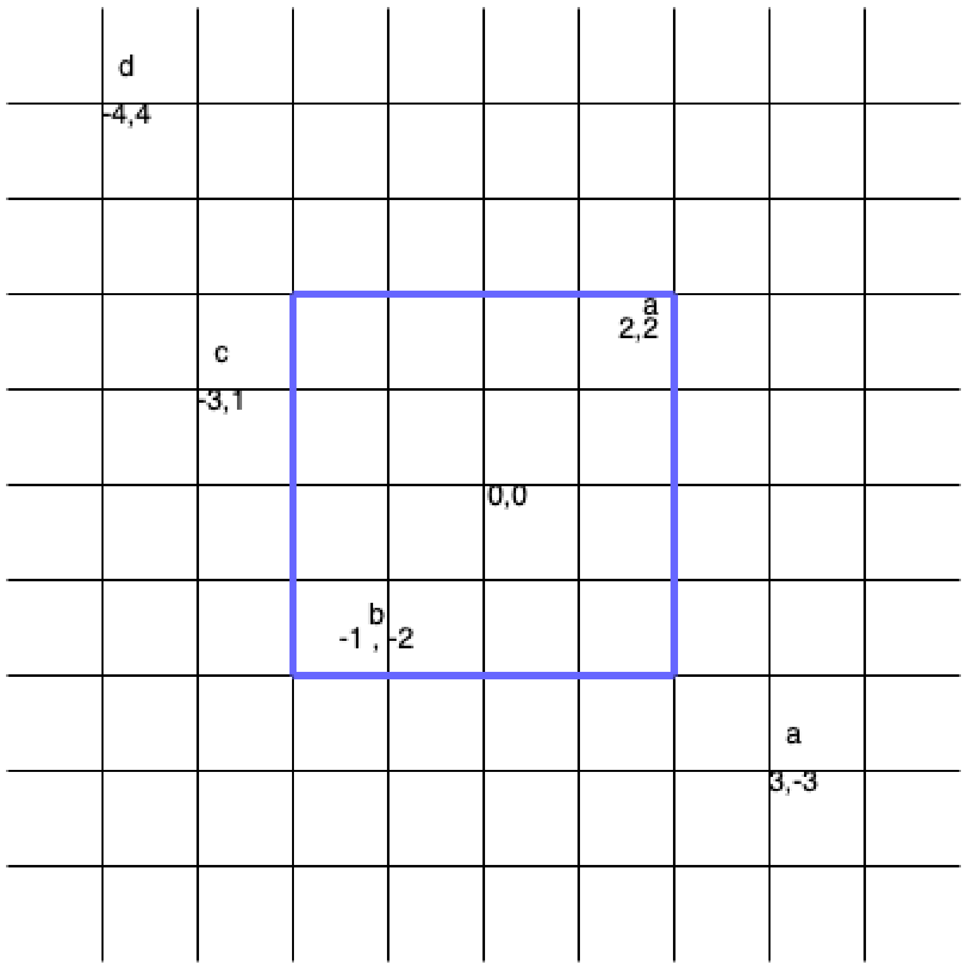
square.

Note:

A point is considered to be inside the square if it lies on or within the square's boundaries.

The side length of the square can be zero.

Example 1:



Input:

points = [[2,2],[-1,-2],[-4,4],[-3,1],[3,-3]], s = "abdca"

Output:

2

Explanation:

