Fundamentos de Algoritmia

Grados en Ingeniería Informática. Grupos E, F

Examen Convocatoria Ordinaria, 21 de enero de 2022.

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 $n \geq 0$ enteros positivos, se desea contar el número de segmentos no vacíos que cumplen que todos sus elementos son pares.

   1. (0.25 puntos) Define un predicado $todosPares(v, p, q)$ que devuelva cierto si y solo si todos los elementos del vector v contenidos entre las posiciones p (incluida) y q (excluida) son pares.
   2. (0.5 puntos) Utilizando el predicado *todosPares*, especifica una función que dado un vector de enteros positivos de longitud ≥ 0, devuelva el número de segmentos no vacíos que cumplen que todos sus elementos son pares.
   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 del número de elementos n del vector, 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
3
1 3 5
3
6 2 7
0
4
1 2 3 4
4
8 2 6 4
```

Salida de ejemplo

```
0
3
0
2
10
```

2.(2.5 puntos) Un dígito de un número natural n se dice que es *multiplicativo* si es igual al producto, módulo 10, de los dígitos que son más significativos que él. El dígito más significativo es multiplicativo si es 1. Por ejemplo, en el número 23638 hay 2 dígitos multiplicativos, el $6 = (2 * 3) \mod 10$ y el $8 = (2 * 3 * 6 * 3) \mod 10$. Se desea contar cuántos dígitos multiplicativos tiene un número.

Se pide:

1. (1.75 puntos) Escribe un algoritmo recursivo eficiente que permita resolver el problema para un número n dado. No está permitido almacenar en un vector auxiliar los dígitos del número.
2. (0.75 puntos) Escribe la recurrencia que corresponde al coste de la función recursiva utilizando el número de dígitos de n como tamaño del problema. Indica también 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 contendrá el número n.

Salida

Por cada caso de prueba el programa escribirá el número de dígitos multiplicativos de ese caso.

Entrada de ejemplo

```
5
0
1
22
65070
23638
```

Salida de ejemplo

```
0
1
1
2
2
```

3. (4 puntos) Un ratón de laboratorio tiene disponibles en su jaula n teclas para pulsar. Cuando pulsa una tecla por primera vez pone en marcha un mecanismo de recompensas y castigos. Cada vez que pulsa de nuevo una tecla recibe una recompensa o un castigo dependiendo de la última tecla que pulsó y la actual. Esta información viene dada en una matriz T en la que para cada par de teclas i, j $(0 \leq i, j < n)$ $T[i][j]$ indica la recompensa (un valor ≥ 0) o el castigo (un valor < 0) que recibe el ratón al pulsar la tecla j inmediatamente después de la i.

Se pide diseñar e implementar un algoritmo de vuelta atrás que resuelva el problema de encontrar una secuencia de teclas de longitud m que maximice la suma de recompensas obtenidas por el ratón, teniendo en cuenta que la suma de los castigos en valor absoluto no puede ser superior a un determinado valor dado $C \geq 0$.

- (3 puntos) Implementa un algoritmo de vuelta atrás que resuelva el problema. Explica claramente los marcadores que has utilizado y describe el espacio de soluciones.
- (1 puntos) Plantea al menos una función de poda de optimalidad e impleméntala en tu algoritmo.

Entrada

La entrada comienza con una línea que contiene el número de casos de prueba. Cada caso de prueba contendrá inicialmente el valor del número de teclas n, la longitud de la secuencia deseada m y el castigo C que no debe superarse. A continuación n filas con las recompensas/castigos que recibe el ratón según lo explicado anteriormente.

Salida

Por cada caso de prueba el programa escribirá la mayor recompensa que el ratón puede conseguir, o NO en caso de que no sea posible conseguir una secuencia de m pulsaciones con castigo acotado por C.

Entrada de ejemplo

```
3
2 3 1
-1 -1
-1 -1
2 3 2
2 3
-1 -2
3 3 0
2 -1 -5
3 2 1
4 -2 2
```

Salida de ejemplo

```
NO
5
6
```