1. Server Selection

Some developers want to deploy their application on different servers with a load balancer in the front. There are *n* servers to choose from where the number of requests that can be handled by the i^{th} server is *server[i].* The number of requests served by any server is a power of 2 i.e. 1, 2, 4, 8, 16,...etc.

Given the array *server* and an integer *expected_load,* find the minimum number of servers that must be chosen such that the total sum of requests served by all the chosen servers is exactly equal to the *expected_load.* If there is no combination of servers that can serve exactly *expected_load* requests, report -1 as the answer.

Example
Suppose *n = 4, servers = [1, 1, 2, 4],* and *expected_load = 3.*

It is optimal to choose the first and the third or the second and the third servers serving a total of 1 + 2 = *expected_load* = 3 requests. Return the minimum number of servers needed, 2.

Function Description:
Complete the function *getMinServers* in the editor below.

The function *getMinServers* has the following parameter:
int expected_load: the number of requests to be served
int server[n]: the number of requests the servers can serve

Return
int: the minimum number of servers such that the sum of the total requests they can serve is exactly *expected_load*

Constraints:

- $1 \leq n \leq 10^5$
- $1 \leq server[i] \leq 10^9$
- It is guaranteed that *server[i]* is a power of 2.
- $1 \leq expected\_load \leq 10^9$

Input Format For Custom Testing

- The first line contains an integer, *expected_load*.
- The next line contains an integer, *n*, the total number of elements.
- The next *n* lines contain an integer, *server[i]*.

Sample Case 0

Sample Input For Custom Testing

```
STDIN       FUNCTION
-----       ----------
10      →   expected_load = 10
5       →   n = 5
1       →   server = [1, 1, 2, 4, 4]
1
2
4
4
```

Sample Output

```
3
```

Explanation

It is optimal to choose the last three servers to serve a total number of requests as 2 + 4 + 4 = 10.

Sample Case 1

Sample Input For Custom Testing

```
STDIN       FUNCTION
-----       ----------
4       →   expected_load = 4
3       →   n = 3
1       →   server = [1, 1, 1]
1
1
```

Sample Output

```
-1
```

Explanation

There is no selection of servers that can serve a total number of requests equal to 4.

```
package main
```

```go
import (
	"fmt"
	"sort"
)

func getMinServers(expectedLoad int, servers []int) int {
	n := len(servers)
	sort.Ints(servers)

	count := 0
	sum := 0
	for reza := n - 1; reza >= 0; reza-- {
		sum += servers[reza]
		count++
		if sum >= expectedLoad {
			return count
		}
	}

	return -1
}

func main() {
	var expectedLoad, n int
	fmt.Scan(&expectedLoad)

	fmt.Scan(&n)

	servers := make([]int, n)
	for i := 0; i < n; i++ {
		fmt.Scan(&servers[i])
	}

	minServers := getMinServers(expectedLoad, servers)
	if minServers == -1 {
		fmt.Println(-1)
	} else {
		fmt.Printf("%d\n", minServers)
	}
}
```

2. Maximize the Value

Rearrange an array of integers so that the calculated value U is maximized. Among the arrangements that satisfy that test, choose the array with minimal ordering. The value of U for an array with n elements is calculated as :

$U = arr[1] \times arr[2] \times (1 \div arr[3]) \times arr[4] \times \ldots \times arr[n-1] \times (1 \div arr[n])$ if n is odd
or
$U = arr[1] \times arr[2] \times (1 \div arr[3]) \times arr[4] \times \ldots \times (1 \div arr[n-1]) \times arr[n]$ if n is even

The sequence of operations is the same in either case, but the length of the array, n, determines whether the calculation ends on $arr[n]$ or $(1 \div arr[n])$.

Arrange the elements to maximize U so the items are in the numerically smallest possible order.

Example
$arr = [21, 34, 5, 7, 9]$

To maximize U and minimize the order, arrange the array as $[9, 21, 5, 34, 7]$ so $U = 9 \times 21 \times (1 \div 5) \times 34 \times (1 \div 7) = 183.6$. The same U can be achieved using several other orders, e.g. $[21, 9, 7, 34, 5] = 21 \times 9 \times (1 \div 7) \times 34 \times (1 \div 5) = 183.6$, but they are not in the minimum order.

Function Description
Complete the function *rearrange* in the editor.

rearrange has the following parameter(s):
int arr[n]: an array of integers

Returns
int[n]: the elements of *arr* rearranged as described

Constraints

- $1 \le n \le 10^5$
- $1 \le arr[i] \le 10^9$

input Format For Custom Testing
The first line contains an integer, n, the number of elements in *arr*.
Each line i of the n subsequent lines (*where* $1 \le i \le n$) contains an integer, $arr[i]$.