The square of side length 4 covers two points

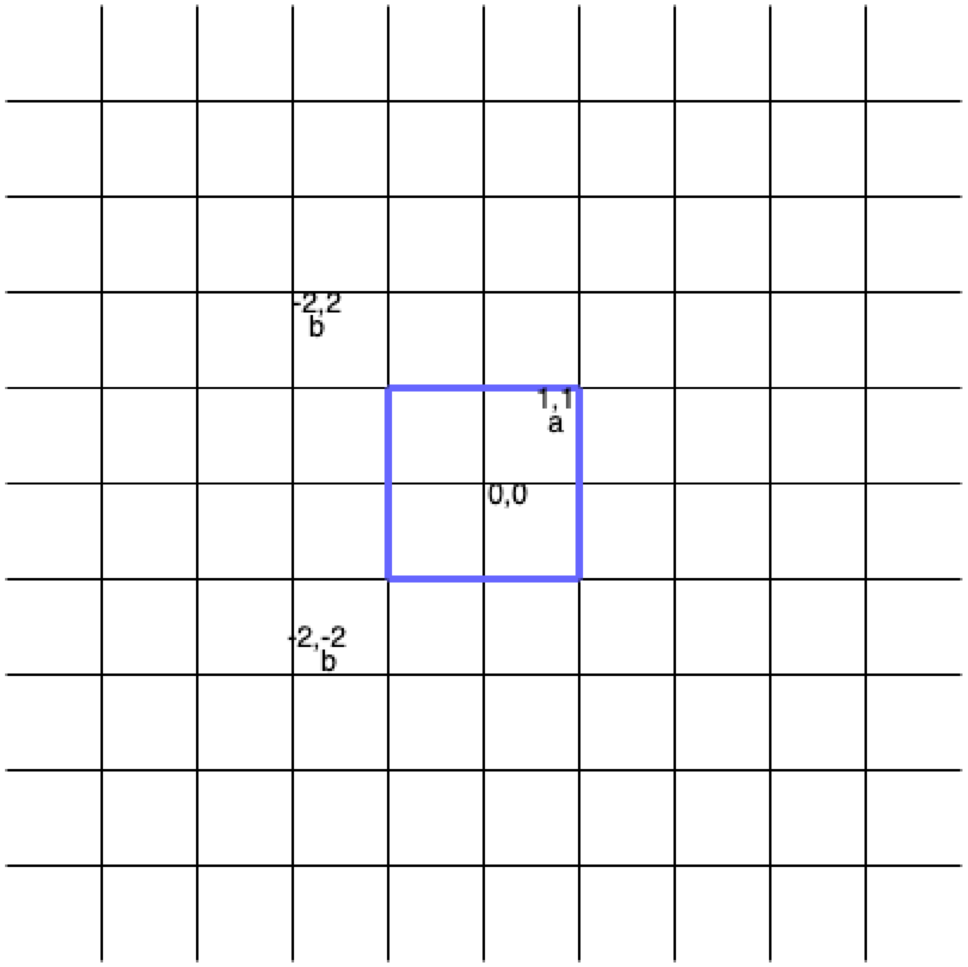
points[0]

and

points[1]

.

Example 2:



Input:

points = [[1,1],[-2,-2],[-2,2]], s = "abb"

Output:

1

Explanation:

The square of side length 2 covers one point, which is

points[0]

.

Example 3:

Input:

points = [[1,1],[-1,-1],[2,-2]], s = "ccd"

Output:

0

Explanation:

It's impossible to make any valid squares centered at the origin such that it covers only one point among

points[0]

and

points[1]

.

Constraints:

1 <= s.length, points.length <= 10

5

points[i].length == 2

-10

9

<= points[i][0], points[i][1] <= 10

9

s.length == points.length

points

consists of distinct coordinates.

s

consists only of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int maxPointsInsideSquare(vector<vector<int>>& points, string s) {

}
};
```

Java:

```
class Solution {
public int maxPointsInsideSquare(int[][] points, String s) {

}
}
```

Python3:

```
class Solution:
def maxPointsInsideSquare(self, points: List[List[int]], s: str) -> int:
```

Python:

```
class Solution(object):
def maxPointsInsideSquare(self, points, s):
```

```
"""
:type points: List[List[int]]
:type s: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[][]} points
* @param {string} s
* @return {number}
*/
var maxPointsInsideSquare = function(points, s) {

};
```

TypeScript:

```
function maxPointsInsideSquare(points: number[][], s: string): number {

};
```

C#:

```
public class Solution {
public int MaxPointsInsideSquare(int[][] points, string s) {

}
}
```

C:

