Pràctica 4. Heaps

1. Introducció

***Objectiu**:* Familiaritzar-se amb les estructures de dades de Heap.

Temes de teoria relacionats amb la pràctica: Tema 5 Heap.

2. Enunciat

L'aplicació que es vol implementar és una "base de dades" per guardar les muntanyes del món. Aquesta estructura haurà de mantenir informació sobre les dades de les muntanyes i permetre accedir a aquesta informació de manera ràpida.

Per a cada muntanya disposarà d'un ID únic, el seu nom i la seva alçada (m).

Un exemple de les dades podria ser:

1::Makalu::8481

2::Everest::8848

3::Annapurna::8091

4::Nanga Parbat::8125

5::K2::8621

…

I també caldria oferir la possibilitat de poder consultar el nom i l'alçada d'una muntanya a partir d'un ID.

En aquesta pràctica es demana implementar aquesta funcionalitat utilitzant una estructura de dades de tipus Heap.

Exercici 1. Heap

Implementeu el TAD **`MaxHeap`** corresponent a un Heap amb **una representació en vector**. Com en el cas de la pràctica anterior, heu de tenir en compte que cada node del Heap estarà representat per un `mountainID` (que serà la clau o *key*) i una muntanya (que serà el valor o *value*).

L'especificació amb el **mínim** d'operacions necessàries al TAD **`MaxHeap`** és la següent:

- **`constructor`**: construeix el Heap buit
- **`size`**: retorna el nombre de nodes que hi ha en el Heap
- **`empty`**: retorna cert si el Heap està buit, fals en cas contrari

- **`insert`**: afegeix un nou element al Heap. Aquesta funció rep la clau i el valor/s d'aquesta clau.
- **`max`**: retorna la clau màxima del Heap
- **`maxValues`**: retorna els valors de la clau màxima del Heap
- **`removeMax`**: elimina el node màxim del Heap
- **`printHeap`**: imprimeix per consola tot el Heap
- **`search:`** Busca la clau donada al Heap i retorna el vector de valors associat;

Es poden implementar altres mètodes que siguin necessaris pel desenvolupament de la pràctica, **tot i justificant a la memòria el seu ús** i el seu cost computacional teòric. A la memòria també heu de justificar la representació del TAD MaxHeap i el cost computacional teòric de les funcions del TAD MaxHeap.

Decidiu vosaltres el TAD per representar dels nodes del Heap i justifiqueu la vostra decisió al codi. Es valorarà l'encapsulament de les dades, el cost d'emmagatzemament, i el cost computacional teòric de les operacions associades al TAD definit. Podeu reaprofitar/optimitzar codi de pràctiques anteriors sempre que ho comenteu adientment a la memòria. A més a més, comenteu a la classe MaxHeap si creieu que una implementació del heap amb enllaços hagués sigut més eficient a la implementació realitzada amb vectors.

Llenceu les excepcions oportunes a cada TAD. El codi s'ha de comentar obligatòriament.

Exercici 2. Cercador de muntanyes amb un MaxHeap

A) Especificació del Cercador de muntanyes

Escriviu la capçalera d'una classe anomenada **HeapMountainFinder** que es cridarà des del main i ha de realitzar les següents operacions:.

- `appendMountain(filename)`: Aquest mètode rep el nom d'un fitxer i emmagatzema el seu contingut a una estructura de dades.
- `insertMountain(mountainID, name, height)`: Aquest mètode rep les dades d'una muntanya i fa la inserció a l'estructura de dades corresponent.
- `showMountain(mountainID)`: Aquest mètode rep un `mountainID` i retorna un sol string amb les dades associades a la muntanya.
- `findMountain(mountainID)`: Aquest mètode mostra per pantalla el nom de `mountainID`, junt amb tota la informació de la muntanya.

B) Implementació del Cercador de muntanyes amb un Heap

Implementeu un cercador de muntanyes **`HeapMountainFinder`** que tingui un objecte de tipus **`MaxHeap`**. En aquest objecte **`MaxHeap`** cada node contindrà elements consistents en una `clau` i una dada associada (Mountain). La clau serà el `mountainID`.

