Sistemes Operatius II

Pràctica 2

Xavi Cano
Grup A

Estructura dels fitxers:

- **Makefile:** S'encarrega de generar l'executable, s'han afegit per compilar els fitxers red-black-tree.c i linked-list.c.
- **file.csv:** Fitxer que conte les dades, 10000 primeres linees.
- **208.csv:** Fitxer que conte tota la base de dades, concretament 7009728 linees, no l'he adjuntat perque sino pesaría molt el rar.
- **main.c:** Programa principal, s'encarrega fer tot el proces del programa, en el seguent apartat es comenta mes al detall les seves funcions.
- **red-black-tree.c:** Conté les funcions de l'arbre, s'han modificat les funcions freeRBData, compLT, compEQ, insertNode i deleteTree.
- **red-black-tree.h:** Conté l'estructura de l'arbre, s'ha modificat el TYPE_RBTREE_KEY de int a char *.
- **linked-list.c:** Conté les funcions de la llista enllaçada, s'han modificat les funcions freeListData, dumpListData i compEQ.
- **linked-list.h:** Conté l'estructura de la llista enllaçada, s'ha modificat el TYPE_RBTREE_KEY de int a char *, i s'han afegit *day, *delay i TYPE_RBTREE_KEY key_sec.

Funcionalitat del programa principal (Main.c):

El programa principal esta divitit en varies funcions que s'encarreguen de realitzar tot el funcionament d'aquesta pràctica, les comentem una mica per sobre, apart també es calcula el temps que triga en executar-se el programa.

- **readCSV():** Aquesta funció rep com a parametre el fitxer passat pel terminal i s'encarrega d'anar llegint el fitxer linea per linea , extreure els dies,retards i codis origen/desti dels aeroports i els va guardant en un altre fitxer "dades_node.csv" aquesta informació que mes endevant l'aprofitarem. També ens va be per saber que realment estem extraient be la informació que volem.

- **read_N_blocks():** Aquesta funció s'encarrega de llegir el fitxer generat per la funció anterior i fa el mateix pero llegeix fins que no superem el valor de N que previament hem asignat i genera un nou fitxer anomenat "blocs_dades_node" que conté les N primeres linees que volem llegir. Aquest valor s'ha de canviar manualment en el define, es podría pasar com a segon argument al executar el programa pero com no ho he vist especificat a l'enunciat de la practica ho he deixat així.

- **insert_data_table_hash_tree():** Aquesta funció s'encarrega de llegir el fitxer generat per la funció anterior, guarda les dades en una taula hash i les insereix a l'arbre. Hi han diversos printf's comentats que no els he borrat per comprobar que relament es mostren be les dades, aquets estan concretament a les linees:

  Linea 294: Es mostra el codia IATA dels aeroports d'origen.
  Linea 304: Es mostra el valor hash de cada element del vector de llistes.
  Linea 319: Es mostren les dades de la llista ListData.

Funcionalitat del les funcions auxiliars:

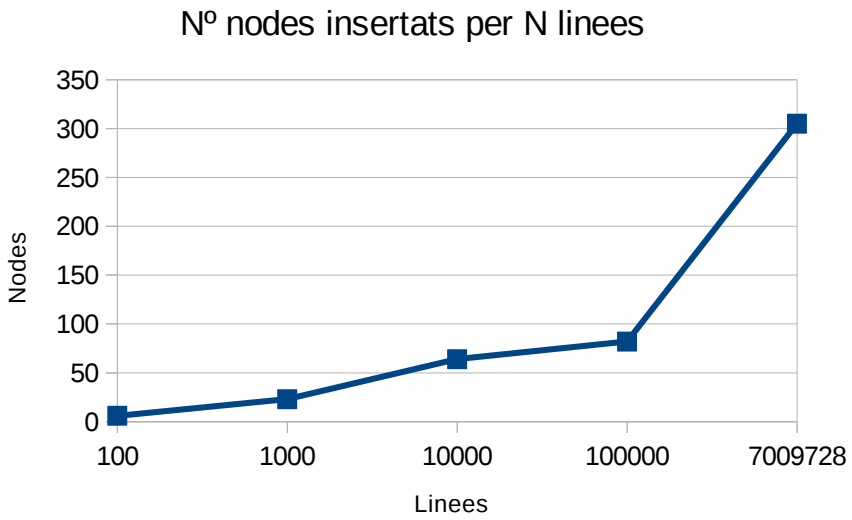
El programa princial (Main.c) per poder realitzar les tres funcions de l'apartat anterior implementa d'altres que les comentarem una mica, que son:

