

UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA

FACULTAD DE PRODUCCION Y SERVICIOS



ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Curso: Laboratorio de Análisis y Diseño de Algoritmos

Aula 09 - Greedy Algorithms

Presentado por:

Tacca Apaza, Nohelia Estefhania

Docente:

Alex Josue Florez Farfan

Grupo-“B”

Arequipa - Perú

Diciembre 2021

Greedy Algorithms

1. Explican cómo llegan a la solución
2. Escriben el código
3. Lo suben a la página
4. Hacen una captura de pantalla de que pasó los casos de prueba (imagen o pdf)
5. Suben el código y la captura de pantalla

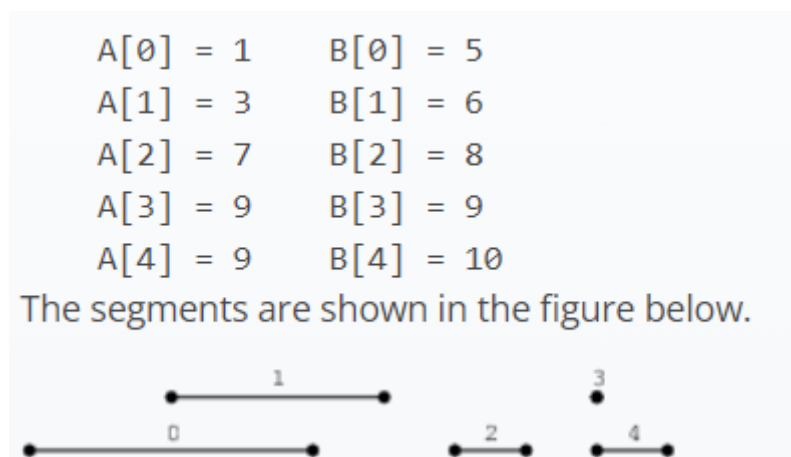
Ejercicio 01 - MaxNonoverlappingSegments

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:



The size of a non-overlapping set containing a maximal number of segments is 3. For example, possible sets are $\{0, 2, 3\}$, $\{0, 2, 4\}$, $\{1, 2, 3\}$ or $\{1, 2, 4\}$. There is no non-overlapping set with four segments.

Write a function:

```
class Solution { public int solution(int[] A, int[] B); }
```

that, given two arrays A and B consisting of N integers, returns the size of a non-overlapping set containing a maximal number of segments.

For example, given arrays A, B shown above, the function should return 3, as explained above.

Write an efficient algorithm for the following assumptions:

N is an integer within the range $[0..30,000]$;

each element of arrays A and B is an integer within the range $[0..1,000,000,000]$;

$A[I] \leq B[I]$, for each I ($0 \leq I < N$);

$B[K] \leq B[K + 1]$, for each K ($0 \leq K < N - 1$).

1. Explican cómo llegan a la solución:

Para explicar el proceso de la solución se utilizó el mismo código y la compilación se muestra en la siguiente página, así mismo el código en github contiene el código con la solución y explicación

Input :
1 3 7 9 9
5 6 8 9 10

count = 1

Final del segmento actual B[0] 5

Inicio del segmento siguiente A[1] 3

--> 3 <= 5

Por lo tanto el segmento que inicia en 3 se cruza con el segmento que termina en 5 entonces, count se mantiene con su valor, count = 1

Inicio del segmento siguiente A[2] 7

--> 7 > 5

Quiere decir que no se cruzarán con el nuevo inicio, por lo tanto count se agregó en uno, entonces count = 2

Final del segmento actual B[2] 8

Inicio del segmento siguiente A[3] 9

--> 9 > 8

Quiere decir que no se cruzarán con el nuevo inicio, por lo tanto count se agregó en uno, entonces count = 3