Sample Case 0

Sample Input For Custom Testing

```
STDIN    Function
-----    --------
4    →   arr[] size n = 4
1    →   arr = [1, 2, 3, 4]
2
3
4
```

Sample Output

```
2
3
1
4
```

Explanation

$U = 2\times3\times(1\div1)\times4 = 24$. All other arrangements where $U = 24$ are numerically higher than this array, e.g. *[2, 3, 1, 4] < [3, 4, 1, 2]*.

Sample Case 1

Sample Input For Custom Testing

```
STDIN    Function
-----    --------
2    →   arr[] size n = 2
4    →   arr = [4, 5]
5
```

Sample Output

```
4
5
```

Explanation

U is always $4\times5 = 20$, and *[4, 5] < [5, 4]*.

```go
package main

import (
    "bufio"
    "fmt"
    "os"
    "sort"
    "strconv"
    "strings"
)

func rearrange(arr []int) []int {
    n := len(arr)

    // Sort the array in ascending order
    sort.Ints(arr)

    // Rearrange the array by alternating the smallest and largest e
lements
    for i := 0; i < n/2; i++ {
        j := n - 1 - i*2
        arr[i], arr[j] = arr[j], arr[i]
    }

    return arr
}

func main() {
    // Read input from stdin
    reader := bufio.NewReader(os.Stdin)

    // Read the size of the array
    input, _ := reader.ReadString('\n')
    input = strings.TrimSpace(input)
    n, _ := strconv.Atoi(input)

    // Read the elements of the array
    arr := make([]int, n)
    for i := 0; i < n; i++ {
        input, _ := reader.ReadString('\n')
        input = strings.TrimSpace(input)
        arr[i], _ = strconv.Atoi(input)
    }

    // Rearrange the array
    arr = rearrange(arr)

    // Calculate U
```

```go
    u := float64(arr[0])
    for i := 1; i < n; i++ {
        if i%2 == 1 {
            u *= float64(arr[i])
        } else {
            u *= float64(arr[i]) / float64(arr[i-1])
        }
    }

    // Print the rearranged array
    for _, x := range arr {
        fmt.Println(x)
    }

    // Print the value of U
    fmt.Println(u)
}
```

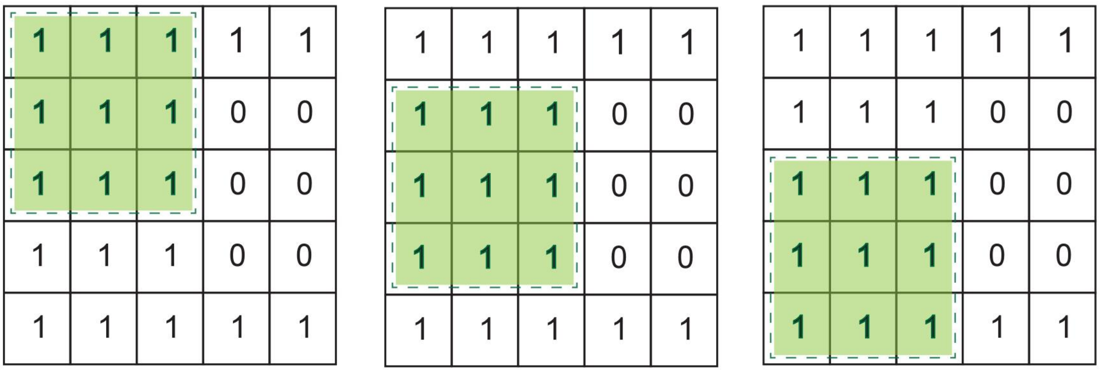
3. Product Defects

A quality agent is responsible for inspecting samples of the finished products in the production line. Each sample contains defective and non-defective products represented by 1 and 0 respectively. The product samples are placed sequentially in a two-dimensional square matrix. The goal is to determine the size of the largest square of defective products in the two-dimensional square matrix.

Example

n x n = 5 x 5 matrix of product samples

samples = [[1,1,1,1,1], [1,1,1,0,0], [1,1,1,0,0], [1,1,1,0,0], [1,1,1,1,1]]



- The first square of defective products is a sub-matrix *3 x 3* starting at (0,0) and ending at (3,3)
- The second square of defective products is also a sub-matrix *3 x 3* starting at (1,0) and ending at (4,3)
- The third square of defective products is also a sub-matrix *3 x 3* starting at (2,0) and ending at (5,3)
- The size of the largest square of defective products is 3

Function Description

Complete the function findLargestSquareSize in the editor below.

findLargestSquareSize has the following parameter:
  int samples[n][n]: a two-dimensional array of integers

Returns:
  int: an integer that represents the size of the largest square sub-matrix of defective products.

