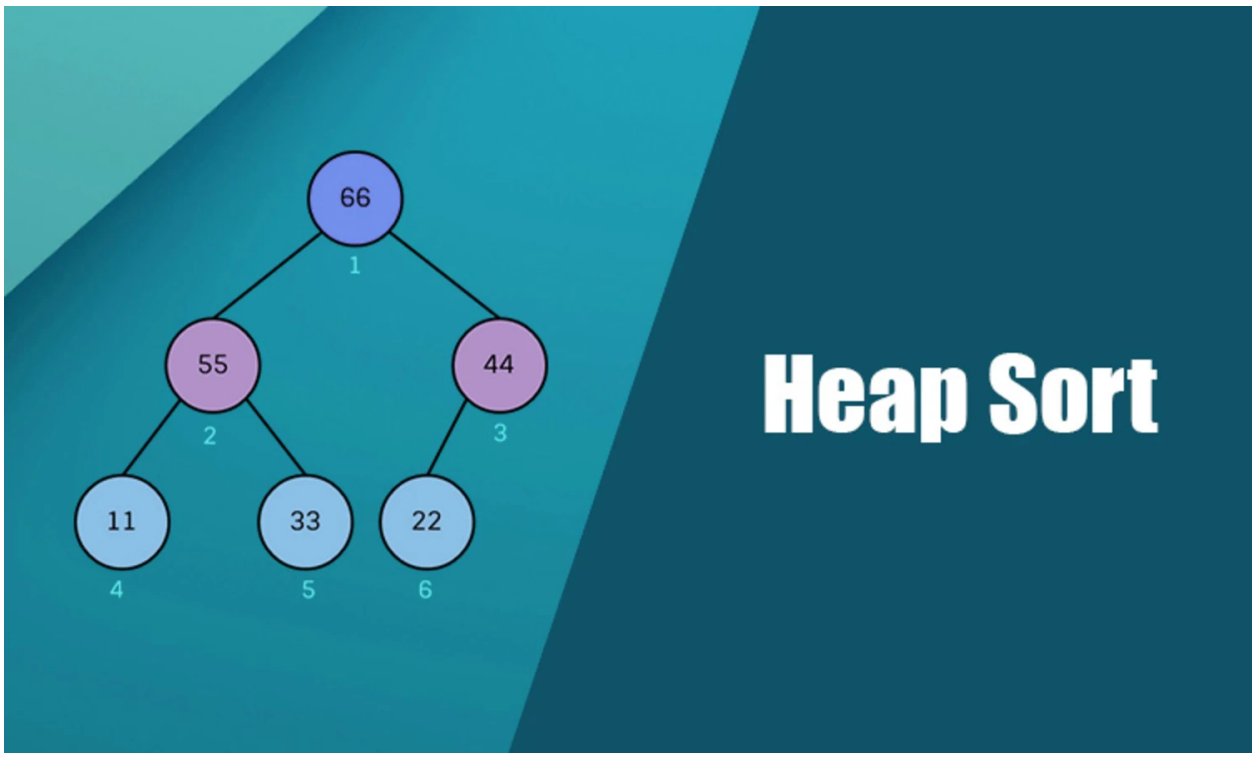
MP05 UF2 OPTIMITZACIÓ DE PROGRAMARI

Projecte UF2 Grup 4: Algorisme Heapsort a Java

Iván Tovar, Sergi Serra, Pol Sánchez, Arnau Vilalta, Yasmine Faddis, Jessica Martínez i Maximiliano Godoy



Índex

Link del GitHub

https://github.com/ssr765/projecte_MP5

Definició detallada del problema.

El problema consisteix a implementar l'algorisme Heapsort en Java per a ordenar un vector o array de nombres desordenat, tenint en compte la prevenció d'excepcions.

L'algorisme es basa en la construcció d'un **Heap màxim** (un arbre binari on l'element de cada node pare és més gran o igual que els seus fills). I llavors en l'**ordenació** mitjançant intercanvis. (S'explica a la solució detallada).

És important **gestionar** les **excepcions** per evitar errors, com verificar que la llista d'entrada no estigui buida o nul·la, controlar els límits de les variables i tenir en compte casos especials com vectors amb un únic element o ja ordenats.

En resum, l'objectiu és obtenir un **algorisme Heapsort robust** i **eficient** per a l'**ordenació** d'**arrays** en **Java**.

Definició detallada de la solució.

S'ha agafat el codi en Python de la pàgina: https://www.programiz.com/dsa/heap-sort i s'ha estudiat per poder comprendre el funcionament d'aquest.

1. Construcció del Heap:

El primer pas és **convertir** la **llista desordenada** en un **Heap**. Això implica **reorganitzar** els **elements** de la llista perquè es **compleixin** les **propietats** del mateix que bàsicament són que **l'element més gran** estigui a **l'arrel** (posició 0 de l'array) i els **elements** més **petits** estiguin a les **branques inferiors**.

Aquest procés es fa utilitzant una sèrie de **comparacions** i **intercanvis** de **posició** dels **elements** dins l'array.

2. Ordenació:

Un cop reorganitzat l'array, el següent pas és extreure l'**element més gran** (l'arrel) i **col·locar-lo** en la seva posició **fina**l a la **llista** ordenada.

Després, es **reconstrueix** el **Heap** amb els **elements restants**, sense tenir en compte l'element final (doncs, ja està col·locat i ordenat).

Aquest procés es **repeteix fins** que **tots** els **elements** hagin estat extrets i **col·locats** a la **posició correcta**.

S'ha **agafat** el **codi base** en **Python** i s'ha realitzat la seva **implementació** en **Java**. Durant aquest procés, s'han realitzat modificacions com **canviar** els noms de les **variables**, **simplificar** les **funcions** i **afegir comentaris** al codi per a una **millor comprensió**. A més, s'ha realitzat la **documentació** del codi amb **Javadoc** i s'han **creat tests** per a **comprovar** que el projecte **funciona correctament**. Aquestes accions han estat realitzades per a **garantir** la **qualitat** i el **correcte funcionament** de la implementació d'**algorisme** d'**Heapsort** en **Java**.

Definició dels casos de prova.

El que hem fet per realitzar les proves del programa és, **crear** un **arxiu** on hi ha especificades **llistes desordenades** per **enviar** al **programa** principal on s'**ordenaran** i ens **donarà** la **sortida** de la **llista ordenada**. En aquest arxiu hem creat **10 tests**.

S'ha **verificat** que la **sortida** sigui **correcta** en cas que les **condicions** de la **llista** passada són **correctes** o que hi hagi **excepcions** si **alguna** de les **condicions no són correctes**.

Webgrafia

Explicació en anglès de com funciona el HeapSort, amb exemples de codi en diferents llenguatges de programació: https://www.programiz.com/dsa/heap-sort

Explicació en castellà amb demostració algorítmica de com funciona el HeapSort: http://algorithmics.lsi.upc.edu/docs/pqueues.pdf