Fundamentos de Algoritmia

Grados en Ingeniería Informática. Grupos C, F y DG

Examen Convocatoria Ordinaria, 3 de Febrero de 2021.

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.
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 la profesora independientemente del veredicto del juez automático. Para ello, la profesora tendrá en cuenta **exclusivamente** el último envío que hayas realizado de cada ejercicio.

1. **(3.5 puntos)** Dado un vector v de enteros positivos y un natural $k \geq 0$, se desea encontrar la longitud del segmento más largo que no contenga ningún subsegmento con (estrictamente) más de k números pares consecutivos.

   a) (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 y q son pares.

   b) (0.25 puntos) Utilizando *todosPares* define un predicado *noMas*(v, p, q, k) que devuelva cierto si y solo si todos los posibles segmentos del vector v entre las posiciones p y q que tengan todos sus elementos pares tienen una longitud menor o igual que k.

   c) (0.25 puntos) Utilizando el predicado *noMas*, especifica una función que dado un vector de enteros positivos de longitud ≥ 0 y un natural $k \geq 0$, devuelva la longitud del segmento más largo que no contiene ningún subsegmento con más de k números pares consecutivos.

   d) (2 puntos) *Diseña e implementa* un algoritmo iterativo que resuelva el problema propuesto.

   e) (0.5 puntos) Escribe el *invariante* del bucle que permite demostrar la corrección del mismo y proporciona una *función de cota*.

   f) (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 número de elementos, el valor de k y a continuación los elementos de la secuencia.

Salida

Por cada caso de prueba el programa escribirá una línea con la longitud del segmento más largo solicitado en el enunciado.

Entrada de ejemplo

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

Salida de ejemplo

```
6
8
4
0
1
```

2.(3 puntos) Un vector no vacío de enteros positivos es *extraño* si la suma de los números pares de la primera mitad más el producto de los números impares de la primera mitad es menor o igual que el producto de los números pares de la segunda mitad más la suma de los números impares de la segunda mitad, y al menos una de sus dos mitades es extraña. Un vector con un único elemento siempre es extraño.

Se pide:

a) (2 puntos) Escribe un algoritmo recursivo *eficiente* que permita resolver el problema para un vector dado suponiendo que el número de elementos es una potencia de dos.

b) (1 puntos) Escribe la *recurrencia* que corresponde al coste de la función recursiva indicando claramente cuál es el *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 de elementos del vector y a continuación una línea con los elementos.

Salida

Por cada caso de prueba el programa escribirá `NO` si el vector no es extraño según la definición o `SI` si el vector lo es.

Entrada de ejemplo

```
5
2
1 2
2
2 2
4
1 2 2 2
4
2 4 1 2
4
2 2 2 2
```

Salida de ejemplo

```
SI
NO
SI
NO
NO
```

3. (3.5 puntos) La pandemia Covid-19 ha forzado a muchos restaurantes a digitalizarse. El restaurante *Come Sano* dispone de n plazas y conoce las distancias d_{ij} entre cada dos plazas i y j, $0 \leq i, j \leq n-1$. Tiene reservas para $m \leq n$ comensales y una matriz de booleanos c_{kl} le indica si dos comensales k y l, $0 \leq k, l \leq m-1$ son o no allegados. Dos personas allegadas pueden sentarse a cualquier distancia, pero dos no allegados han de estar separados al menos dos metros entre sí. Se quiere decidir cómo sentar a todos los comensales en las plazas del restaurante de forma que se respeten las distancias de seguridad y se *maximice* el número de parejas de comensales allegados donde los dos componentes están sentados a menos de dos metros de distancia.

Nota: se entiende que la pareja formada por el comensal k y el comensal l es la misma pareja que la formada por l y k, así que solo se cuenta una vez. Nadie es allegado de sí mismo. Las distancias entre plazas y la relación de ser allegado son simétricas.

a) (0,25 puntos) Define el *espacio de soluciones* e indica cómo es el *árbol de exploración*.

b) (2.25 puntos) Implementa un algoritmo de *vuelta atrás* que resuelva el problema. Explica claramente los *marcadores* que has utilizado.

c) (1 punto) 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 plazas del restaurante n y de comensales m. A continuación n filas con las distancias d_{ij} entre las plazas del restaurante. Y finalmente m filas que indican mediante 0s (falso) y 1s (cierto) si dos comensales son o no allegados.

Salida

Por cada caso de prueba el programa escribirá `CANCELA` si no se puede sentar a los comensales respetando las distancias de seguridad, y en caso contrario escribirá `PAREJAS` seguida de la cantidad máxima de parejas de comensales allegados sentados a menos de dos metros de distancia.

Entrada de ejemplo

```
3
5 4
0 1 1 4 5
1 0 1 3 4
1 1 0 2 3
4 3 2 0 1
5 4 3 1 0
0 1 1 0
1 0 1 0
1 1 0 0
0 0 0 0
2 2
0 0.5
0.5 0
0 0
0 0
4 4
0 0.5 3.5 4
0.5 0 2.5 3
3.5 2.5 0 3
4 3 3 0
0 1 1 0
1 0 1 0
1 1 0 0
0 0 0 0
```

Salida de ejemplo

```
PAREJAS 3
CANCELA
PAREJAS 1
```