# Problem 2280: Minimum Lines to Represent a Line Chart

## Problem Information

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

## Problem Description

You are given a 2D integer array

stockPrices

where

stockPrices[i] = [day

i

, price

i

]

indicates the price of the stock on day

day

i

is

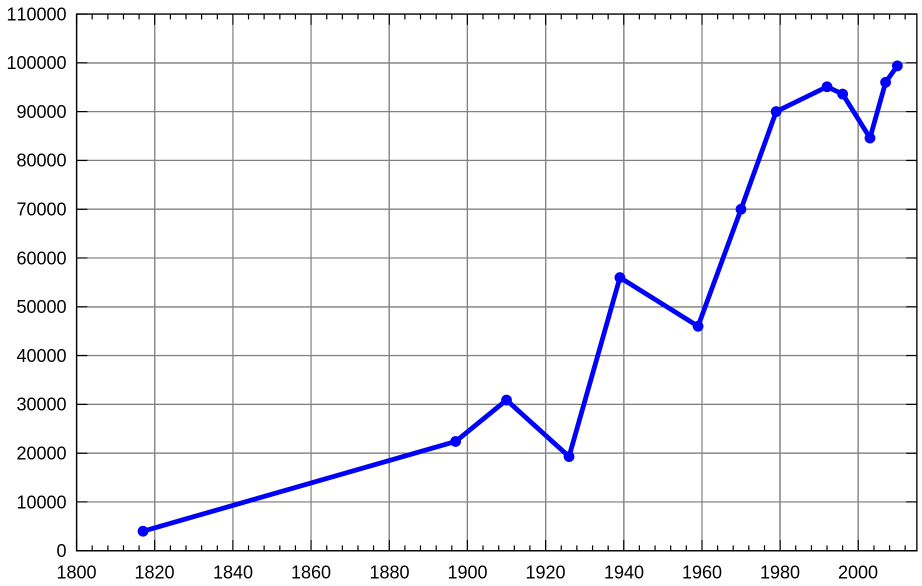
price

i

. A

line chart

is created from the array by plotting the points on an XY plane with the X-axis representing the day and the Y-axis representing the price and connecting adjacent points. One such example is shown below:




Return

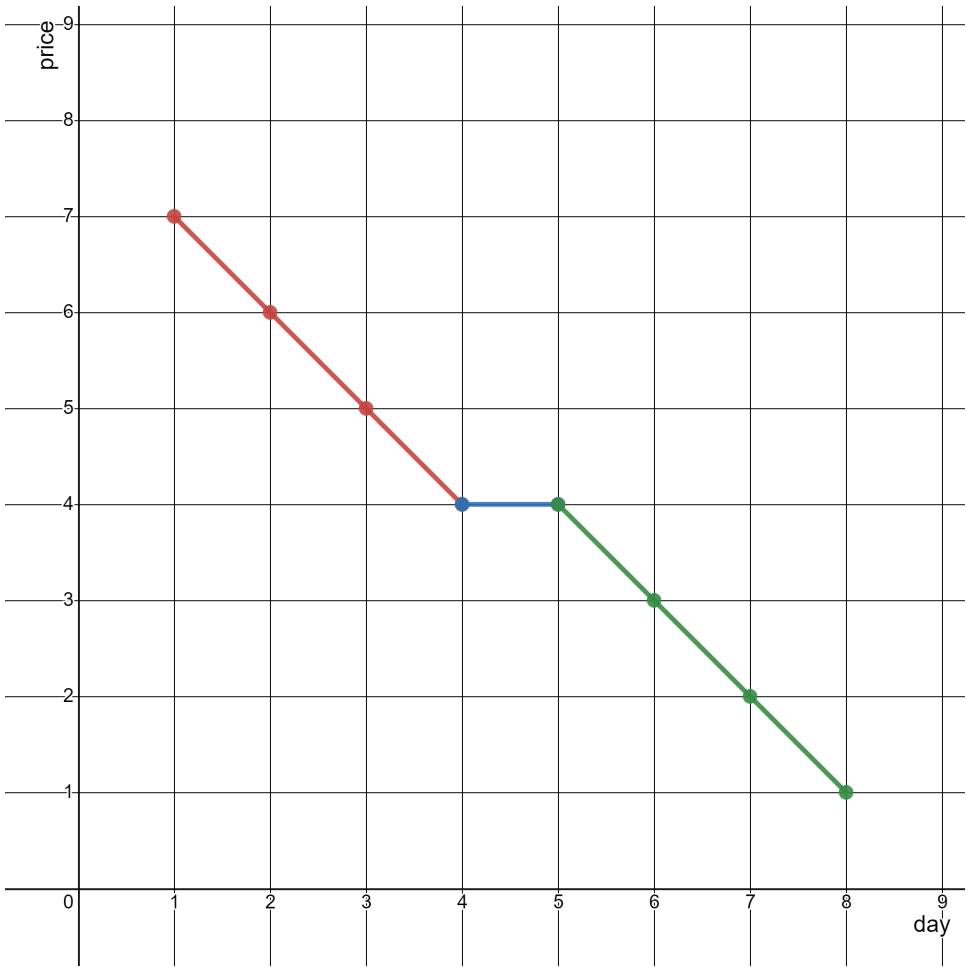
the

minimum number of lines

needed to represent the line chart

.

