ESCOLA TÈCNICA SUPERIOR D'ENGINYERIA
Universitat Rovira i Virgili

Estructures de Dades 2019/20

Pràctica 1. Estructures lineals

Exercici 3 / 3. Llista doblement encadenada i Skip List

En aquesta pràctica implementarem i utilitzarem estructures dinàmiques amb memòria dinàmica. Començarem per una estructura senzilla amb operacions bàsiques, la llista doblement encadenada. A continuació, crearem una estructura molt més eficient per tal de buscar elements quan la llista està ordenada, una Skip List.

Especificació i implementació de la llista doblement encadenada

Es demana implementar una llista doblement encadenada que conté internament un punt d'interès (PDI). El tipus es dirà `llista_encadenada` i s'utilitzarà per a emmagatzemar enters. Les operacions a proporcionar són:

```
int Crear(llista_encadenada *ll);
int Destruir(llista_encadenada *ll);
int Principi(llista_encadenada *ll);
int Final(llista_encadenada *ll);
int Avancar(llista_encadenada *ll);
int Retrocedir(llista_encadenada *ll);
int Es_Final(llista_encadenada ll, bool *esfinal);
int Actual(llista_encadenada ll, int *elem);
int Inserir(llista_encadenada *ll, int elem);
int Esborrar(llista_encadenada *ll);
int Longitud(llista_encadenada ll, int *lon);
int Buscar(llista_encadenada ll, int elem, bool *trobat);
int Cost_Buscar(llista_encadenada ll, int elem, int *cost);
```

Les operacions `Principi` i `Final` serveixen per a posicionar el PDI a l'inici i al final de la llista, `Avancar` i `Retrocedir` permeten moure el PDI una posició endavant o endarrere, i `Es_Final` ens informa de si el PDI està al final de la llista. `Actual` serveix per a consultar el contingut de la llista en la posició actual del PDI. L'operació `Inserir` afegirà un element a la posició actual del PDI, la posició del qual no canvia, i `Esborrar` eliminarà l'element que està indexant pel PDI, quedant posicionat sobre la posició del següent element de la llista. `Longitud` permet conèixer el nombre d'elements actualment guardats a la llista. `Buscar` permet saber si un element està dins de la llista, sense modificar el PDI. Finalment, `Cost_Buscar` permet saber el nombre de comparacions realitzades (o elements de la llista visitats) quan es fa la cerca d'un element. Per

exemple, si l'element buscat no està a la llista el cost és igual al nombre d'elements emmagatzemats a la llista, mentre que si es troba en primera posició el cost és la unitat.

El retorn de totes aquestes funcions és un codi d'èxit o error, que podeu definir vosaltres de manera similar als proposats a l'exercici anterior.

Un cop tingueu la vostra llista doblement encadenada funcionant i perfectament testejada, s'ha de fer una gràfica on es mostri el cost mitjà de buscar un element de la llista en funció de la mida de la llista. Per fer aquesta gràfica, es crearan llistes de mides entre 1.000 i 50.000 elements, en passos de 1.000 elements. Els elements de la llista seran enters generats de manera aleatòria entre 1 i 2N, on N és la longitud de la llista. Per exemple, si la llista té longitud 6.000, els números aleatoris s'han de seleccionar en el rang entre 1 i 12.000. Una vegada creada la llista, es realitzaran 1.000 cerques d'enters aleatoris per a cada mida de la llista. Per a cada una d'aquestes cerques es guardarà el nombre d'elements que hem hagut de consultar per saber si el nombre hi era o no a la llista. Una vegada realitzat l'experiment, guardarem els resultats en un arxiu CSV de tres columnes, indicant: mida de la llista, cost mitjà i la seva desviació estàndard; les mitjanes aritmètiques i les desviacions estàndard es fan sobre els costos de les 1.000 cerques (de cada mida de llista). Aquest arxiu és el que utilitzareu per a fer la gràfica, utilitzant el programa que vulgueu (gnuplot, grace, matplotlib, ggplot, full de càlcul, etc.).

Especificació i implementació de l'estructura skip list

La skip list es una estructura de dades similar a una llista encadenada en la qual les operacions de cerca funcionen de manera molt més eficients. La principal diferència és que en la skip list els elements es guarden de forma ordenada, mentre en les llistes encadenades els elements poden estar disposats sense cap mena d'ordre. Aquesta estructura no ha estat explicada a classe. L'objectiu d'aquest exercici és que busqueu com funcionen aquest tipus de llistes i implementeu les funcions bàsiques. Podeu començar per llegir informació de la seva descripció a:

- https://brilliant.org/wiki/skip-lists/
- www.cs.cmu.edu/~ckingsf/bioinfo-lectures/skiplists.pdf
- https://www.youtube.com/watch?v=Q9MdwzewSZg

Els dos primers enllaços expliquen la idea, les operacions bàsiques i els avantatges i inconvenients de les skip list, mentre el tercer és útil per a entendre la seva implementació. El tipus es dirà `skip_list`, i es demana implementar la següent especificació:

```
int Crear(skip_list *sl);
int Destruir(skip_list *sl);
int Inserir(skip_list *sl, int elem);
int Esborrar (skip_list *sl, int elem);
int Longitud(skip_list sl, int *lon);
int Buscar(skip_list sl, int elem, bool *trobat);
int Cost_Buscar(skip_list sl, int elem, int *cost);
```

L'operació `Inserir` permet introduir un element a la skip list, que quedarà col·locat en la seva posició corresponent. L'ordre dels enters emmagatzemats ha de ser creixent. Observeu que un mateix element pot estar repetit més d'una vegada dins de la skip list, i això no suposa cap condició d'error. La funció `Esborrar` elimina de la skip list totes les aparicions de l'element

proporcionat. `Longitud` permet conèixer el nombre d'elements actualment guardats a la llista. `Buscar` i `Cost_Buscar` es comporten igual que amb la llista encadenada. El retorn de totes aquestes funcions és un codi d'èxit o error, que podeu definir vosaltres.

Mentre que per a les llistes doblement encadenades s'utilitzen registres que contenen un element (en el nostre cas, un enter) i dos apuntadors (un cap al següent registre, i un altre cap a l'anterior), en les skip list els registres han de tenir quatre apuntadors en lloc de dos. Concretament, calen dos apuntadors per a moure's endavant i retrocedir dins del mateix nivell, un altre per a pujar de nivell, i un altre per a baixar de nivell (això es mostra bé en el tercer enllaç explicatiu).

Un cop tingueu la vostra skip list funcionant i perfectament testejada, haureu d'avaluar el seu rendiment en operacions de cerca de la mateixa forma que amb les llistes doblement encadenades: generant llistes aleatòries de diferents mides, fent cerques sobre elles, creant un arxiu CSV de resultats, i construint la gràfica de cost. De fet, hauríeu de fer una única gràfica amb les dues corbes de cost, la de les llistes doblement encadenades i la de les skip lists.

Lliurament

- Les pràctiques són individuals.
- Es fa el lliurament d'un únic arxiu `PR1.3-NOM_COGNOMS.zip` que conté:
    - Els arxius de codi font
    - Els arxius amb els resultats del càlcul del cost mitjà de les cerques
    - Informe (en pdf) incloent explicacions dels aspectes més rellevants de les vostres implementacions, gràfic del cost mitjà de les cerques, i anàlisi dels resultats