Final del segmento actual B[3] 9

Inicio del segmento siguiente A[4] 9

--> 9 <= 9

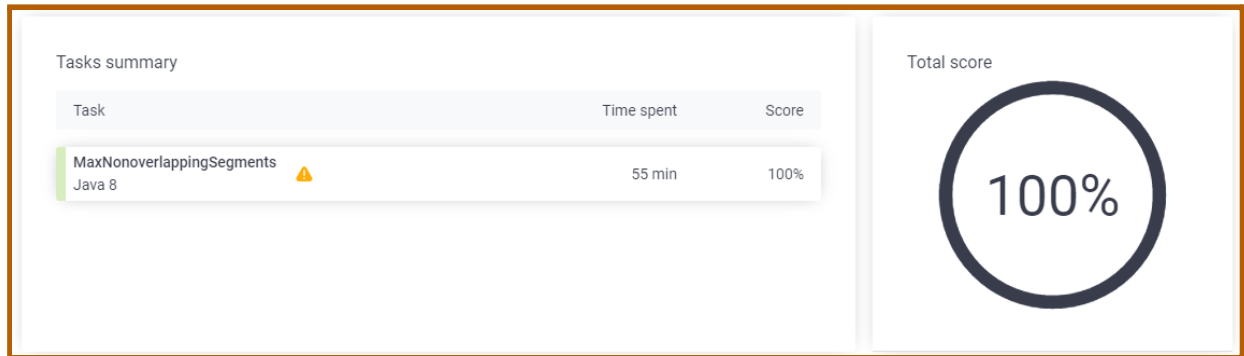
Por lo tanto el segmento que inicia en 9 se cruza con el segmento que termina en 9 entonces, count se mantiene con su valor, count = 3

Output : 3

2. Escriben el código (✓)

3. Lo suben a la página (✓)

4. Hacen una captura de pantalla de que pasó los casos de prueba (imagen o pdf)



5. Suben el código y la captura de pantalla

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a folder 'task1' containing 'solution.java' and 'test-input.txt'. The code editor shows the following Java code:

```
1 // you can also use imports, for example:
2 // import java.util.*;
3
4 // you can write to stdout for debugging purposes, e.g.
5 // System.out.println("this is a debug message");
6
7 class Solution {
8     public int solution(int[] A, int[] B) {
9         if (A.length == 0) {
10             return 0;
11         }
12         if (A.length == 1) {
13             return 1;
14         }
15         int count = 1;
16         int finalOfActualSegment = B[0];
17
18         for (int I = 1; I < A.length; I++) {
19             if (A[I] > finalOfActualSegment) {
20                 count++;
21                 finalOfActualSegment = B[I];
22             }
23         }
24         return count;
25     }
26 }
27
```

Below the code editor, there is a 'Test Output' section with a green checkmark icon. It shows the following text:

Compilation successful.

Example test: ([1, 3, 7, 9, 9], [5, 6, 8, 9, 10])
OK

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.

Ejercicio 02 - TieRopes

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.



We can tie:

rope 1 with rope 2 to produce a rope of length $A[1] + A[2] = 5$;

rope 4 with rope 5 with rope 6 to produce a rope of length $A[4] + A[5] + A[6] = 5$.

After that, there will be three ropes whose lengths are greater than or equal to $K = 4$. It is not possible to produce four such ropes.

Write a function:

```
class Solution { public int solution(int K, int[] A); }
```

that, given an integer K and a non-empty array A of N integers, returns the maximum number of ropes of length greater than or equal to K that can be created.

For example, given $K = 4$ 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 function should return 3, as explained above.

Write an efficient algorithm for the following assumptions:

N is an integer within the range $[1..100,000]$;

K is an integer within the range $[1..1,000,000,000]$;

each element of array A is an integer within the range $[1..1,000,000,000]$.

1. Explican cómo llegan a la solución

```
Input :  
1 2 3 4 1 1 3  
  
*Iniciamos el análisis*  
  
Vamos agregando el largo de las cuerdas en el orden en el que se encuentran  
El largo de la cuerda es 1  
El largo de la cuerda es 3  
El largo de la cuerda es 6  
--> El largo de la cuerda es mayor o igual a 4  
Entonces, agregamos directamente nuestro contador en uno, por lo tanto count = 1  
luego inicializamos nuevamente el valor del largo de la cuerda  
El largo de la cuerda es 4  
--> El largo de la cuerda es mayor o igual a 4  
Entonces, agregamos directamente nuestro contador en uno, por lo tanto count = 2  
luego inicializamos nuevamente el valor del largo de la cuerda  
El largo de la cuerda es 1  
El largo de la cuerda es 2  
El largo de la cuerda es 5  
--> El largo de la cuerda es mayor o igual a 4  
Entonces, agregamos directamente nuestro contador en uno, por lo tanto count = 3  
luego inicializamos nuevamente el valor del largo de la cuerda  
  
Output : 3
```

