

Asignatura: Análisis y Diseño de Algoritmo - Laboratorio 'B'

Alumno: Hincho Jove, Angel Eduardo

Semestre: 2021 - B

Este breve informe complementa lo desarrollado en el **Laboratorio 09**. Aquí se encontrarán las capturas de pantalla para los casos de pruebas de los ejercicios mientras que el código y documentación respectiva se encuentran en GitHub.

- **Comentario:** Al momento de entregar el código para los diferentes problemas en las páginas web se obviaron ciertos detalles como el método principal o los comentarios de la documentación. Solo se entregó el método pedido por la página.

1. MaxNonoverlappingSegments - Casos de Prueba

The screenshot shows a code editor with a task description on the left and a Java solution on the right. The task description explains the problem: given two arrays A and B representing segments, find the maximum number of non-overlapping segments. The solution is a Java class named Solution with a method solution that iterates through the segments and counts the maximum number of non-overlapping segments.

Located on a line are N segments, numbered from 0 to $N - 1$, whose positions are given in arrays A and B . For each i ($0 \leq i < N$) the position of segment i is from $A[i]$ to $B[i]$ (inclusive). The segments are sorted by their ends, which means that $B[K] \leq B[K + 1]$ for K such that $0 \leq K < N - 1$.

Two segments i and j , such that $i \neq j$, are *overlapping* if they share at least one common point. In other words, $A[i] \leq A[j] \leq B[i]$ or $A[j] \leq A[i] \leq B[j]$.

We say that the set of segments is *non-overlapping* if it contains no two overlapping segments. The goal is to find the size of a non-overlapping set containing the maximal number of segments.

For example, consider arrays A, B such that:

| $A[i]$ | $B[i]$ |
|------------|-------------|
| $A[0] = 1$ | $B[0] = 5$ |
| $A[1] = 3$ | $B[1] = 6$ |
| $A[2] = 7$ | $B[2] = 8$ |
| $A[3] = 9$ | $B[3] = 9$ |
| $A[4] = 9$ | $B[4] = 10$ |

The segments are shown in the figure below.

```
5 // System.out.println("this is a debug message");
6
7 class Solution {
8     public int solution(int[] a, int[] b) {
9
10        int startPos = -1;
11        int segmentAmount = 0;
12
13        for(int i = 0; i < a.length; i++) {
14            if(a[i] > startPos) {
15                startPos = b[i];
16                segmentAmount++;
17            }
18        }
19
20        return segmentAmount;
21    }
22 }
23
24 }
25
```

Test Output: Your code is syntactically correct and works properly on the example test. Note that the example tests are not part of your score. On submission at least 8 test cases not shown here will assess your solution.

Codility

CodeCheck Report: trainingZM4NGD-WRH

Test Name:

[Check out Codility training tasks](#)

Summary Timeline

Tasks summary

| Task | Time spent | Score |
|-------------------------------------|------------|-------|
| MaxNonoverlappingSegments Java 8 | 2 min | 100% |

Total score

100%

Tasks Details

AVAILABLE LESSONS:

| |
|-------------------|
| Lesson 1 |
| Iterations |
| Lesson 2 |
| Arrays |
| Lesson 3 |
| Time Complexity |
| Lesson 4 |
| Counting Elements |
| Lesson 5 |
| Prefix Sums |
| Lesson 6 |
| Sorting |
| Lesson 7 |

[100%] MaxNonoverlappingSegments

Find a maximal set of non-overlapping segments.

START

Programming language: Java 8 ▾

Located on a line are N segments, numbered from 0 to $N - 1$, whose positions are given in arrays A and B . For each i ($0 \leq i < N$) the position of segment i is from $A[i]$ to $B[i]$ (inclusive). The segments are sorted by their ends, which means that $B[K] \leq B[K + 1]$ for K such that $0 \leq K < N - 1$.

Two segments i and j , such that $i \neq j$, are *overlapping* if they share at least one common point. In other words, $A[i] \leq A[j] \leq B[i]$ or $A[j] \leq A[i] \leq B[j]$.

We say that the set of segments is *non-overlapping* if it contains no two overlapping segments. The goal is to find the size of a non-overlapping set containing the maximal number of segments.

For example, consider arrays A , B such that:

| | |
|------------|-------------|
| $A[0] = 1$ | $B[0] = 5$ |
| $A[1] = 3$ | $B[1] = 6$ |
| $A[2] = 7$ | $B[2] = 8$ |
| $A[3] = 9$ | $B[3] = 9$ |
| $A[4] = 9$ | $B[4] = 10$ |

The segments are shown in the figure below.

