Universitat de Lleida
Escola Politècnica Superior
Grau en Enginyeria
Informàtica

Estructura de dades

Laboratori 1
Anàlisi d'algoritmes d'ordenació

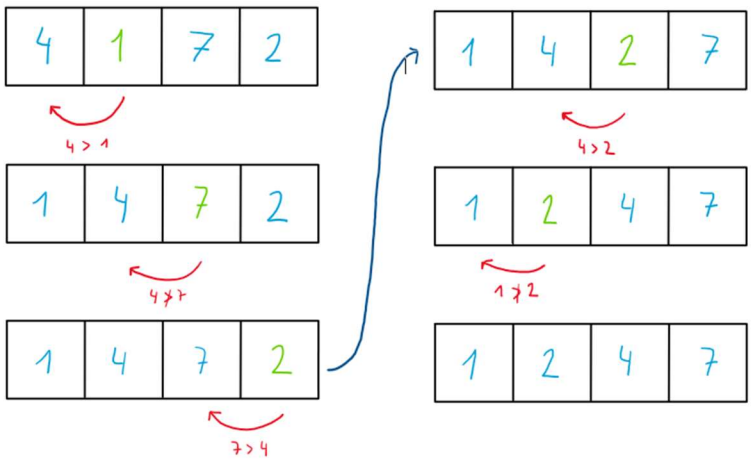
Laura Haro Escoi
Guillem Mora Bea

PraLab2

Dimarts 5 d'octubre de 2021

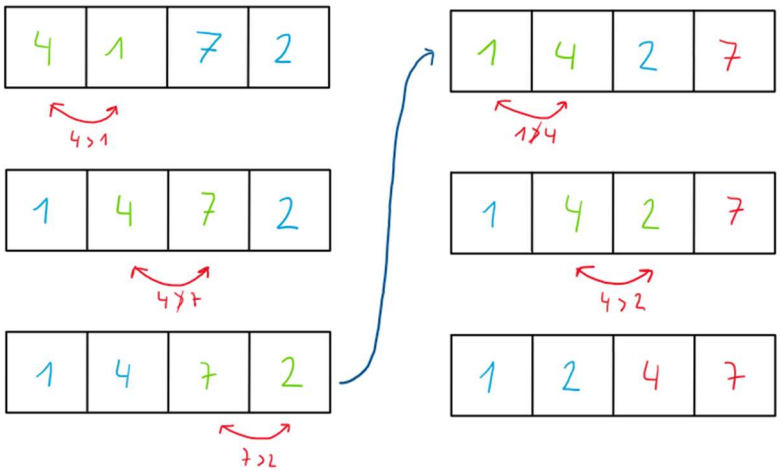
InsertionSort

En aquest algoritme començarem agafant el segon element de l'array i comparant-lo amb el primer element. Si aquest és més gran, es fa un intercanvi, deixant així el més petit dels dos a la primera posició. A continuació, agafarem el següent element de l'array i aplicarem el mateix procediment que a l'anterior, amb la diferencia de que haurà d'anar fent les comparacions amb tots els elements de l'array que tingui davant fins que trobi un de més petit per fer l'intercanvi.



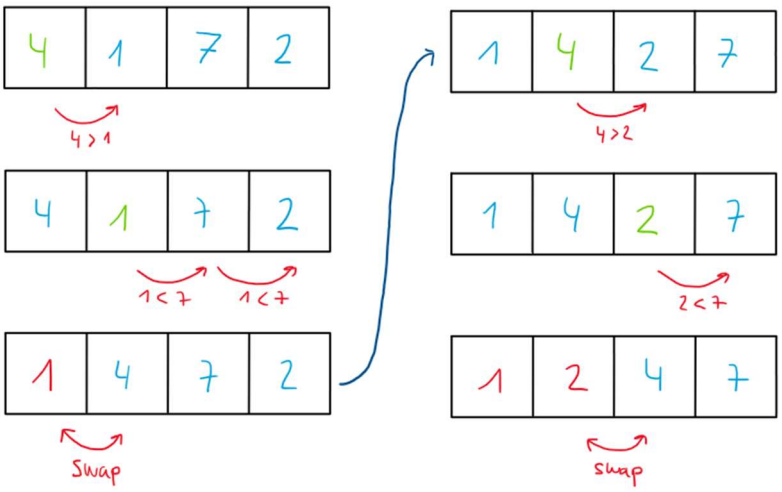
BubbleSort

Aquest algoritme compara el valor de l'element en el que es trobi amb el seu consecutiu. Si aquest és més petit, els hi intercanviarà la posició. Aquest recorregut es repetirà deixant els nombres de la dreta de l'array ordenats fins a arribar al nombre més petit, que estarà situat al primer element de l'array.