2. Escriben el código (✓)

3. Lo suben a la página (✓)

4. Hacen una captura de pantalla de que pasó los casos de prueba (imagen o pdf)

Tasks summary

Task	Time spent	Score
TieRopes Java 8	24 min	100%

Total score

100%

5. Suben el código y la captura de pantalla

Files

task1

- solution.java
- test-input.txt

solution.java x

```
1 // you can also use imports, for example:
2 // import java.util.*;
3
4 // you can write to stdout for debugging purposes, e.g.
5 // System.out.println("this is a debug message");
6
7 class Solution {
8     public int solution(int K, int[] A) {
9         int count = 0;
10        int ropeLength = 0;
11
12        for(int i = 0; i < A.length; i++){
13
14            ropeLength += A[i];
15
16            if(ropeLength >= K){
17                count++;
18                ropeLength = 0;
19            }
20        }
21        return count;
22    }
23 }
24
```

To leave editor use Ctrl + M

Test Output ✓

Compilation successful.

Example test: (4, [1, 2, 3, 4, 1, 1, 3])
OK

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.

Ejercicio 03 - Bank Queue

Oliver is a manager of a bank near KTH and wants to close soon. There are many people standing in the queue wanting to put cash into their accounts after they heard that the bank increased the interest rates by 42% (from 0.01% per year to 0.0142% per year).

However, there are too many people and only one counter is open which can serve one person per minute. Greedy as Oliver is, he would like to select some people in the queue, so that the total amount of cash stored by these people is as big as possible and that money then can work for the bank overnight.

There is a problem, though. Some people don't have the time to wait until the bank closes because they have to run somewhere else, so they have to be served before a certain time, after which they just leave. Oliver also turned off the infrared door sensor outside the bank, so that no more people can enter, because it's already too crowded in the hall.

Task

Help Oliver calculate how much cash he can get from the people currently standing in the queue before the bank closes by serving at most one person per minute.

Input

The first line of input contains two integers N ($1 \leq N \leq 10000$) and T ($1 \leq T \leq 47$), the number of people in the queue and the time in minutes until Oliver closes the bank. Then follow N lines, each with 2 integers c_i and t_i , denoting the amount of cash in Swedish crowns person i has and the time in minutes from now after which person i leaves if not served. Note that it takes one minute to serve a person and you must begin serving a person at time t_i at the latest. You can assume that $1 \leq c_i \leq 100000$ and $0 \leq t_i < T$.

Output

Output one line with the maximum amount of money you can get from the people in the queue before the bank closes.