Constraints

- $0 \le n \le 500$
- samples[i][j] is in the set {0, 1} (0 denotes anon-defective products and 1 denotes a defective product)

Input Format For Custom Testing

The first line contains an integer, *n*, the number of *rows (the number of samples)*.
The second line contains the integer, *n*, the number of columns (the number of products in a sample).
Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains *n* space-separated integers that describe *samples[i]*.

Sample Case 0

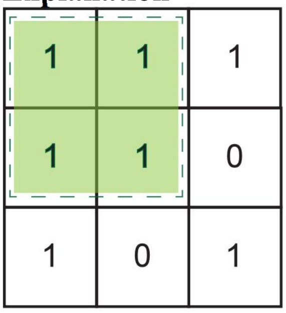
Sample Input For Custom Testing

```
STDIN       Function
-----       --------
3      →  samples[] size n = 3
3      →  samples[i][] size n = 3
1 1 1  →  samples=[[1,1,1],[1,1,0],[1,0,1]]
1 1 0
1 0 1
```

Sample Output

```
2
```

Explanation



- The first square of defective products is a sub-matrix *2 x 2* starting at (0,0) and ending at (1,1)
- The other square of defective products are a sub-matrix *1 x 1* at (2,0) , (0,2) and (2,2)
- The size of the largest square of defective products is 2

Sample Case 1

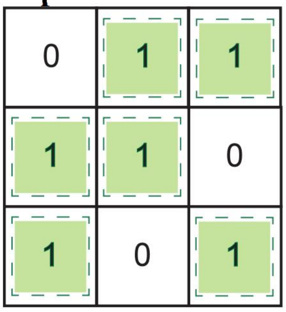
Sample Input For Custom Testing

```
STDIN     Function
-----     --------
3    →   samples[] size n = 3
3    →   samples[i][] size n = 3
0 1 1 →  samples=[[0,1,1],[1,1,0],[1,0,1]]
1 1 0
1 0 1
```