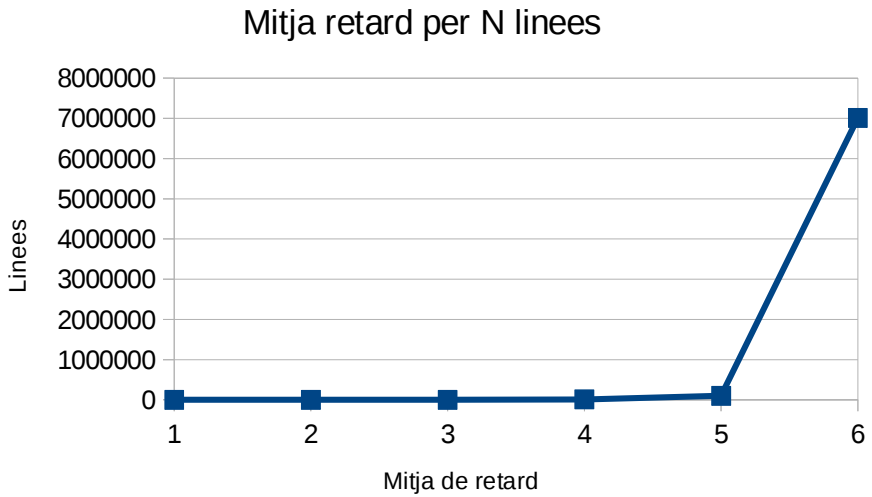
- **getHashValue():** Aquesta funció rep com a parametre el codi IATA origen i amb aquest genera un valor de Hash que ens servira d'index.
- **allocHashTable():** Aquesta funció crea asigna memòria per a la taula hash. La taula hash es compon d'un vector de llistes enllaçades.
- **deleteHashTable():** Aquesta funció simplement s'encarrega d'eliminar la taula hash.

Banc de proves:

Els experiments que he fet en aquesta practia han esta utiliztant el fitxer de 100.000 linees(file.csv) i el de 7.000.000 linees (2008.csv). A continuació es mostra una taula amb els resultats obtinguts de difernts parametres:

Fitxer	Nº de linees (N)	Nodes insertats	Mitja de retard	Temps(amb l'opció valgrind)	Temps(sense l'opció valgrind)
file.csv	10	2	6 minuts	0.488159 segons	0.017643 segons
file.csv	100	6	14 minuts	0.496097 segons	0.019612 segons
file.csv	1000	23	25 minuts	0.556916 segons	0.020377 segons
file.csv	10000	64	16 minuts	1.001799 segons	0.037954 segons
2008.csv	100000	82	5 minuts	4.92310945 minuts	9.607852 segons
2008.csv	7009728	305	7 minuts	11.36159071667 minuts	22.215263 segons





```
xavi@xavi-GA-MA790XT-UD4P:~/Dropbox/XAVI/Universidad/Cuatrimestre1/SO2/2016/Practica2/src$ valgrind ./main file.csv
==7037== Memcheck, a memory error detector
==7037== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==7037== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright info
==7037== Command: ./main file.csv
==7037==
Aeroport d'origen: IAD
Valor_Hash[0]: 36
Valor_Hash[1]: 36
Aeroport d'origen: IND
Valor_Hash[2]: 39
Valor_Hash[3]: 39
Valor_Hash[4]: 39
Valor_Hash[5]: 39
Valor_Hash[6]: 39
Valor_Hash[7]: 39
Valor_Hash[8]: 39
Valor_Hash[9]: 39


Numero d'aeroports d'origen insertats: 2
Mitja de retard dels aeroports: 6 minuts
Temps d'execució: 0.489300 segons
==7037==
==7037== HEAP SUMMARY:
==7037==     in use at exit: 404 bytes in 80 blocks
==7037==   total heap usage: 137 allocs, 57 frees, 14,025,392 bytes allocated
==7037==
==7037== LEAK SUMMARY:
==7037==    definitely lost: 404 bytes in 80 blocks
==7037==    indirectly lost: 0 bytes in 0 blocks
==7037==      possibly lost: 0 bytes in 0 blocks
==7037==    still reachable: 0 bytes in 0 blocks
==7037==         suppressed: 0 bytes in 0 blocks
==7037== Rerun with --leak-check=full to see details of leaked memory
==7037==
==7037== For counts of detected and suppressed errors, rerun with: -v
==7037== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```c
/**
 *
 * Main file
 *
 * This file calls the red-black-tree.c and linked-lit.c functions.
 *
 * Xavi Cano, September 2016.
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <sys/stat.h>
#include <time.h>

#define MAXCHAR  100
#define MAXLINES  7009728
#define N 10
#define HASH_SIZE  100

#include "red-black-tree.h"
#include "linked-list.h"
```

Sortida del programa per N = 10 , amb els diferents printf's descomentats per veure els valors dels diferents parametres.