Fundamentos de Algoritmia

Grados en Ingeniería Informática. Grupos E, F

Examen Convocatoria Ordinaria, 18 de enero de 2024.

Nombre: _____ **Grupo:** _____

Laboratorio: _____ **Puesto:** _____ **Usuario de DOMjudge:** _____

Normas de realización del examen

1. Debes programar soluciones para cada uno de los tres ejercicios, probarlas y entregarlas en el juez automático accesible en la dirección `http://exacrc/domjudge/team`.
2. Escribe comentarios que expliquen tu solución, justifiquen por qué se ha hecho así y ayuden a entenderla. Calcula la complejidad de todas las funciones que implementes.
3. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen. El nombre de usuario y contraseña que has estado utilizando durante la evaluación continua **no** son válidos.
4. Escribe tu **nombre y apellidos** en un comentario en la primera línea de cada fichero que subas al juez.
5. Tus soluciones serán evaluadas por el profesor independientemente del veredicto del juez automático. Para ello, el profesor tendrá en cuenta **exclusivamente** el último envío que hayas realizado de cada ejercicio.

1. **(3.5 puntos)** Dado un vector v de $n \geq 1$ enteros y un entero $l \geq 1$, se desea contar el número de segmentos de longitud mayor o igual que l que tienen todos sus elementos iguales. Se puede suponer que $n \geq l$.

   Ten en cuenta que los segmentos se pueden superponer. Por ejemplo, el vector $[8, 8, 8]$ contiene 6 segmentos de longitud mayor o igual que uno con todos sus elementos iguales (3 segmentos de longitud uno, 2 segmentos de longitud dos, y 1 segmento de longitud tres).

   1. (0.25 puntos) Define un predicado *todosIguales*(v, p, q) que se evalúe a cierto si y solo si entre las posiciones p (incluida) y q (excluida) todos los elementos son iguales.
   2. (0.5 puntos) Utilizando el predicado *todosIguales*, especifica una función que dado un entero $l \geq 1$ y un vector v de enteros de longitud $\geq l$, devuelva el número de segmentos de longitud mayor o igual que l que cumplen que todos sus elementos son iguales.
   3. (2 puntos) Diseña e implementa un algoritmo iterativo que resuelva el problema propuesto.
   4. (0.5 puntos) Escribe el invariante del bucle que permite demostrar la corrección del mismo y proporciona una función de cota.
   5. (0.25 puntos) Indica el coste asintótico del algoritmo en el caso peor y justifica adecuadamente tu respuesta.

Entrada

La entrada comienza con una línea que contiene el número de casos de prueba. Cada caso de prueba contendrá el valor de $l \geq 1$ y el número de elementos del vector $n \geq l$ ($n \leq 40.000$), y a continuación los elementos del vector.

Salida

Por cada caso de prueba el programa escribirá una línea con el número de segmentos solicitado en el enunciado.

Entrada de ejemplo

```
5
1 1
-5
2 2
7 7
1 3
8 8 8
3 10
2 2 -1 -1 -1 -1 -1 2 2 2
1 10
0 1 2 3 4 5 6 7 8 9
```

Salida de ejemplo

```
1
1
6
7
10
```

2.(2.5 puntos) Se dice que un vector de $n \geq 2$ números enteros es picudo si la secuencia de diferencias de cada elemento respecto al anterior está formada por una secuencia estrictamente creciente (posiblemente vacía) hasta un valor y a partir de ese valor otra estrictamente decreciente (posiblemente vacía) . Por ejemplo, el vector que contiene los elementos $[1, 4, 10, 20, 29, 35, 40]$ es picudo, ya que la secuencia de diferencias es $[3, 6, 10, 9, 6, 5]$. El vector $[-3, 4, 9, 12]$ también es picudo porque la secuencia de diferencias es $[7, 5, 3]$. Dado un vector picudo, se desea encontrar la mayor diferencia, que en los ejemplos anteriores son respectivamente 10 y 7. Se pide:

1. (2 puntos) Escribe un algoritmo recursivo eficiente que resuelva el problema. No está permitido usar memoria adicional proporcional o superior a n.
2. (0.5 puntos) Escribe la recurrencia que corresponde al coste de la función recursiva e indica a qué orden de complejidad asintótica pertenece dicho coste.

Entrada

La entrada comienza con una línea que contiene el número de casos de prueba. Cada caso de prueba consta de dos líneas, en la primera se indica el número n de elementos del vector ($n \leq 40.000$) y en la segunda los valores del vector picudo separados por blancos.

Salida

Para cada caso de prueba se escribe en una línea la mayor diferencia.

Entrada de ejemplo

```
7
2
7 8
3
1 4 10
3
4 2 1
4
-3 4 9 12
4
7 6 6 5
7
1 4 10 20 29 35 40
8
4 3 3 4 6 10 10 8 3
```


Salida de ejemplo

```
1
6
-1
7
0
10
4
```

3. (4 puntos) Santiago, un alumno de la Facultad de Informática ha de decidir de qué asignaturas matricularse el año que viene, teniendo en cuenta que el número de créditos de las asignaturas en las que se matricule ha de ser mayor o igual que una cierta cantidad C. Algunas de las n asignaturas de la titulación que está cursando tienen un prerrequisito. Estos prerrequisitos vienen dados en un vector r de n componentes, donde $r[i]$ representa la asignatura que es necesario tener previamente aprobada para poder matricularse de la asignatura i (o -1 en caso de no tener prerrequisito). Por supuesto, Santiago sabe para cada asignatura si la tiene aprobada o no, el número de créditos y el coste.

Se pide diseñar e implementar un algoritmo de vuelta atrás que resuelva el problema de decidir en qué asignaturas matricularse (de las aún no aprobadas) cuya suma de créditos supere C, respete los prerrequisitos y la suma de los costes de dichas asignaturas sea mínima. Describe claramente el espacio de soluciones y los marcadores utilizados. Se valorará realizar poda de optimalidad.

Entrada

La entrada comienza con una línea que contiene el número de casos de prueba. Cada caso de prueba contendrá inicialmente el número de asignaturas de la titulación n ($1 \leq n \leq 20$) y el número de créditos mínimos a matricular C ($0 \leq C \leq 60$) ; y a continuación una línea para cada una de las informaciones sobre las asignaturas. Primero una secuencia de valores que indican si la asignatura está aprobada (A) o por aprobar (N). A continuación otra con la secuencia de prerrequisitos donde -1 representa que la asignatura no tiene prerrequisito. Después, el número de créditos por asignatura. Y por último la secuencia de los costes de matriculación.

Salida

Por cada caso de prueba el programa escribirá el menor coste para matricularse cumpliendo las restricciones del problema. En caso de que no sea posible matricularse cumpliendo las restricciones del problema se escribirá NO.

Entrada de ejemplo

```
3

3 10
A N N
-1 0 1
5 5 5
5 10 15

3 10
A N N
-1 0 0
5 5 7
5 10 15

10 10
A N A N N N N N N A
-1 -1 0 0 0 1 2 3 5 5
4 5 4 5 5 6 6 5 5 3
10 30 10 20 10 15 10 20 30 5
```

Salida de ejemplo

```
NO
25
20
```