

# Estructura de datos

## Actividad 7

Dagoberto Quevedo

11 de marzo de 2020

### Resumen

En esta actividad se realiza en primera instancia la diferencia conceptual y abstracta entre arreglos y listas, posteriormente se realiza la implementación computacional de un algoritmo clásico de recuento de inversas con una variante de implementación usando arreglos y en otra listas.

## 1. Arreglos y listas

### 1.1. Arreglos

Un arreglo es una colección ordenada de elementos, donde cada uno de ellos dentro del arreglo tiene un índice. Los índices de los elementos pueden iniciar desde cero  $a[] = [a_0, \dots, a_{n-1}]$  o alternativamente uno  $a[] = [a_1, \dots, a_n]$ . El tiempo de acceso a un elemento con índice  $k$  es  $\mathcal{O}(1)$ , el tiempo para determinar si un elemento  $x$  exista en  $a$  es  $\mathcal{O}(n)$ , se asume que los elementos guardados no están ordenados.

### 1.2. Listas

Las listas son una colección de elementos (también llamados nodos) ordenados de manera secuencial y enlazados con un puntero con el elemento siguiente. Agregar un nuevo elemento o nodo es en tiempo  $\mathcal{O}(1)$ , por lo general las listas se implementan como listas enlazadas (individual, doble, circular, etc.) o como matriz dinámica (redimensionables). Para insertar un nuevo elemento  $v$  después de  $u$ , se ajusta el puntero de  $v$  para que apunte ahora a  $u$ . La operación de eliminación es similar, si se elimina el elemento  $v$  que precede a  $u$ , entonces el puntero del elemento que precede a  $v$ , que puede ser  $q$  apuntará a  $u$ .

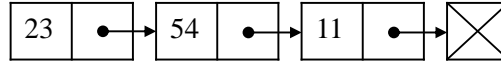


Figura 1: Representación gráfica de una lista enlazada. La primer celda del nodo expresa el dato y la segunda el puntero que enlaza al siguiente nodo.

### 1.3. Diferencias entre ambas estructuras

La principal diferencia entre las dos estructuras de datos es un acceso secuencial o acceso directo. Los arreglos permiten ambos accesos, mientras que las listas solo acceso secuencial. Si bien con las listas se logra flexibilidad y escalabilidad en el tamaño dinámico de las estructuras, las búsquedas y consultas de un elemento tienen un costo de  $\mathcal{O}(n)$  a diferencia de los arreglos que tiene un costo de acceso de  $\mathcal{O}(1)$ .

## 2. Implementación computacional

Se realiza la implementación computacional del algoritmo recuento de inversas, el cual se define como dado un vector  $A$ , encontrar el número de inversas posibles de este, si  $(i < j)$  y  $a_i > a_j$  entonces el par ordenado  $(i, j)$  es llamado como una inversa del vector  $A$ . Se realizan dos implementaciones de este método:

- *Ingenuo*: Es la solución simple debido a que por cada elemento del vector se cuenta todos los elementos menores que su derecha que cumplan la condición  $a_i > a_j$ . La complejidad de esta solución es  $\mathcal{O}(n^2)$ . Se realiza la implementación a través de arreglos.
- *Ordenamiento por mezcla*: Este problema puede resolverse usando un ordenamiento de mezcla. Básicamente para cada elemento del vector, contamos todos los elementos mayores que a su izquierda. Esta implementación permite el uso de listas y recursividad. La complejidad de esta solución es  $\mathcal{O}(n \log n)$ .

## 2.1. Condiciones de experimentación

Se realiza la implementación computacional en Python, la evaluación se realiza ejecutando una serie de instancias, en este caso cada instancia se define como un vector  $A$  de tamaño  $2^n$  con elementos enteros seleccionados aleatoriamente con una distribución uniforme en un rango  $[1, 1 \times 10^8]$ , donde  $n \in \{1, \dots, 15\}$ , la ejecución se repite  $k = 10$  veces.

## 2.2. Resultados

El gráfico 2 muestra los resultados del diseño experimental en el tiempo de computo expresado en escala logarítmica, el eje vertical expresa el valor de  $n$  que define el tamaño de la instancia  $2^n$ .

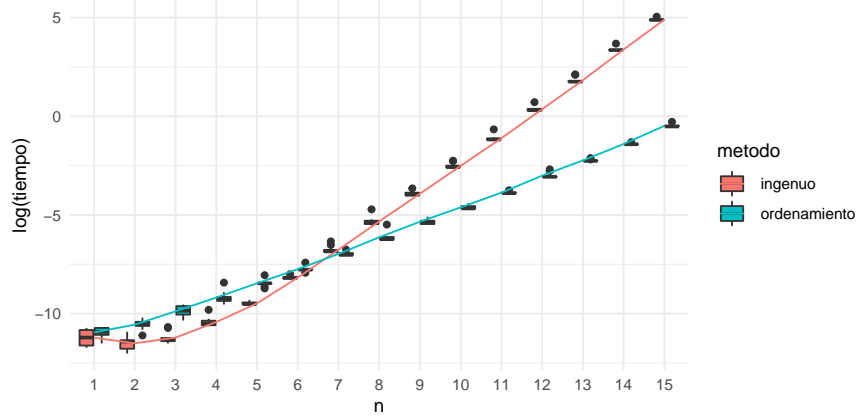


Figura 2: Resultados del diseño experimental con  $k = 10$  ejecuciones para cada tipo de implementación.

## 2.3. Conclusiones

Basado en el resultado gráfico del experimento, el ordenamiento por mezclas es superior en función del tiempo requerido para converger a un resultado, esto se anticipa dado la complejidad de este método es menor a la requerida por el método básico o ingenuo. Si bien este experimento no se logra captar el aporte de cada una de las estructuras, si se puede discernir que la implementación ingenua sólo es posible a través de arreglos y el ordenamiento por mezclas permite mayor flexibilidad en el uso de arreglos o listas.

## Referencias

- [1] Donald Knuth, *Sorting and searching*, The Art of Computer Programming, Addison-Wesley Professional, 1998.
- [2] Elisa Schaeffer, *Modelos computacionales*, Complejidad computacional de problemas y el análisis y diseño de algoritmos, notas de curso, 2020.