ALGORITMOS: AYUDANTÍA 2

Ayudante: Yerko Ortiz

Objetivo de la ayudantía: Practicar Java, listas enlazadas y búsqueda binaria.

1. Conceptos

- ¿Qué es un algoritmo recursivo?
- ¿Qué es un tipo de dato abstracto?
- ¿Cuáles algoritmos de ordenamiento conoce?
- Para usar búsqueda binaria en un conjunto de datos, ¿qué se debe saber a priori del conjunto?

2. Diseño e implementación

2.1. Arreglo ascendente

Dado un arreglo de números enteros, se desea modificar el arreglo para que quede ordenado de forma ascendente(modificar no ordenar), es decir que todo elemento en la posición x es menor o igual al elemento en la posición x+1.
En cada turno usted puede aumentar en uno el valor de un elemento, ¿cuántos turnos son necesarios para qué el arreglo quede totalmente ordenado?

Entrada

- Un entero N que es el largo del arreglo.
- N enteros $x_1, x_2, ..., x_N$; los contenidos del arreglo.

Salida

- Imprimir el número de turnos.

Dominio de la entrada

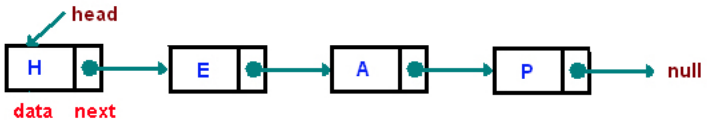
- $1 \leq N \leq 2 \cdot 10^5$
- $1 \leq x_i \leq 10^9$

Caso de prueba:

- **Input:**
  5
  3 2 5 1 7
- **Output:**
  5

2.2. Lista enlazada

Implemente una lista enlazada en Java con los métodos **insertar al principio**, **buscar** y **eliminar al principio**. Para esta lista considere que los nodos almacenan un numero entero y solo hacen referencia al nodo siguiente; el único nodo centinela en la lista ha de ser la cabeza de la misma.



2.3. Búsqueda binaria

Los algoritmos de búsqueda permiten saber si un elemento pertenece o no a un conjunto de datos.
Sea el conjunto A = {1, 2, 3, 4, 5}, ¿6 pertenece al conjunto?, ¿4 pertenece al conjunto?.
Las preguntas de arriba pueden parecer triviales, puesto que el conjunto de datos es pequeño, pero ¿qué tal si el conjunto tuviese diez millones de elementos, podría hacer esta tarea en un tiempo razonable?, a mayor cantidad de datos el tiempo de búsqueda aumenta; existen ciertos cálculos que incluso la mejor maquina contemporánea demoraría eones en calcular.
La estrategia más simple para buscar es búsqueda lineal **O(n)**, empezar desde el primer elemento del conjunto, verificar si este elemento es el que buscamos, en caso positivo terminamos la búsqueda retornando verdadero, en caso negativo continuamos con el siguiente elemento en el conjunto, repitiendo esto de manera sucesiva hasta encontrar el elemento o haber examinado todos los elementos del conjunto.
Búsqueda binaria es una estrategia que realiza la búsqueda en tiempo logarítmico **O(lg(n))**, pero requiere que el conjunto de datos este previamente ordenado de forma ascendente o descendente.
Sea el siguiente arreglo de enteros:
int[] array = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20}
Busquemos el número 20 usando búsqueda binaria: sabemos a priori que el conjunto de datos está ordenado de forma ascendente, es decir que el elemento x_i es menor o igual al elemento x_{i+1}, con esto podemos realizar la búsqueda sin tener que examinar todo el conjunto.
Empezamos en el medio del arreglo, si el arreglo tiene 21 elementos, 21 / 2 = 10, empezamos en el índice 10.
Y preguntamos: ¿array[10] es igual a 20?, array[10] = 10, 10 es menor que 20 por lo que debemos seguir buscando.
Sabíamos de antemano que el arreglo estaba ordenado de forma ascendente, por lo que si sabemos que 10 es menor que 20, no debemos buscar en los elementos menores que 10(elementos a la izquierda de array[10]) por lo que continuamos nuestra búsqueda con los elementos a la derecha de array[10]:
{11, 12, 13, 14, 15, 16, 17, 18, 19, 20}
Ahora seleccionamos el elemento al centro de los elementos a la derecha de array[10], que sería el 16 (en el arreglo original está en array[16]).
Y preguntamos: ¿es array[16] igual a 20?, no 16 sigue siendo menor por lo que toca seguir buscando.
Al ser 16 menor que 20, buscamos a la derecha de array[16]:
{17, 18, 19, 20}
Elemento central es 19(está en array[19] en el arreglo original):
Preguntamos: ¿es array[19] igual a 20?, 19 es menor a 20, por lo que continuamos a la derecha de 19:
Solo nos queda un elemento por examinar:
{20} Preguntamos: ¿es array[20] igual a 20?, si lo es por lo que retornamos verdadero y finalizamos la búsqueda.
Ahora diseñe e implemente búsqueda binaria en Java.

3. Una frase que les podría interesar pero que tal vez no les interese

"Premature optimization is the root of all evil."- Donald Knuth