Example 1:




Input:

stockPrices = [[1,7],[2,6],[3,5],[4,4],[5,4],[6,3],[7,2],[8,1]]

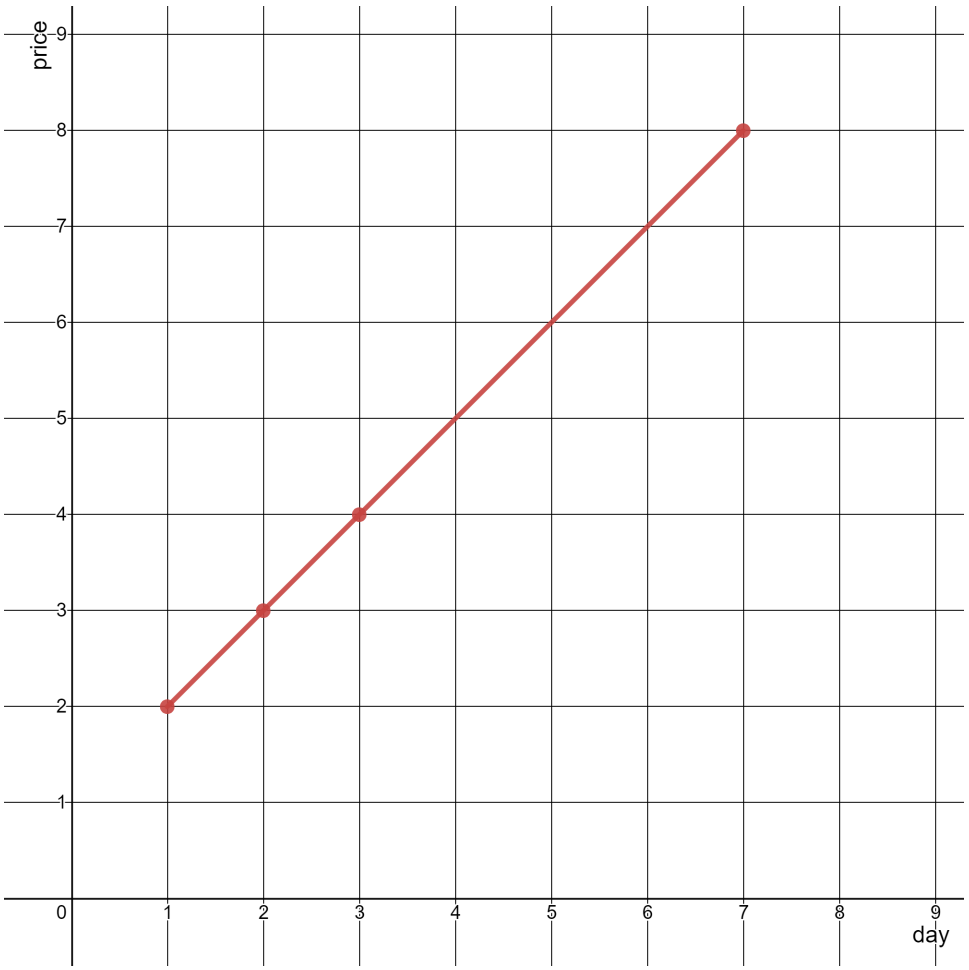
Output:

3

Explanation:

The diagram above represents the input, with the X-axis representing the day and Y-axis representing the price. The following 3 lines can be drawn to represent the line chart: - Line 1 (in red) from (1,7) to (4,4) passing through (1,7), (2,6), (3,5), and (4,4). - Line 2 (in blue) from (4,4) to (5,4). - Line 3 (in green) from (5,4) to (8,1) passing through (5,4), (6,3), (7,2), and (8,1). It can be shown that it is not possible to represent the line chart using less than 3 lines.

Example 2:




Input:

stockPrices = [[3,4],[1,2],[7,8],[2,3]]

Output:

1

Explanation:

As shown in the diagram above, the line chart can be represented with a single line.

Constraints:

1 <= stockPrices.length <= 10

5

stockPrices[i].length == 2

1 <= day

i

, price

i

<= 10

9

All

day

i

are

distinct

.

## Code Snippets

**C++:**

```
class Solution {
public:
int minimumLines(vector<vector<int>>& stockPrices) {

}
};
```

**Java:**

```
class Solution {
public int minimumLines(int[][] stockPrices) {

}
}
```

**Python3:**

```
class Solution:
def minimumLines(self, stockPrices: List[List[int]]) -> int:
```

**Python:**

```
class Solution(object):
def minimumLines(self, stockPrices):
"""
:type stockPrices: List[List[int]]
:rtype: int
"""
```

**JavaScript:**

```
/**
* @param {number[][]} stockPrices
* @return {number}
*/
var minimumLines = function(stockPrices) {

};
```