Sample Input 1

```
4 4
1000 1
2000 2
500 2
1200 0
```

Sample Output 1

```
4200
```

Sample Input 2

```
3 4
1000 0
2000 1
500 1
```

Sample Output 2

```
3000
```

1. Explican cómo llegan a la solución

Usar priority queue es más conveniente para este ejercicio.

```
Input :
1000 2000 500 1200
1 2 2 0

Aún hay personas en la cola
Pasamos a la siguiente persona
Si el dinero es mayor al requerido o el tiempo disponible de la persona es mayor al restante del banco
El número de personas que serán atendidas se incrementa en 1, entonces, acceptedPersons = 1
Si la persona fue seleccionada la marcamos select = true

Aún hay personas en la cola
Pasamos a la siguiente persona
Si el dinero es mayor al requerido o el tiempo disponible de la persona es mayor al restante del banco
El número de personas que serán atendidas se incrementa en 1, entonces, acceptedPersons = 2
Si la persona fue seleccionada la marcamos select = true

Aún hay personas en la cola
Pasamos a la siguiente persona
Si el dinero es mayor al requerido o el tiempo disponible de la persona es mayor al restante del banco
El número de personas que serán atendidas se incrementa en 1, entonces, acceptedPersons = 3
Si la persona fue seleccionada la marcamos select = true

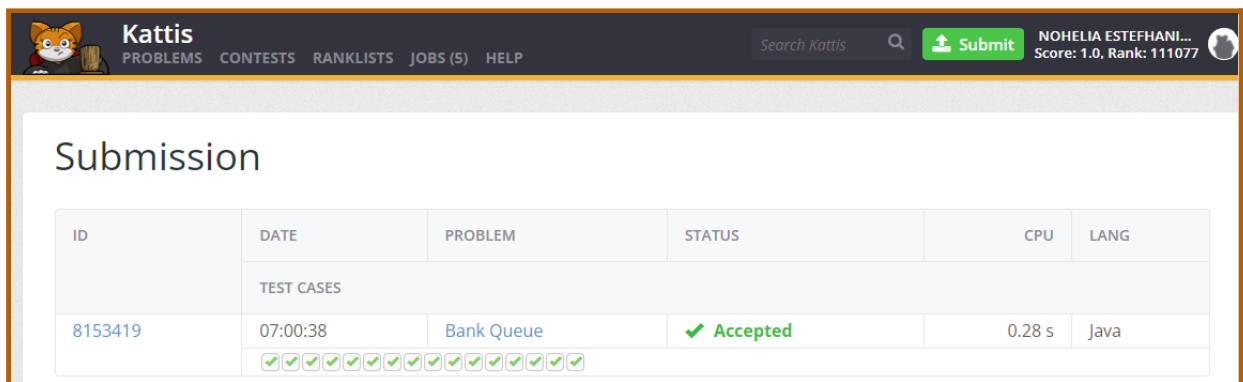
Aún hay personas en la cola
Pasamos a la siguiente persona
Si el dinero es mayor al requerido o el tiempo disponible de la persona es mayor al restante del banco

Output : 4200
```

2. Escriben el código (✓)

3. Lo suben a la página (✓)

4. Hacen una captura de pantalla de que pasó los casos de prueba (imagen o pdf)



The screenshot shows the Kattis submission interface. At the top, there's a navigation bar with 'Kattis' logo, 'PROBLEMS', 'CONTESTS', 'RANKLISTS', 'JOBS (5)', and 'HELP'. A search bar and a 'Submit' button are also present. The user's name 'NOHELIA ESTEFHANI...' and score 'Score: 1.0, Rank: 111077' are displayed. The main content area is titled 'Submission' and contains a table with submission details.

ID	DATE	PROBLEM	STATUS	CPU	LANG
8153419	07:00:38	Bank Queue	✓ Accepted	0.28 s	Java
12 test cases passed (12 green checkmarks)					

5. Suben el código y la captura de pantalla

FILENAME	FILESIZE	SHA-1 SUM
main.java	1298 bytes	299066c8d92347bd2bfc3ef45ec2464ffa00503c

[Edit and resubmit](#) this submission.

main.java

```

1  import java.util.*;
2
3  class main {
4      public static void main(String[] args) {
5          maxResult();
6      }
7
8      public static void maxResult() {
9          Scanner s = new Scanner(System.in);
10         int N = s.nextInt();
11         int T = s.nextInt();
12         int totalCash = 0;
13         int acceptedPersons = 0;
14
15         PriorityQueue<Person> pq = new PriorityQueue<Person>(N, Collections.reverseOrder());
16
17         for (int i = 0; i < N; i++) {
18             pq.add(new Person(s.nextInt(), s.nextInt()));
19         }
20
21         boolean[] select = new boolean[T];
22         Person nextPerson;
23         while (acceptedPersons < N && !pq.isEmpty()) {
24             nextPerson = (Person)pq.poll();
25             int start = nextPerson.time;
26
27             while (start >= 0 && select[start]) {
28                 start--;
29             }
30
31             if (start != -1) {
32                 acceptedPersons++;
33                 select[start] = true;
34                 totalCash += nextPerson.cash;
35             }
36         }
37         System.out.println(totalCash);
38     }
39 }
40
41 class Person implements Comparable<Person> {
42     int cash;
43     int time;
44
45     public Person(int cash, int time) {
46         this.cash = cash;
47         this.time = time;
48     }
49
50     public int compareTo(Person p) {
51         int pcash = p.cash;
52         int ptime = p.time;
53
54         if (cash < pcash) {
55             return -1;
56         }
57         if (cash > pcash) {
58             return 1;
59         }
60         if (time < ptime) {
61             return -1;
62         }
63         if (time > ptime) {
64             return 1;
65         }
66         return 0;
67     }
68 }
69

```