Problem 2373: Largest Local Values in a Matrix

Problem Information

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

Problem Description

You are given an

n x n

integer matrix

grid

.

Generate an integer matrix

maxLocal

of size

(n - 2) x (n - 2)

such that:

maxLocal[i][j]

is equal to the

largest

value of the

3 x 3

matrix in

grid

centered around row

i + 1

and column

j + 1

.

In other words, we want to find the largest value in every contiguous

3 x 3

matrix in

grid

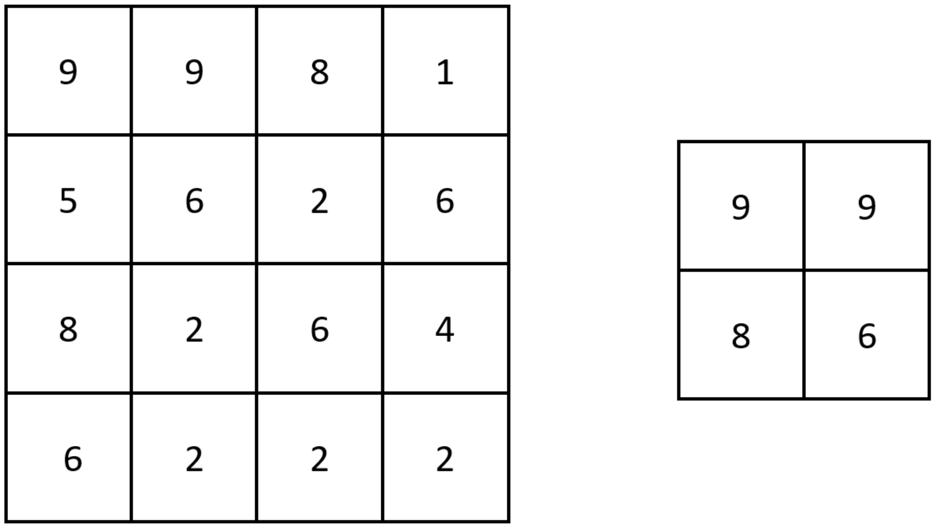
.

Return

the generated matrix

.

Example 1:



Input:

grid = [[9,9,8,1],[5,6,2,6],[8,2,6,4],[6,2,2,2]]

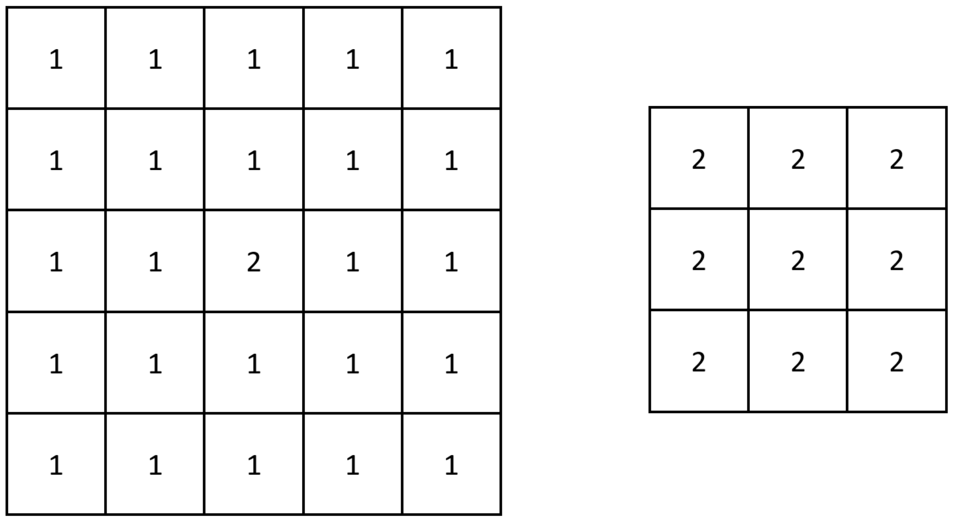
Output:

[[9,9],[8,6]]

Explanation:

The diagram above shows the original matrix and the generated matrix. Notice that each value in the generated matrix corresponds to the largest value of a contiguous 3 x 3 matrix in grid.

