Fundamentos de Algoritmia

Grados en Ingeniería Informática. Grupos DG, E, F

Examen Convocatoria Ordinaria, 10 de enero de 2025

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 puntos)** Dado un vector v de $n \geq 1$ enteros positivos (≥ 0) y un entero $l \geq 1$, se desea contar el número de segmentos de longitud l que cumplen que la suma de los valores pares sea estrictamente mayor que la suma de los impares.

    1. (0.25 puntos) Define un predicado *paresMayorImpares*(v, p, q) que se evalúe a cierto si y solo si entre las posiciones p (incluida) y q (excluida) la suma de los valores pares es estrictamente mayor que la de los impares.
    2. (0.5 puntos) Utilizando el predicado *paresMayorImpares*, especifica una función que dado un entero $l \geq 1$ y un vector v de enteros positivos, devuelva el número de segmentos de longitud l que cumplen que la suma de los valores pares es estrictamente mayor que la de los impares.
    3. (1.5 puntos) Diseña e implementa un algoritmo iterativo que resuelva el problema propuesto.
    4. (0.5 puntos) Escribe los invariantes de los bucles que permitan demostrar la corrección de algoritmo y proporciona funciones 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 ($n \leq 100.000$), 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
4 2
2 1
4 4
1 2 3 4
4 6
5 1 2 6 4 13
1 3
2 3 0
3 10
4 7 1 4 2 9 3 10 2 5
```

Salida de ejemplo

```
0
1
2
1
3
```

2.(3 puntos) En Falcon Heights (Minnesota) la empresa FalHeights quiere ser la primera en usar exclusivamente drones para la entrega de paquetes a sus clientes. Para ello, los dueños han comprado una pequeña flota de flamantes drones, cuya principal limitación es la batería que se puede usar. Por ello, planificar los repartos se ha convertido en un desafío.

Los ingenieros de FalHeights conocen de cuántos drones disponen para hacer los repartos, así como de la lista de pesos (en kg) de los paquetes a entregas. Lo que buscan es la capacidad mínima de batería necesaria para hacer todos los repartos teniendo en cuenta lo siguiente:

- Por política de empresa, los repartos deben realizarse en estricto orden de compra (no se pueden reorganizar las entregas).
- El gasto de batería de un dron es proporcional al peso total de los paquetes transportados. En concreto, entregar 1 kg de carga supone necesitar 2 hA (hectoamperios) de batería.
- Sale más barato comprar todas las baterías de los drones de la misma capacidad.

Por ejemplo, si disponemos de dos drones y hay que entregar 3 paquetes de pesos `4 2 3`, la batería mínima será de 10 hA (un dron repartirá 4 kg y otro 5 kg).

Se pide:

1. (2.5 puntos) Diseña un algoritmo recursivo eficiente que permita resolver el problema.
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 consta de varios casos de prueba. Cada uno de ellos ocupa dos líneas. La primera línea indica la cantidad N de paquetes ($1 \leq N \leq 10^8$) y el valor d ($1 \leq d \leq N$) de drones que dispone la empresa. La segunda línea contiene los N pesos (en kg) de los paquetes, todos ellos valores enteros positivos. La entrada termina con 0 0.

Salida

Para cada caso de prueba debe escribirse una línea indicando la capacidad mínima necesaria de las baterías (en hA) para poder realizar todas las entregas con la cantidad de drones disponibles.

Entrada de ejemplo

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

Salida de ejemplo

```
10
8
18
16
```

3. (4 puntos) La Comunidad de FALuca ha sufrido una catástrofe natural que ha afectado a $m \geq 1$ poblaciones. Es necesario distribuir ayuda a dichas poblaciones, cada una de las cuales ha comunicado sus necesidades (kilos de ayuda) al jefe de protección civil de la comunidad. Se dispone de $n \geq 1$ voluntarios para repartir las ayudas y de una tabla que indica para cada voluntario, cuántos kilos de ayuda puede llevar a cada una de las poblaciones (esto depende de muchos factores como la accesibilidad de la población, la localización y los medios de transporte del voluntario etc.) A una población puede enviarse cualquier número de voluntarios pero cada voluntario solo puede ir a una población. Como no se sabe cuando cada voluntario será capaz de llegar hasta la población que se le asigne, se les ha pedido que cada uno lleve los todos los kilos especificados en la tabla. Si luego la cantidad de ayuda que los voluntarios llevan a una población excede sus necesidades solo se repartirá la notificada como necesaria, y en caso contrario se repartirá toda la que han llevado los voluntarios. El objetivo es maximizar la cantidad de kilogramos de ayuda repartida en las poblaciones teniendo en cuenta que el jefe de protección civil quiere que al menos l poblaciones ($l \geq 0$, $l \leq n$ y $l \leq m$) reciban la ayuda que solicitaron.

- Define el *espacio de soluciones* e indica cómo es el *árbol de exploración*.
- Implementa un algoritmo de *vuelta atrás* que resuelva el problema. Explica claramente los *marcadores* que has utilizado.
- Plantea dos posibles funciones de poda de optimalidad, razona sobre cual de ellas es mejor 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 el valor del número de personas n, de poblaciones m y el número de poblaciones mínimo l cuyas necesidades se han de cubrir. A continuación figura la cantidad de kilos de ayuda necesarios en cada población. Finalmente una fila para cada voluntario indicando la cantidad de kilos de ayuda llevará a cada población.

Salida

Por cada caso de prueba el programa escribirá una línea con la cantidad de kilos máxima que se puede repartir. En caso de no sea posible cumplir la restricción de cubrir las necesidades de al menos l poblaciones se escribirá `IMPOSIBLE`.

Entrada de ejemplo

```
4

1 1 1
3
5

4 3 2
10 7 6
2 3 1
4 3 3
2 1 3
4 4 4

4 3 3
10 7 6
2 3 1
4 3 3
2 1 3
4 4 4

4 3 0
10 7 6
2 3 1
4 3 3
2 1 3
4 4 4
```

Salida de ejemplo

```
3
13
IMPOSIBLE
14
```