Pràctica 2

SISTEMES OPERATIUS II

DAVID DE LA PAZ / ORIOL RABASSEDA

ÍNDEX

Observacions del codi

En aquest apartat explicarem detalls sobre la implementació del codi no mencionats a la resta d'apartats respecte a algunes decisions de disseny:

- Per tal de mantenit tots els fitxers en les seves respectives carpetes, hem hagut de modificar el ***makefile***, doncs d'altra manera hauríem d'haver ficat tots els arxius al mateix lloc.
- A l'igual que a la pràctica anterior, el nombre màxim de carácters que poden haver en una fila és fix (tenim *MAX_LINE_SIZE* = 100). No obstant, la primera línia del fitxer de dades és molt més llarga, per la qual cosa necessitem una altra variable per acomodar tal quantitat de caràcteres (agafem *FLIGHT_LINE_SIZE = 500*).
- Pel que fa al fitxer d'aeroports sabem que totes les dades tenen 3 caràcters, per la qual cosa fixem una constant *IATA_CODE* = 3 per canviar l'últim caràcter per '\0'.
- Al final del codi alliberem tota la memòria amb les funcions ***free*** i ***delete_tree***, la qual elimina l'arbre de manera recursiva. També fem free a la funció *flight_list* dels 3 mallocs fets per obtenir les dades del fitxer de vols.
- Tot i que al principi utilitzàvem la funció *strcpy* per copiar la informació de dades a l'arbre, a vegades donava errors al intentar copiar més, per la qual cosa hem optat per utilitzar la funció ***memcpy***, ja que coneixèm la mida de les dades a copiar. A l'hora d'agafar alguna dada d'un fitxer, rescrivim l'últim caràcter amb '\0'.
- Hem hagut de modificar els fitxers ***red-black-tree*** i ***linked-list*** per tal de completar la pràctica. Els canvis principals han consistit en guardar una llista de vols a un node i especificar els atributs de la llista.

Funcionament del programa

El primer pas de la pràctica consisteix en llegir la llista d'aeroports per tal d'omplir un arbre. Començem per inicialitzar l'arbre, el qual llegeix el fitxer *'aeroports'* de forma similar a com ho hem fet a la pràctica anterior. Les diverses funcions de l'arbre venen donades als fitxers ***red-black-tree.h i .c***. La primera línia ens indica el nombre de files a llegir(atoi). Tot seguit guardem memòria per l'arbre i anem llegint les files del fitxer amb la funció *fgets*. En aquest cas, com tots els elements de la llista tenen el mateix tamany, ho utilitzem per reescriure l'últim caràcter amb '\0'. Al llegir una fila, mirem si aquesta està a l'arbre. De no ser així, fem malloc per guardar espai per a un node i hi guardem el nom de l'aeroport així com una ***linked-list*** per guardar els vols. Un cop creat el node, l'afegim a l'arbre i continuem fins haver recorregut tota la llista d'aeroports.

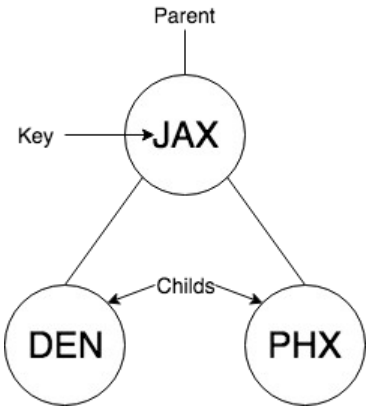

Fig. 1: Esquema d'un node buit

Un cop inicialitzat l'arbre, ara hem d'obrir el fitxer de vols i assignar a cada node la informació pertinent al nombre de vols, així com el retard. En aquest cas la primera línia no ens indica el nombre total de línies, per tant el que fem és saltar-la amb un *if* i seguir llegint amb *fgets* fins que el resultat sigui NULL. Per cada línia del fitxer volem obtenir les dades situades a les columnes 15, 17 i 18. Per fer-ho hem creat una funció ***split*** que rep per paràmetre la línia a explorar, la posició de la columna que volem obtenir i un punter a la variable *size*. La funció compta caràcter per caràcter fins que arriba a una coma o al final de la línia, modifica el valor de la variable size i retorna una referència a la posició després de la coma.

