ALGORITMOS: AYUDANTÍA 5

Ayudante: Yerko Ortiz

Objetivo de la ayudantía: Reforzar conceptos referentes a algoritmos de ordenamiento.

Números distintos

A usted le dan una lista de n enteros y su tarea es contar el número de elementos distintos en la lista.

Entrada

- La primera linea contiene un entero n que representa la cantidad de elementos en la secuencia.
- La segunda linea contiene n enteros x_i que representan la lista de números.

Dominio

- $0 \leq n \leq 2 \cdot 10^6$
- $0 \leq x_i \leq 10^9$

Salida

Un entero m que representa la cantidad de números distintos en la secuencia de entrada.

Caso de prueba

- Entrada:
  7
  1 2 1 6 1 2 5
- Salida:
  3

Comensales

Le dan el momento de llegada y el momento de salida de n clientes de un local de comida, datos con los cuales tendrá que determinar el número máximo de clientes que estaban en el local, durante un mismo intervalo de tiempo.

Entrada

- La primera linea contiene un entero n que representa el número de clientes que el local tuvo.
- Las n lineas subsiguientes contienen 2 enteros a, b que representan el momento de llegada de un cliente x_i y el momento de salida del mismo cliente x_i.

Dominio

- $1 \leq n \leq 2 \cdot 10^6$
- $1 \leq a < b \leq 10^9$

Salida

- Un entero m que representa el número máximo de clientes que tuvo el local durante un mismo intervalo de tiempo.

Caso de prueba

- Entrada
  3
  5 8
  2 4
  3 9
- Salida
  2

Compilador mental

Dada cierta entrada, ejecute y compile el siguiente algoritmo usando lápiz, papel e imaginación; describa la complejidad del algoritmo usando la notación **O(f(n))** y responda la pregunta que se presenta.

```
static void countingSort(int[] array, int n)
{
    int i, j, size_table;
    int max = ~0, min = -(~0);
    for(i = 0; i < n; ++i) {
        if(max < array[i]) max = array[i];
        if(min > array[i]) min = array[i];
    }
    size_table = max - min + 1;
    int[] table = new int[size_table];
    for(i = 0; i < n; ++i)
        ++table[array[i] - min];
    for(i = 0, j = 0; i < size_table; ++i)
        if((table[i]--) > 0) array[j++] = (i + min);
}
```

- **Entrada:**
  `int[] arr = {4, 1, 2, 9, 7, -1};`
  `int n = arr.length;`
- **Salida:**
- **Complejidad:**
- **¿Usted conoce algún algoritmo de ordenamiento que pueda considerarse mejor que counting sort? justifique.**

Contar Permutaciones

En este problema, usted deberá analizar un algoritmo de ordenamiento muy particular. El algoritmo procesa una secuencia de n enteros distintos, intercambiando la posición de *dos elementos adyacentes* hasta que la secuencia quede ordenada de forma ascendente.
El calculo a realizar consiste en determinar cuántas operaciones de permutación entre elementos adyacentes es necesario realizar, para que la entrada quede totalmente ordenada.

Entrada

- La primera linea contiene un entero n que representa la cantidad de elementos en la secuencia.
- La segunda linea contiene n enteros x_i distintos que representan la secuencia a ordenar.

Dominio

- $0 \leq n \leq 5 \cdot 10^6$
- $0 \leq x_i \leq 10^9$

Salida

- Un entero m que corresponde al número de permutaciones

Caso de prueba

- Entrada:
  5
  9 1 0 5 4
- Salida:
  6

Gracias por su atención!

"With software there are only two possibilites: either the users control the programme or the programme controls the users. If the programme controls the users, and the developer controls the programme, then the programme is an instrument of unjust power."

Richard Stallman