**TypeScript:**

```
function minimumLines(stockPrices: number[][]): number {

};
```

**C#:**

```
public class Solution {
public int MinimumLines(int[][] stockPrices) {

}
}
```

**C:**

```
int minimumLines(int** stockPrices, int stockPricesSize, int*
stockPricesColSize) {

}
```

**Go:**

```
func minimumLines(stockPrices [][]int) int {

}
```

**Kotlin:**

```
class Solution {
fun minimumLines(stockPrices: Array<IntArray>): Int {

}
}
```

**Swift:**

```
class Solution {
func minimumLines(_ stockPrices: [[Int]]) -> Int {

}
}
```

**Rust:**

```
impl Solution {
pub fn minimum_lines(stock_prices: Vec<Vec<i32>>) -> i32 {

}
}
```

**Ruby:**

```
# @param {Integer[][]} stock_prices
# @return {Integer}
def minimum_lines(stock_prices)
```

```
end
```

**PHP:**

```
class Solution {

/**
* @param Integer[][] $stockPrices
* @return Integer
*/
function minimumLines($stockPrices) {

}
}
```

**Dart:**

```
class Solution {
int minimumLines(List<List<int>> stockPrices) {

}
}
```

**Scala:**

```
object Solution {
def minimumLines(stockPrices: Array[Array[Int]]): Int = {

}
}
```

**Elixir:**

```
defmodule Solution do
@spec minimum_lines(stock_prices :: [[integer]]) :: integer
def minimum_lines(stock_prices) do

end
end
```

**Erlang:**

```
-spec minimum_lines(StockPrices :: [[integer()]]) -> integer().
minimum_lines(StockPrices) ->
.
```

**Racket:**

```
(define/contract (minimum-lines stockPrices)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Minimum Lines to Represent a Line Chart
* Difficulty: Medium
* Tags: array, math, sort
*
* 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 minimumLines(vector<vector<int>>& stockPrices) {

}
};
```

### Java Solution:

```
/**
* Problem: Minimum Lines to Represent a Line Chart
* Difficulty: Medium
* Tags: array, math, sort
*
* 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 minimumLines(int[][] stockPrices) {

}
}
```

### Python3 Solution:

```
"""
Problem: Minimum Lines to Represent a Line Chart
Difficulty: Medium
Tags: array, math, sort

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 minimumLines(self, stockPrices: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

### Python Solution:

```
class Solution(object):
def minimumLines(self, stockPrices):
"""
:type stockPrices: List[List[int]]
:rtype: int
"""
```

### JavaScript Solution:

```
/**
* Problem: Minimum Lines to Represent a Line Chart
* Difficulty: Medium
* Tags: array, math, sort
*
```

```
* 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[][]} stockPrices
* @return {number}
*/
var minimumLines = function(stockPrices) {

};
```

### TypeScript Solution:

```
/**
* Problem: Minimum Lines to Represent a Line Chart
* Difficulty: Medium
* Tags: array, math, sort
*
* 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 minimumLines(stockPrices: number[][]): number {

};
```

### C# Solution:

```
/*
* Problem: Minimum Lines to Represent a Line Chart
* Difficulty: Medium
* Tags: array, math, sort
*
* 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 MinimumLines(int[][] stockPrices) {

}
}
```

### C Solution:

```
/*
* Problem: Minimum Lines to Represent a Line Chart
* Difficulty: Medium
* Tags: array, math, sort
*
* 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 minimumLines(int** stockPrices, int stockPricesSize, int*
stockPricesColSize) {

}
```

### Go Solution:

```
// Problem: Minimum Lines to Represent a Line Chart
// Difficulty: Medium
// Tags: array, math, sort
//
// 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 minimumLines(stockPrices [][]int) int {

}
```

### Kotlin Solution:

```
class Solution {
fun minimumLines(stockPrices: Array<IntArray>): Int {
```

```
}
}
```

### Swift Solution:

```
class Solution {
func minimumLines(_ stockPrices: [[Int]]) -> Int {

}
}
```

### Rust Solution:

```
// Problem: Minimum Lines to Represent a Line Chart
// Difficulty: Medium
// Tags: array, math, sort
//
// 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_lines(stock_prices: Vec<Vec<i32>>) -> i32 {

}
}
```

### Ruby Solution:

```
# @param {Integer[][]} stock_prices
# @return {Integer}
def minimum_lines(stock_prices)

end
```

### PHP Solution:

```
class Solution {

/**
```

```
* @param Integer[][] $stockPrices
* @return Integer
*/
function minimumLines($stockPrices) {

}
}
```

**Dart Solution:**

```
class Solution {
int minimumLines(List<List<int>> stockPrices) {

}
}
```

**Scala Solution:**

```
object Solution {
def minimumLines(stockPrices: Array[Array[Int]]): Int = {

}
}
```

**Elixir Solution:**

```
defmodule Solution do
@spec minimum_lines(stock_prices :: [[integer]]) :: integer
def minimum_lines(stock_prices) do

end
end
```

**Erlang Solution:**

```
-spec minimum_lines(StockPrices :: [[integer()]]) -> integer().
minimum_lines(StockPrices) ->
.
```

**Racket Solution:**

```
(define/contract (minimum-lines stockPrices)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```