SelectionSort

Aquest algoritme consisteix en buscar el valor més de l'array i col·locar-lo a la primera posició d'aquest, i així successivament fins a ordenar tots els valors. Per a fer això, comença triant el primer valor de l'array com a clau i el compara amb els altres elements.

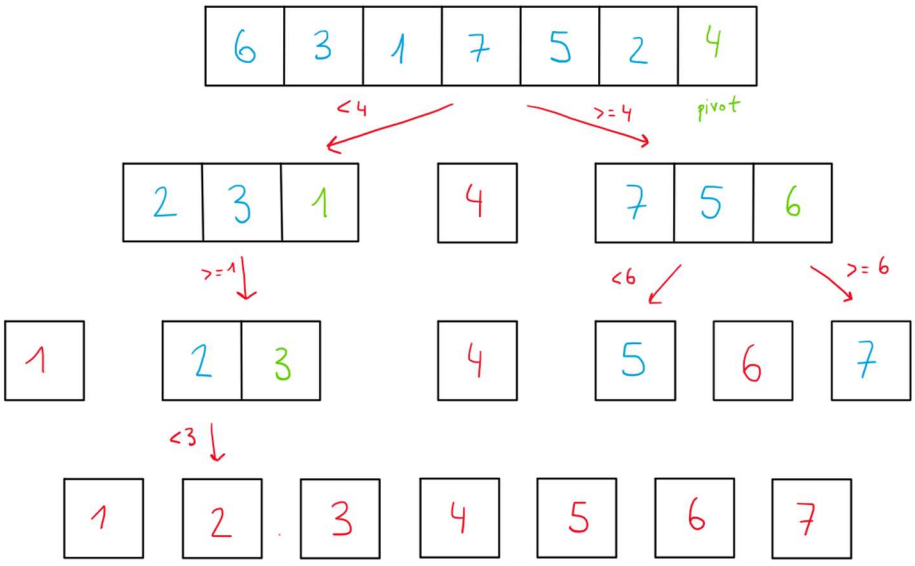


Quan troba un element més petit que la clau, canvia el valor d'aquesta per el valor més petit. Quan arriba al final, canvia de posició el primer element de l'array amb el valor de la clau. A continuació, seguirem fent aquest procés amb els següents valors de l'array fins a tenir tots els nombres ordenats.

QuickSort

Aquest algoritme consisteix en anar dividint l'array mitjançant un valor que anomenarem pivot, on col·locarem els valors més petits que aquest a la seva esquerra i els més grans a la seva dreta. Això no ens assegura que haguem acabat, doncs només sabem que els elements de cada partició son més grans o més petits que un valor.

Per a poder fer-ho, hem de seguir implementant aquest mètode de manera recursiva a les particions o subarrays que ens queden, fins aconseguir que tots els elements quedin ordenats.



Anàlisi de la complexitat

- **InsertionSort**

Aquest algoritme serà de complexitat O(N^2) perquè té dos bucles for, un dins de l'altre, de complexitat O(N) cadascun. Per tant es multipliquen i queda: O(N*N) = O(N^2)

- **BubbleSort**

Aquest algoritme serà de complexitat O(N^2) perquè té dos bucles for, un dins de l'altre, de complexitat O(N) cadascun. Per tant es multipliquen i queda: O(N*N) = O(N^2)

- **SelectionSort**

Aquest algoritme serà de complexitat O(N^2) perquè té dos bucles for, un dins de l'altre, de complexitat O(N) cadascun. Per tant es multipliquen i queda: O(N*N) = O(N^2)

- **QuickSort**

Aquest algoritme serà de complexitat O(N^2) perquè té tres bucles while: dos de consecutius i aquests dins de l'altre. Això vol dir que ens quedaria: O(N*(N+N) = O(N*2N) = O(N^2)

Comentaris

Aquesta pràctica ens ha ajudat a comprendre alguns tipus d'algoritmes d'ordenació i els seus procediments d'execució. Tot hi així, ens hem trobat una sèrie de complicacions a l'hora de resoldre alguns d'aquest de la manera més eficient possible.

El nostre problema principal el vam trobar amb el Quicksort. Primer vam haver d'entendre i analitzar molt bé el seu funcionament, ja que trobàvem que, a diferència dels altres, era molt més complex de programar.

El que ens va costar més va ser plantejar les condicions que feien que l'algoritme acabés, ja que ens quedava un bucle infinit que no sabíem com parar.

Una vegada arreglat això i veien que passava tots els testos, vam provar la seva eficiència amb el Benchmark. Allí vam veure que era terriblement ineficient ja que tardava 100 vegades més en executar-se que tots els altres algoritmes (quan es suposa que és el més ràpid de tots), per tant vam haver de buscar una solució.

Aquí és on vam veure que el nostre problema principal era que les condicions estaven mal plantejades, i aquestes feien que el programa repetís iteracions que no calien. Després de unes modificacions vam poder optimitzar molt més el temps d'execució.