2. TieRopes - Casos de Prueba

Task 1

Java 8

Files

task1

solution.java

test-input.txt

1

Task 1

There are N ropes numbered from 0 to $N - 1$, whose lengths are given in an array A , lying on the floor in a line. For each i ($0 \leq i < N$), the length of rope i on the line is $A[i]$.

We say that two ropes i and $i + 1$ are *adjacent*. Two adjacent ropes can be tied together with a knot, and the length of the tied rope is the sum of lengths of both ropes. The resulting new rope can then be tied again.

For a given integer K , the goal is to tie the ropes in such a way that the number of ropes whose length is greater than or equal to K is maximal.

For example, consider $K = 4$ and array A such that:

| |
|------------|
| $A[0] = 1$ |
| $A[1] = 2$ |
| $A[2] = 3$ |
| $A[3] = 4$ |
| $A[4] = 1$ |
| $A[5] = 1$ |
| $A[6] = 3$ |

The ropes are shown in the figure below.

1h 58min

Submit Task

solution.java x

```

5 // System.out.println("this is a debug message");
6
7 class Solution {
8     public int solution(int k, int[] a) {
9
10         int tiedRopes = 0;
11         int currentLength = 0;
12
13         for(int i = 0; i < a.length; i++) {
14             currentLength += a[i];
15             if(currentLength >= k) {
16                 currentLength = 0;
17                 tiedRopes++;
18             }
19         }
20
21         return tiedRopes;
22     }
23 }
24
25

```

Test Output

Run Code

Codility

CodeCheck Report: trainingZQZ29D-RNM

Test Name:

[Check out Codility training tasks](#)

Summary Timeline

Tasks summary

| Task | Time spent | Score |
|--------------------|------------|-------|
| TieRopes Java 8 | 3 min | 100% |

Total score

100%

Tasks Details

AVAILABLE LESSONS:

Lesson 1

Iterations

Lesson 2

Arrays

Lesson 3

Time Complexity

Lesson 4

Counting Elements

Lesson 5

Prefix Sums

Lesson 6

Sorting

Lesson 7

Binary Search

100%

TieRopes

Tie adjacent ropes to achieve the maximum number of ropes of length $\geq K$.

Programming language:

Java 8

There are N ropes numbered from 0 to $N - 1$, whose lengths are given in an array A , lying on the floor in a line. For each i ($0 \leq i < N$), the length of rope i on the line is $A[i]$.

We say that two ropes i and $i + 1$ are *adjacent*. Two adjacent ropes can be tied together with a knot, and the length of the tied rope is the sum of lengths of both ropes. The resulting new rope can then be tied again.

For a given integer K , the goal is to tie the ropes in such a way that the number of ropes whose length is greater than or equal to K is maximal.

For example, consider $K = 4$ and array A such that:

$A[0] = 1$

$A[1] = 2$


$A[2] = 3$

$A[3] = 4$

$A[4] = 1$

$A[5] = 1$

3. Bank Queue - Casos de Prueba



Kattis

PROBLEMS CONTESTS RANKLISTS JOBS (5) HELP

Search Kattis

Submit

ANGEL EDUARDO HI...
Score: 7.7, Rank: 56956

Submit a solution to Bank Queue

```
123 // al cliente actual, siendo 0 la posición de tiempo mínima
124
125
126 if(start != -1) {
127     // Establecemos el tiempo de atención para el cliente y recaudamos
128     // su dinero. Posteriormente incrementamos la cantidad de dinero
129     waitingTime++;
130     select[start] = true;
131     totalCash += currentClient.getCash();
132 }
133
134
135 }
136
137 // Finalmente retornamos el monto total de dinero recaudado antes de cerrar
138 return totalCash;
139
```

Switch to file upload

Language


java

Main class

BankQueue

Cancel

Submit



Kattis

[PROBLEMS](#)
[CONTESTS](#)
[RANKLISTS](#)
[JOBS \(5\)](#)
[HELP](#)

ANGEL EDUARDO HI...
Score: 7.7, Rank: 56956

Submission

| ID | DATE | PROBLEM | STATUS | CPU | LANG |
|--|----------|------------|------------|--------|------|
| TEST CASES | | | | | |
| 8153114 | 22:10:55 | Bank Queue | ✓ Accepted | 0.27 s | java |
| <div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> <div>✓</div> </div> | | | | | |

Submission contains 1 file:

download zip archive

| FILENAME | FILESIZE | SHA-1 SUM | |
|----------------|------------|--|---------------------|
| BankQueue.java | 7496 bytes | 202bcfcdd8342c236822fc2099f884084584acf1 | <div>download</div> |

[Edit](#) and [resubmit](#) this submission.