Example 2:



1	1	1	1	1
1	1	1	1	1
1	1	2	1	1
1	1	1	1	1
1	1	1	1	1

2	2	2
2	2	2
2	2	2

Input:

grid = [[1,1,1,1,1],[1,1,1,1,1],[1,1,2,1,1],[1,1,1,1,1],[1,1,1,1,1]]

Output:

[[2,2,2],[2,2,2],[2,2,2]]

Explanation:

Notice that the 2 is contained within every contiguous 3 x 3 matrix in grid.

Constraints:

n == grid.length == grid[i].length

3 <= n <= 100

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

Code Snippets

C++:

```
class Solution {
public:
```

```
vector<vector<int>> largestLocal(vector<vector<int>>& grid) {

}
};
```

Java:

```
class Solution {
public int[][] largestLocal(int[][] grid) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function largestLocal(grid: number[][]): number[][] {

};
```

C#:

```
public class Solution {
public int[][] LargestLocal(int[][] grid) {

}
}
```

C:

```
/**
* Return an array of arrays of size *returnSize.
* The sizes of the arrays are returned as *returnColumnSizes array.
* Note: Both returned array and *columnSizes array must be malloced, assume
caller calls free().
*/
int** largestLocal(int** grid, int gridSize, int* gridColSize, int*
returnSize, int** returnColumnSizes) {

}
```

Go:

```
func largestLocal(grid [][]int) [][]int {

}
```

Kotlin:

```
class Solution {
fun largestLocal(grid: Array<IntArray>): Array<IntArray> {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

```
class Solution {
List<List<int>> largestLocal(List<List<int>> grid) {

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (largest-local grid)
(-> (listof (listof exact-integer?)) (listof (listof exact-integer?)))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Largest Local Values in a Matrix
* Difficulty: Easy
* Tags: array
*
* 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:
vector<vector<int>> largestLocal(vector<vector<int>>& grid) {
}
};
```

Java Solution:

```
/**
* Problem: Largest Local Values in a Matrix
* Difficulty: Easy
* Tags: array
*
* 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[][] largestLocal(int[][] grid) {

}
}
```

Python3 Solution:

```
"""
Problem: Largest Local Values in a Matrix
Difficulty: Easy
Tags: array

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Largest Local Values in a Matrix
* Difficulty: Easy
* Tags: array
*
* 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[][]} grid
* @return {number[][]}
*/
var largestLocal = function(grid) {

};
```

TypeScript Solution:

```
/**
* Problem: Largest Local Values in a Matrix
* Difficulty: Easy
* Tags: array
*
* 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 largestLocal(grid: number[][]): number[][] {

};
```

C# Solution:

```
/*
* Problem: Largest Local Values in a Matrix
* Difficulty: Easy
* Tags: array
*
* 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[][] LargestLocal(int[][] grid) {

}
}
```

C Solution:

```
/*
* Problem: Largest Local Values in a Matrix
* Difficulty: Easy
* Tags: array
*
* 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
*/

/**
* Return an array of arrays of size *returnSize.
* The sizes of the arrays are returned as *returnColumnSizes array.
* Note: Both returned array and *columnSizes array must be malloced, assume
caller calls free().
*/
int** largestLocal(int** grid, int gridSize, int* gridColSize, int*
returnSize, int** returnColumnSizes) {

}
```

Go Solution:

```
// Problem: Largest Local Values in a Matrix
// Difficulty: Easy
// Tags: array
//
// 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 largestLocal(grid [][]int) [][]int {

}
```

Kotlin Solution:

```
class Solution {
fun largestLocal(grid: Array<IntArray>): Array<IntArray> {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Largest Local Values in a Matrix
// Difficulty: Easy
// Tags: array
//
// 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 largest_local(grid: Vec<Vec<i32>>) -> Vec<Vec<i32>> {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

```
class Solution {
List<List<int>> largestLocal(List<List<int>> grid) {

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (largest-local grid)
(-> (listof (listof exact-integer?)) (listof (listof exact-integer?)))
)
```