```
int maxPointsInsideSquare(int** points, int pointsSize, int* pointsColSize,
char* s) {

}
```

Go:

```
func maxPointsInsideSquare(points [][]int, s string) int {

}
```

Kotlin:

```
class Solution {
fun maxPointsInsideSquare(points: Array<IntArray>, s: String): Int {

}
}
```

Swift:

```
class Solution {
func maxPointsInsideSquare(_ points: [[Int]], _ s: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_points_inside_square(points: Vec<Vec<i32>>, s: String) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[][]} points
# @param {String} s
# @return {Integer}
def max_points_inside_square(points, s)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $points
* @param String $s
* @return Integer
*/
function maxPointsInsideSquare($points, $s) {

}
}
```

Dart:

```
class Solution {
int maxPointsInsideSquare(List<List<int>> points, String s) {

}
}
```

Scala:

```
object Solution {
def maxPointsInsideSquare(points: Array[Array[Int]], s: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_points_inside_square(points :: [[integer]], s :: String.t) ::
integer
def max_points_inside_square(points, s) do

end
end
```

Erlang:

```
-spec max_points_inside_square(Points :: [[integer()]], S ::
unicode:unicode_binary()) -> integer().
max_points_inside_square(Points, S) ->
.
```

Racket:

```
(define/contract (max-points-inside-square points s)
(-> (listof (listof exact-integer?)) string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Points Inside the Square
* Difficulty: Medium
* Tags: array, string, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int maxPointsInsideSquare(vector<vector<int>>& points, string s) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Points Inside the Square
* Difficulty: Medium
* Tags: array, string, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int maxPointsInsideSquare(int[][] points, String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Points Inside the Square
Difficulty: Medium
Tags: array, string, hash, sort, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def maxPointsInsideSquare(self, points: List[List[int]], s: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maxPointsInsideSquare(self, points, s):
"""
:type points: List[List[int]]
:type s: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Maximum Points Inside the Square
* Difficulty: Medium
* Tags: array, string, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
```

```
/**
* @param {number[][]} points
* @param {string} s
* @return {number}
*/
var maxPointsInsideSquare = function(points, s) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Points Inside the Square
* Difficulty: Medium
* Tags: array, string, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function maxPointsInsideSquare(points: number[][], s: string): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Points Inside the Square
* Difficulty: Medium
* Tags: array, string, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int MaxPointsInsideSquare(int[][] points, string s) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Points Inside the Square
* Difficulty: Medium
* Tags: array, string, hash, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int maxPointsInsideSquare(int** points, int pointsSize, int* pointsColSize,
char* s) {

}
```

Go Solution:

```
// Problem: Maximum Points Inside the Square
// Difficulty: Medium
// Tags: array, string, hash, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func maxPointsInsideSquare(points [][]int, s string) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxPointsInsideSquare(points: Array<IntArray>, s: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func maxPointsInsideSquare(_ points: [[Int]], _ s: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum Points Inside the Square
// Difficulty: Medium
// Tags: array, string, hash, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn max_points_inside_square(points: Vec<Vec<i32>>, s: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} points
# @param {String} s
# @return {Integer}
def max_points_inside_square(points, s)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $points
* @param String $s
* @return Integer
*/
function maxPointsInsideSquare($points, $s) {

}
}
```

Dart Solution:

```
class Solution {
int maxPointsInsideSquare(List<List<int>> points, String s) {

}
}
```

Scala Solution:

```
object Solution {
def maxPointsInsideSquare(points: Array[Array[Int]], s: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_points_inside_square(points :: [[integer]], s :: String.t) ::
integer
def max_points_inside_square(points, s) do

end
end
```

Erlang Solution:

```
-spec max_points_inside_square(Points :: [[integer()]], S ::
unicode:unicode_binary()) -> integer().
max_points_inside_square(Points, S) ->
.
```

Racket Solution:

```
(define/contract (max-points-inside-square points s)
(-> (listof (listof exact-integer?)) string? exact-integer?)
)
```