Fundamentos de Algoritmia

Grados en Ingeniería Informática. Grupos E, F

Examen Convocatoria Ordinaria, 19 de enero de 2023.

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 y un entero $l \geq 1$, se desea contar el número de segmentos de longitud l que cumplen que contienen tantos 0s como 1s. Se puede suponer que $n \geq l$ y que los l primeros números del vector son distintos de 0 y de 1

    1. (0.25 puntos) Define un predicado $cerosUnos(v, p, q)$ que devuelva cierto si y solo si entre las posiciones p (incluida) y q (excluida) hay tantos ceros como unos.
    2. (0.5 puntos) Utilizando el predicado *cerosUnos*, especifica una función que dado un entero $l \geq 1$ y un vector v de enteros positivos de longitud $\geq l$ cuyas primeras l posiciones son distintas de 0 y de 1, devuelva el número de segmentos de longitud l que cumplen que contienen tantos ceros como unos.
    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 $n \leq 20000$ 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 10
-3 2 7 1 3 0 0 8 1 0
2 5
7 -9 6 2 7
1 5
8 1 1 1 1
10 10
2 3 4 5 6 7 8 9 10 11
2 10
-1 -1 1 0 1 0 1 0 1 0
```

Salida de ejemplo

```
4
4
1
1
8
```

2.(2.5 puntos) Mi banco ha establecido un sistema de seguridad por el cual cada vez que quiero realizar una transacción me envía un número $n \geq 0$ a mi móvil y yo he de calcular otro número c a partir de él e introducirlo en la aplicación. Para obtener el código c a partir del número n se realiza la suma de las siguientes cantidades: cada dígito en posición impar se multiplica por 3 y se le suma el menor dígito a su derecha y cada dígito en posición par se multiplica por 2 y se le suma el mayor dígito a su izquierda. Se considera que el dígito más significativo está en la posición 1 y que el número de dígitos de n es par.

Por ejemplo, si el número es 7214, el código que debo introducir es :

$$(7 * 3 + 1) + (2 * 2 + 7) + (1 * 3 + 4) + (4 * 2 + 7) = 55$$

Se pide:

1. (2 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. (0.5 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 $n \leq 2^{31} - 1$.

Salida

Por cada caso de prueba el programa escribirá el número codificado.

Entrada de ejemplo

```
6
7214
85
22
10
1000
1274
```

Salida de ejemplo

```
55
47
14
4
5
50
```

3. (4 puntos) Protección Civil desea organizar m patrullas de personas en una comarca con n ciudades. El número de personas disponibles en cada patrulla viene dado por un vector P. Una patrulla puede ser asignada a lo sumo a una ciudad pero una ciudad puede requerir de varias patrullas para cubrir sus necesidades. Los alcaldes de las ciudades envían dos números: el número de personas que necesitan como mínimo y el mayor número de personas que están dispuestos a admitir en sus instalaciones. Esta información viene dada en dos vectores L y G (se puede asumir $G[i] \geq L[i]$).

El sueldo de los trabajadores dependerá de la ciudad en la que desempeñarán su labor, y dicha información vendrá dada en un vector de sueldos S (por persona). Aunque el número de personas supere las necesidades de una ciudad, las patrullas se mandan completas así que se pagará el sueldo a todas las personas que conforman las patrullas asignadas según la ciudad en la que se encuentren destinadas, mientras que las no asignadas percibirán un sueldo de reserva r por persona.

Se pide diseñar e implementar un algoritmo de vuelta atrás que resuelva el problema de asignar patrullas a ciudades de forma que todas las ciudades tengan sus necesidades cubiertas pero sin pasarse de los máximos admitidos por los alcaldes y se minimice el sueldo a pagar. Describe claramente el espacio de soluciones, los marcadores utilizados y la poda de optimalidad realizada.

Entrada

La entrada comienza con una línea que contiene el número de casos de prueba. Cada caso de prueba contendrá inicialmente el número de ciudades n, el número de patrullas m y el sueldo para las personas en reserva r. A continuación tres líneas correspondientes a las necesidades y los sueldos de las n ciudades L, G, y S, y por último una línea correspondiente a las personas de las m patrullas P.

Salida

Por cada caso de prueba el programa escribirá el número mínimo de personas a asignar para garantizar que las ciudades están cubiertas en las condiciones descritas. Si no es posible realizar dicha asignación se escribirá `NO`.

Entrada de ejemplo

```
2
2 4 3
3 5
5 5
10 2
3 2 4 3
3 4 5
3 5 2
5 5 6
10 2 4
1 6 1 3
```

Salida de ejemplo

```
52
NO
```