BankQueue.java

```

1
2 /*
3  * Autor: Híncho Jove, Angel Eduardo
4  *
5  * Descripción del Problema:

```

4. A Vicious Pikeman - Casos de Prueba

 **Kattis**
PROBLEMS CONTESTS RANKLISTS JOBS (5) HELP

Search Kattis

Submit

ANGEL EDUARDO HI...
Score: 7.7, Rank: 56957

Submit a solution to A Vicious Pikeman (Easy)

```
56 # tiempo maximo del concurso que se nos da para los ejercicios
57 ...
58
59 def solveProblems(n, givenTime, timeSolve):
60     # Iniciamos variables auxiliares para el tiempo
61     # que tomaremos y la penalidad que recibiremos
62     totalTime, penalty = 0, 0
63     # Empezamos a iterar sobre los ejercicios y sus
64     # tiempos para ser resueltos (t) en la lista
65     for i, t in enumerate(timeSolve):
66         # Analizamos si el tiempo total mas el tiempo que
67         # tomaremos para resolver el ejercicio es mayor
68         # al tiempo maximo o dado dentro del Concurso
69         if (totalTime + t) > givenTime:
70             # Cuando es mayor entonces llegamos al limite
71             # del tiempo dado para el Concurso y devolvemos
72             # la cantidad de solucionador necesario o penalidad
```


Switch to file upload

Language
Python 3

Cancel Submit

Help

SEE FEED FOR NEW PROBLEMS | POWERED BY KATTIS | SUPPORT KATTIS ON PATREON

 **Kattis**
PROBLEMS CONTESTS RANKLISTS JOBS (5) HELP

Search Kattis

Submit

ANGEL EDUARDO HI...
Score: 7.7, Rank: 56957

Submission

| ID | DATE | PROBLEM | STATUS | CPU | LANG |
|------------------|----------|--------------------------|----------|--------|----------|
| TEST CASES | | | | | |
| 8153263 | 23:25:04 | A Vicious Pikeman (Easy) | Accepted | 0.06 s | Python 3 |
| ✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓✓ | | | | | |

Submission contains 1 file: [download zip archive](#)

| FILENAME | FILESIZE | SHA-1 SUM | |
|-------------------|------------|--|--------------------------|
| ViciousPikeman.py | 4803 bytes | f91e95261142b64aa2e8f7d0c97a5d6ddd2d160b | download |

[Edit and resubmit](#) this submission.


ViciousPikeman.py

```
1 ...
2 ...
3 ...
4 Autor: Híncho Jove, Angel Eduardo
5
6 Descripción del Problema:
```

Help

SEE FEED FOR NEW PROBLEMS | POWERED BY KATTIS | SUPPORT KATTIS ON PATREON

5. Watering Grass - Casos de Prueba

 **Kattis**
PROBLEMS CONTESTS RANKLISTS JOBS (5) HELP

Search Kattis

Submit

ANGEL EDUARDO HI...
Score: 12.9, Rank: 42520

Submit a solution to Watering Grass

```
80 # mayor o igual a 1. Esto debido a que existe el caso que no hayan
81 # aspersores disponibles para regar nuestra franja o banda de cesped
82 if currentLong >= 1:
83     # Devolvemos la cantidad de aspersores utilizados
84     return useAmount
85
86 # Metodo principal del programa - Main
87
88 # En este caso usaremos la libreria Sys para ingresar los
89 # datos enviados por consola. Así mismo tambien usaremos
90 # la funcion 'sqrt' o 'Raiz Cuadrada' de la Libreria Math
91
92 def main():
93     # Empezamos con el contador de lineas a leer en 0
94     linesAmount = 0
95     # Por cada linea enviada al input iteraremos
96     for line in sys.stdin:
```


Switch to file upload

Language
Python 3

Cancel Submit

Help

SEE FEED FOR NEW PROBLEMS | POWERED BY KATTIS | SUPPORT KATTIS ON PATREON



Kattis

PROBLEMSCONTESTSRANKLISTSJOBS (5)HELP

Search Kattis

Submit

ANGEL EDUARDO HI...
Score: 12.9, Rank: 42520

Submission

| ID | DATE | PROBLEM | STATUS | CPU | LANG |
|------------|----------|----------------|------------|--------|----------|
| TEST CASES | | | | | |
| 8153503 | 01:39:13 | Watering Grass | ✓ Accepted | 0.31 s | Python 3 |
| ✓✓ | | | | | |

Submission contains 1 file: [download zip archive](#)

| FILENAME | FILESIZE | SHA-1 SUM | |
|------------------|------------|--|--------------------------|
| WateringGrass.py | 7340 bytes | 845cd0e9f368b25f60cacc10c66e4655492afcd7 | download |

[Edit and resubmit this submission.](#)

WateringGrass.py

```
1
2 ...
3
4 Autor: Hincho Jove, Angel Eduardo
5
6 Recepcion del Problema...
```

Help