PUBLICACIÓ DOCENT

MANUAL DE LABORATORI D'ESIN Sessió 8

AUTOR: Bernardino Casas, Jordi Esteve
ASSIGNATURA: Estructura de la Informació (ESIN)
CURS: Q3
TITULACIONS: Grau en Informàtica
DEPARTAMENT: Ciències de la Computació
ANY: 2018

Vilanova i la Geltrú, 22 de novembre de 2018

8

Exercici

L'objectiu d'aquest exercici és resoldre problemes d'ordenació eficient, o sigui que tinguin un cost mig de $\Theta(n) = n \cdot log(n)$. Si useu algorismes amb cost quadràtic obtindreu errors del tipus "Time limit exceeded".

Caldrà resoldre els següents problemes de la plataforma jutge.org; els trobareu en l'apartat Ordenació eficient del curs ESIN (Vilanova):

- https://jutge.org/problems/P52205_ca. Ordenació per fusió.
- https://jutge.org/problems/P40558_ca. Ordenant amb cues de prioritats.

Encara que el títol de primer exercici sigui "Ordenació per fusió" i el segon "Ordenant amb cues de prioritats", obtindreu un semàfor verd en aquests exercicis si programeu qualsevol algorisme d'ordenació eficient. Per tant us recomano que en el primer exercici envieu dues versions, una usant l'algorisme MergeSort adaptat a vectors i una altra usant l'algorisme QuickSort. I en el segon exercici useu HeapSort tal com es demana.

8.1 Consells

L'algorisme QuickSort vist a teoria treballava directament amb vectors, pel que us serà molt fàcil adaptar-lo per resoldre el primer problema.

L'algorisme MergeSort vist a teoria treballava amb llistes encadenades en memòria dinàmica. Per tant caldrà adaptar-lo per treballar amb vectors: la funció $mergesort()$ principal serà pràcticament igual però haureu d'adaptar les funcions $partir()$ i $fusionar()$ perqué ara haureu de partir un vector en dos i fusionar dos vectors en un.

L'algorisme HeapSort vist a teoria treballava directament amb vectors, pel que no serà gaire complicat adaptar-lo per resoldre el segon problema. Tingueu en compte que en la implementació de teoria s'ha considerat que els elements del vector comencen en la posició 1. Per resoldre l'exercici caldria implementar dues versions de HeapSort, una que ordeni de menor a major i una altra de major a menor. Però podeu usar només

una versió si feu servir el truc d'inserir els enters amb el signe canviat per tal d'obtenir l'ordenació al revés.

8.2 Exercicis opcionals

Podeu resoldre problemes de cues de prioritat, teniu una secció específica en el curs ESIN (Vilanova).

Els problemes d'aquesta secció no poden utilitzar la classe priority_queue de la STL. Cal incloure la definició i implementació pròpia de la classe $CuaPrio$ usant Heaps que hem vist a teoria (en cas que el conjunt de possibles prioritats sigui reduït es més convenient una taula de cues normals). Per tal de superar els jocs de prova privats, cal retocar la implementació per guardar els Heaps en un vector que no estiguin limitats per un nombre MAX d'elements (podem afegir i eliminar elements del final del vector que guarda el heap).

L'especificació i implementació de cues de prioritat amb heaps les trobareu en els apunts de teoria; per evitar problemes copiant des de fitxers PDF les podeu copiar de la carpeta `/home/public/esin/sessio8`).