Sample Output

```
1
```

Explanation

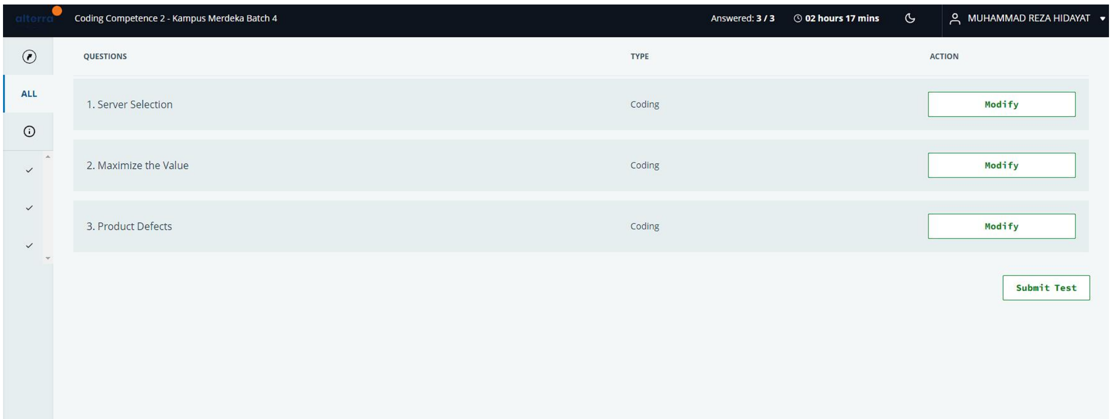


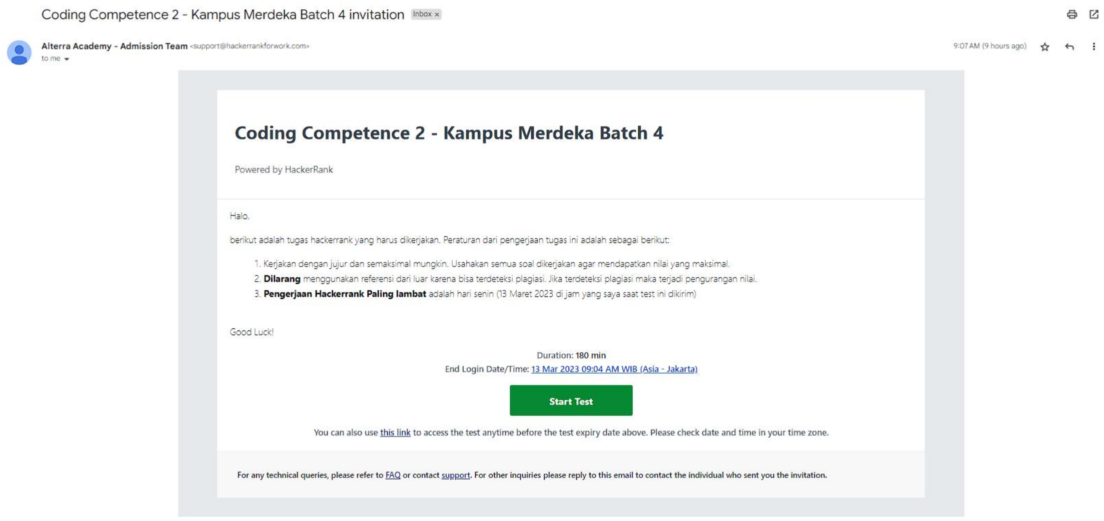
- All square of defective products are a sub-matrix *1 x 1* at (0,1) , (0,2) , (1,0), (1,1), (2,0) and (2,2).
- The size of the largest square of defective products is 1

```go
package main

import (
    "fmt"
)
func findLargestSquareSize(rezasamples [][]int) int {
    n := len(rezasamples)
    maxSize := 0
    // membuat tabel dp untuk menyimpan ukuran sub-
matriks persegi terbesar
    // produk cacat yang berakhir pada setiap posisi
(i,j) dalam matriks sampel
    dp := make([][]int, n)
    for p := range dp {
        dp[p] = make([]int, n)
    }
    // menginisialisasi baris pertama dan kolom pertama
dari tabel dp
    for p := 0; p < n; p++ {
        dp[p][0] = rezasamples[p][0]
        dp[0][p] = rezasamples[0][p]
    }
    // menghitung ukuran sub-matriks persegi terbesar
dari produk cacat
    // berakhir pada setiap posisi (i,j) dalam matriks
sampel
    for p := 1; p < n; p++ {
        for b := 1; b < n; b++ {
            if rezasamples[p][b] == 1 {
                dp[p][b] = 1 + min(dp[p-1][b-1],
min(dp[p-1][b], dp[p][b-1]))
                if dp[p][b] > maxSize {
                    maxSize = dp[p][b]
                }
            }
        }
    }
    return maxSize
```

```go
}
func min(a, b int) int {
    if a < b {
        return a
    }
    return b
}
func main() {
    // mendapatkan masukan
    var n int
    fmt.Scan(&n)
    rezasamples := make([][]int, n)
    for p := range rezasamples {
        rezasamples[p] = make([]int, n)
        for b := range rezasamples[p] {
            fmt.Scan(&rezasamples[p][b])
        }
    }
    //  menemukan ukuran sub-matriks persegi terbesar
dari produk cacat
    maxSize := findLargestSquareSize(rezasamples)
    // mencetak hasilnya
    fmt.Println(maxSize)
}
```





Coding Competence 2 - Kampus Merdeka Batch 4 invitation Inbox

Alterra Academy - Admission Team <support@hackerrankforwork.com>
to me

9:07 AM (9 hours ago)

Coding Competence 2 - Kampus Merdeka Batch 4

Powered by HackerRank

Halo,

berikut adalah tugas hackerrank yang harus dikerjakan. Peraturan dari pengerjaan tugas ini adalah sebagai berikut:

1. Kerjakan dengan jujur dan semaksimal mungkin. Usahakan semua soal dikerjakan agar mendapatkan nilai yang maksimal.
2. **Dilarang** menggunakan referensi dari luar karena bisa terdeteksi plagiasi. Jika terdeteksi plagiasi maka terjadi pengurangan nilai.
3. **Pengerjaan Hackerrank Paling lambat** adalah hari senin (13 Maret 2023 di jam yang saya saat test ini dikirim)

Good Luck!

Duration: **180 min**
End Login Date/Time: 13 Mar 2023 09:04 AM WIB (Asia - Jakarta)

Start Test

You can also use this link to access the test anytime before the test expiry date above. Please check date and time in your time zone.

For any technical queries, please refer to FAQ or contact support. For other inquiries please reply to this email to contact the individual who sent you the invitation.