Per cada element de la llista de vols cridem a la funció *split* tres cops, i amb l'ajuda d'una variable auxiliar obtenim la mida de la informació de la columna, la qual cosa ens permet fer un malloc i obtenir les dades amb un *memcpy*. Un cop obtingudes les tres dades busquem el node a l'arbre (utilitzant la dada 'origen') i explorem la linked-list del node per veure si conté el 'destí'. De ser així, incrementem el paràmetre del nombre de vols i sumem el retard. En cas contrari reservem memòria per un nou element de la llista amb el nou destí, la inicialitzem i la inserim al final de la llista. Repetim el procés fins acabar de llegir la llista de vols.

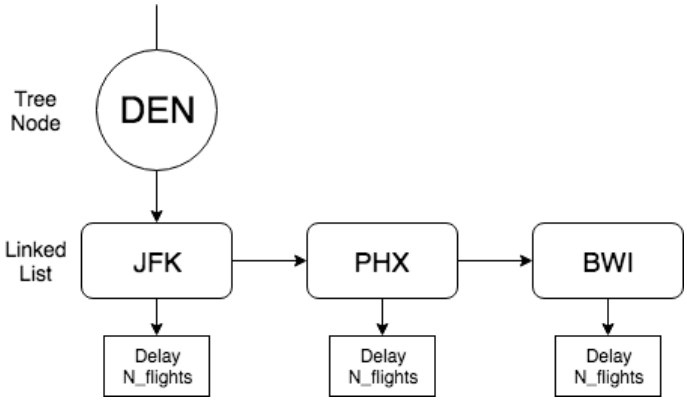

Fig. 2: Esquema d'un node amb Linked-List

Ara només queda dur a terme les tasques demanades per l'enunciat. Primer de tot busquem el node passat per paràmetre utilitzant la funció ***find_node()*** del red-black-tree. Si existeix, agafem la seva llista de vols i anem explorant els resultats, dividint el retard de cada vol entre el nombre de vols i imprimint els resultats per pantalla. Finalment recorrem tots els nodes de l'arbre utilitzant ***postorder*** i imprimim l'aeroport amb el major nombre de destinacions, i tot seguim acabem fent *free* de l'arbre.

Qüestions

Per aquesta pràctica hem fet ús del codi ***red-black-tree*** per ordenar els aeroports seguint una estructura d'arbre. Cada node de l'arbre conté una struct data amb una *key*, que en aquest cas és el nom de l'aeroport. També hem afegit a data una *linked-list* per guardar els vols que surten de cada aeroport. Per tal de comparar les diferents keys a l'hora d'ordenar, hem modificat els mètodes ***compare_key1_equal_to/less_than_key2*** de forma que utilitzin la funció ***strcmp*** per comparar les dos keys. També hem modificat el mètode *free_node_data* per eliminar la linked-list a l'hora de fer free.

De manera simil·lar, també hem modificat la struct *list-data*, la qual contenia una key, per tal d'afegir dos variables més: un int per comptar el delay total i un altre pel nombre de vols. Igualment hem modificat el mètode ***compare_key1_equal_key2*** per tal de que compari les dos keys utilitzant ***strcmp***.

Veiem ara un parell d'experiments amb diferents aeroports:

- Aeroport a buscar **[DEN]**:

Destination: TPA Average delay: 55.666668
Destination: SLC Average delay: 27.285715
Destination: SEA Average delay: -0.545455
Destination: PHX Average delay: 18.437500
Destination: OKC Average delay: 7.500000
Destination: OAK Average delay: 13.538462
Destination: MDW Average delay: 17.437500
Destination: MCO Average delay: 13.000000
Destination: MCI Average delay: 13.000000
Destination: LAS Average delay: 17.631578
Destination: HOU Average delay: 23.777779
Destination: BWI Average delay: 16.000000
Destination: BNA Average delay: 24.833334
Destination: AUS Average delay: -4.333333
Destination: AMA Average delay: 13.833333
Destination: ABQ Average delay: 13.888889

- