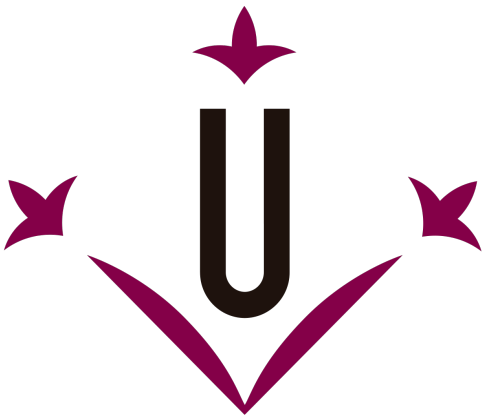

**Universitat de Lleida**

# Algorismes d'Ordenació

Estructura de Dades

Escola Politècnica Superior

Grau en Enginyeria Informàtica

Yhislaine Nataly, Jaya Salazar

Valeria Adriana, Escalera Flores

15.10.2024

# 1 Introducció

Aquest informe presenta els resultats del Laboratori 1, on es van estudiar diversos mètodes d'ordenació: *Selection*, *Bubble*, *Quicksort* i *Insertion*. Es va utilitzar JMH per mesurar els temps d'execució amb diferents quantitats de dades i es van emprar JUnit i Maven per avaluar el funcionament i la complexitat dels algorismes en escenaris reals.

# 2 Bubble Sort

El *Bubble Sort* [1] recorre repetidament un array comparant elements adjacents i els intercanvia si estan desordenats. Després de cada passada, el més gran acaba en la seva posició correcta, formant la part ordenada. El procés es repeteix fins que no hi ha més intercanvis.

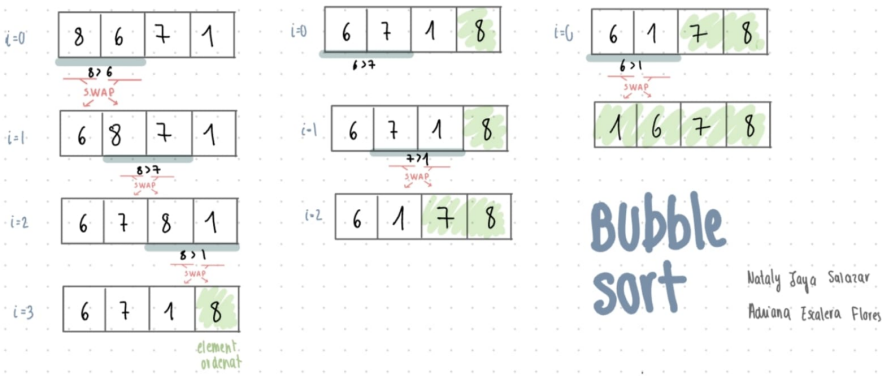


Figura 1: Diagrama del funcionament del Bubble Sort

En termes de *modelling cost* ens trobem amb un codi que es basa en un `do-while` on dins se situa un `for`. És a dir, que el resultat dependrà de les iteracions necessàries. Ens trobem davant d'un $O(n^2)$ en el pitjor cas. En contra, en el millor cas serà un *O(n)*.

# 3 Selection Sort

El *Selection Sort* troba l'element més petit de la part desordenada de l'array i el col·loca a la seva posició correcta, expandint la part ordenada. A cada passada, selecciona el següent element més petit i l'intercanvia amb el primer de la part desordenada. Aquest procés es repeteix fins que l'array està totalment ordenat.

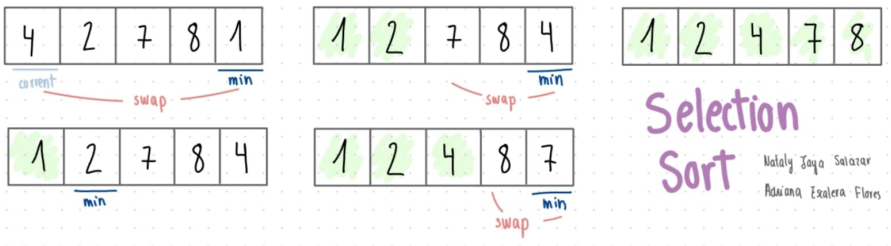


Figura 2: Diagrama del funcionament del Selection Sort

El codi d'aquest algoritme es basa en un bucle `for` on dins conté un altre bucle `for`. Llavors, en aquest cas, tenim un temps total $O(n^2)$ , ja que en tots els casos l'algorisme realitza un nombre fixe de comparacions, independentment de l'ordenació dels elements.

# 4 Insertion Sort

L'*Insertion Sort* [2] construeix la part ordenada de l'array de manera gradual. Pren el primer element de la part desordenada i l'insereix en la posició correcta dins la part ordenada, desplaçant els elements més grans cap a la dreta. Aquest procés es repeteix fins que la llista està ordenada.

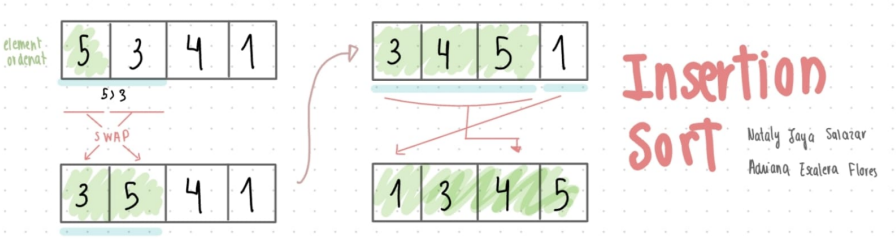


Figura 3: Diagrama del Funcionament del Insertion Sort

L'algorisme d'*Insertion Sort* té una estructura de bucles `for` que genera una complexitat de $O(n^2)$ en el pitjor cas, ja que compara cada element amb els anteriors per col·locar-los en l'ordre adequat. En el millor cas, amb l'array ja ordenat, la complexitat és $O(n)$ perquè no hi ha intercanvis.

# 5 Quick Sort

El *QuickSort* selecciona un pivot i reorganitza l'array en dos subarrays: un amb elements menors o iguals al pivot i un altre amb elements superiors. A continuació, aplica recursivament el mateix procés a cada subarray. Aquest enfocament de *"dividir i conquerir"*

permet ordenar l'array de manera eficient, generalment, en un temps mitjà de $O(n \log n)$. No obstant això, el seu rendiment pot veure's afectat si el pivot no es tria adequadament.

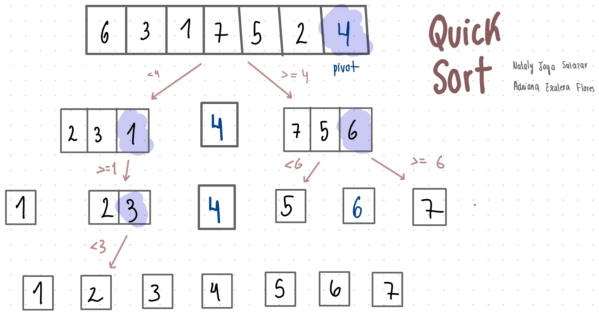


Figura 4: Diagrama del Funcionament del Quick Sort

El millor cas és $O(n \log n)$ gràcies a la selecció del pivot com el valor del mig, que promou una divisió equilibrada. En canvi, el pitjor cas és $O(n^2)$ si l'array ja està ordenat o quasi ordenat, ja que la divisió resulta desigual. Així, es compleixen els comportaments esperats de complexitat temporal.

# 6 Conclusions

Aquesta pràctica ens ha permès aprendre i entendre de manera pràctica els diferents algoritmes d'ordenació. La part que ens ha resultat més difícil ha estat l'anàlisi i la interpretació dels algorismes, ja que hem hagut de canviar la nostra mentalitat i pensar en el pitjor dels casos. Aquest canvi de perspectiva ens ha facilitat la implementació del nostre propi test, *NegativeSortTest*, la qual cosa ens ha donat l'oportunitat d'experimentar i aprofundir en el desenvolupament del nostre codi. En conclusió, tot i que hem afrontat alguns reptes, hem aconseguit assolir l'objectiu de la pràctica i hem après tant la finalitat com el funcionament d'aquests algorismes.

# Bibliografía

[1] GeeksforGeeks. "Bubble Sort Algorithm." (27 de set. de 2024), adr.: `https://www.geeksforgeeks.org/bubble-sort-algorithm/`. (accés: 01 de oct. de 2024).

[2] GeeksforGeeks. "Insertion Sort Algorithm." (6 d'ag. de 2024), adr.: `https://www.geeksforgeeks.org/insertion-sort-algorithm/`. (accés: 01 de oct. de 2024).