TST - Programación II - Primer Parcial                                                30-06-2016

Ejercicio 1

Sabiendo que el algoritmo de Búsqueda Binaria consiste en buscar un valor en un arreglo ordenado siguiendo el proceso de dividir por la mitad el arreglo dependiendo si el valor buscado es mayor o menor al valor que se encuentra en el centro del arreglo y así sucesivamente hasta dar con el resultado o no.

Escriba un programa que transforme la siguiente versión del algoritmo iterativo en uno recursivo:

|  |
| --- |
| class BusquedaBinaria {    public int buscar(int[] valores, int valor) {     int inicio = 0;     int fin = arreglo.length - 1;     int pos;     while (inicio <= fin) {        pos = (inicio + fin) / 2;        if (valores[pos] == valor)          return pos;        else if (valores[pos] < valor) {          inicio = pos + 1;        } else {          fin = pos-1;        }     }     return -1;    } } |

Aclaraciones

Los valores del arreglo se ingresarán ordenados.

El proceso recursivo debe finalizar cuando se encuentra o no el valor buscado.

Puede modificar la firma del método para incluir más argumentos que crea necesarios para el algoritmo recursivo.

Entrada

La primera línea corresponde a los N números enteros, donde 1 <= N <= 100.

La segunda línea corresponde al X valor entero buscado.

La última línea son los i-ésimo números naturales comprendidos entre 1 <= N <= 100 separados por espacios.

Salida

La posición del valor dentro del arreglo (índice) para el caso de que se lo encuentre o un valor negativo u otra frase indicando que el mismo no existe

Ejemplos

|  |  |
| --- | --- |
| Entrada | Salida |
| 10  2  1 2 3 4 5 6 7 8 9 10 | 1 |
| 10  11  1 2 3 4 5 6 7 8 9 10 | -1 o Valor no existe |

Ejercicio 2:

Referencia: http://codeforces.com/problemset/problem/701/A

Tenemos n cartas (n es par) en un mazo. Cada carta tiene un entero positivo escrito en ella. N / 2 jugadores van a jugar el juego. Al comenzar el juego, cada jugador obtiene dos cartas, las cartas no se repiten, cada carta es entregada a un y solo un jugador.

Encuentra la manera de repartir las cartas, tal que la suma de los valores de las mismas, sea equivalente para cada jugador. Se garantiza que esto siempre será posible.

Entrada

La primer línea de la entrada contiene un entero n (2 ≤ n ≤ 100) - que representa la cantidad de cartas en el mazo. Se garantiza que n es par.

La segunda línea, contiene una secuencia de n enteros positivos a1, a2, ..., an (1 ≤ ai ≤ 100) donde ai equivale al numero escrito (valor) de la i-ésima carta.

Salida

Mostrar n / 2 pares de enteros, el i-ésimo par representa las cartas que deberían repartirse para el iésimo jugador. Cada carta se repartirá para un y solo un jugador. Las cartas se enumeran, respetando el orden en el que aparecieron en la entrada.

Se garantiza que existe una solución. Si hay más de una respuesta correcta, se te permite mostrar cualquiera de ellas.

Ejemplos

|  |  |
| --- | --- |
| Entrada | Salida |
| 6  1 5 7 4 4 3 | 1 3  6 2  4 5 |
| 4  10 10 10 10 | 1 2  3 4 |

Aclaraciones

En el primer ejemplo, las cartas están distribuidas de forma tal, que cada jugador obtiene una suma del valor de sus cartas equivalente a 8.

En el segundo ejemplo, cada ai es equivalente. Por lo tanto, cualquier distribución es aceptable.

Ejercicio 3:

Un supermercado tiene tres cajas para la atención de los clientes. Las cajeras tardan entre 7, 9 y 11 minutos respectivamente para la atención de cada cliente. Los clientes llegan a la zona de cajas cada 2 minutos, si alguna de las cajas está libre, entonces es atendido y en caso contrario, el cliente elige la caja con la cola de menor tamaño para esperar ser atendido. Si todas las cajas tienen 6 personas, el cliente se marcha del supermercado.

Escriba un programa para realizar una simulación durante 8 horas y de manera de obtener la siguiente información:

1.- Cantidad de clientes atendidos por cada caja.

2.- Cantidad total de clientes que se marcharon sin hacer compras.

3.- Tiempo promedio en cola (expresado en minutos), tomando en cuenta las 3 cajas.

Ej: Tiempo total espera en cola / total de personas en cola

Sabiendo que la estructura dinámica interna para Cola es la siguiente, realice los cambios necesarios en la misma para poder resolver el ejercicio de simulación.

|  |
| --- |
| /\*\*  \* Clase Nodo.  \*/  public class Nodo {     public int valor;     public Nodo siguiente;     public Nodo(int valor) {         this.valor = valor;         this.siguiente = null;     }     // toString() retorna el valor del nodo.     @Override     public String toString() {         return this.valor.toString();     }  }  \*\*  \* Clase cola.  \*/  public class Cola {     private Nodo primero;     private Nodo ultimo;     private int tamaño;     public Cola() {         this.primero = null;         this.ultimo = null;         this.tamaño = 0;     }     public boolean vacia() {         return this.primero == null;     }     // enqueue     public void encolar(int valor) {         Nodo nuevo = new Nodo(valor);         if (vacia()) {             this.primero = nuevo;             this.ultimo = nuevo;         } else {             this.ultimo.siguiente = nuevo;             this.ultimo = nuevo;         }         this.tamaño++;     }     // dequeue     public int desencolar() {         if (vacia()) {             return -1;  // cola vacía         } else {             int aux = this.primero.valor;             this.primero = this.primero.siguiente;             this.tamaño--;             return aux;         }     }     public void imprimir() {         if (vacia()) {             System.out.println("Cola vacia");         } else {             Nodo aux = this.primero;             for(int i = 0; i<this.tamaño; i++) {                 System.out.println(aux.valor);                 aux = aux.siguiente;             }         }     }  } |

Aclaraciones

Se considera que cada iteración o ciclo repetitivo dura un 1 minuto.

Solo de debe expresar los tiempos en minutos sin discriminar en horas.

La simulación comienza a las 9 am de la mañana.

Salida

Los valores enteros correspondientes a los clientes atendidos por cada caja.

Un valor entero indicando la cantidad total de clientes que no se atendieron.

Un valor entero indicando la tiempo promedio de espera en cola.