C) Programació del main

A l'algorisme principal s'ha de fer:

- Demanar a l'usuari el nom del fitxer que volem utilitzar amb la pregunta: "Quin fitxer vols (P/G)?" (per petit o gran). Al campus virtual trobareu dos fitxers de text que podeu utilitzar per fer les proves: *mountain_list_small.txt* i *mountain_list.txt*. Aquests fitxers s'hauran d'incorporar a la carpeta del projecte i incloure'ls com a recursos en el projecte. Avaluar el temps d'inserció i mostrar-lo al final de tot. En aquest apartat, a més de demanar el nom del fitxer i llegir-lo, cal:
    - Crear un `HeapMountainFinder`.
    - Inicialitzar el `HeapMountainFinder` a partir del contingut del fitxer.
    - Mostrar el temps de creació del `HeapMountainFinder`.

- Mostrar el MaxHeap en amplada. Fer un comptador que demani confirmació cada 40 muntanyes per seguir mostrant l'índex. NOTA: Es demana un llistat complet del contingut del heap en ordre decreixent de `mountainID`, junt amb tots els altres camps que hi han associats.
- Llegir el fitxer *cercaMuntanyes.txt* que us proporcionem i per a cada mountainID contingut en el fitxer, fer una cerca en el heap del `HeapMountainFinder`. Avaluar el temps de cerca de tots els elements de *cercaMuntanyes.txt* i comptar el nombre d'elements que estant a `HeapMountainFinder`. Mostrar les dues dades per pantalla.
- Visualitzar per pantalla la profunditat de l'arbre equivalent al Heap.

Implementeu aquestes funcionalitats en forma d'un menú al main que permeti realitzar les 4 accions descrites anteriorment i l'opció 5 per sortir del menú.

Atenció: En aquest exercici sols es demana fer amb templates els TAD MaxHeap. El TAD Mountain o HeapMountainFinder no s'han de fer amb templates.

Exercici 3. Avaluació d'estructures

Feu una avaluació del rendiment de la implementació anterior (**Heap**):

1. Compteu el temps de generació de l'estructura per les dos llistes de muntanyes de diferent grandària. Féu les dues proves en el mateix ordinador i en condicions similars.

2. Compteu el temps d'accés (cerca) de les muntanyes de *cercaMuntanyes.txt* amb el **heap** inicialitzat en base als dos fitxers donats.

Raoneu els resultats de temps obtinguts.

Indiqueu quin és el cost computacional teòric de les operacions d'inserció i cerca en el heap. Aquesta avaluació es deixarà com a comentari al main de l'exercici 2.

3. Lliurament

A partir de la descripció del problema, es demana:

- Implementar els exercicis en C++ Versió 11 i incloure el codi de cada exercici en un projecte NetBeans. Lliurar el codi C++ corresponent als vostres exercicis en una carpeta anomenada codi, amb una subcarpeta per a cada exercici.

Com a màxim el dia del lliurament es penjarà en el campus virtual un fitxer comprimit **en format ZIP** amb el nom de l'alumne/a, el nom del grup (A, B, C o F), i el numero de la pràctica com a nom de fitxer, **GrupA_LisaSimpson_P4.zip**, on P4 indica que és la "pràctica 4". El fitxer ZIP inclourà: les carpetes amb tot el codi.

Els criteris per acceptar la pràctica són:

- La pràctica ha de funcionar en la seva totalitat.
- La pràctica ha de ser orientada a objectes.
- El codi ha d'estar comentat.

IMPORTANT: La còpia de la implementació de la pràctica implica un zero a la nota de pràctiques de l'assignatura per les persones implicades (tant la que ha copiat com la que ha deixat copiar).

4. Planificació

Lab 8 i Lab 9, Maig de 2019

Per aquesta pràctica els professors proposen la següent planificació:

- **Setmana 1** *(Classe de Laboratori 8)*
  1. Implementació del TAD de l'exercici 1 i comentar el codi
  2. Implementació del programa principal l'exercici 1 i comentar el codi
- **Setmana 2** *(Classe de Laboratori 9)*
  1. Implementació de l'exercici 2 i comentar el codi
  2. Implementació de l'exercici 3 i comentar el codi

Lliurament: Dia 19 de Maig de 2019

IMPORTANT!! A la classe de Laboratori 10 es farà la prova de la pràctica P3 i P4.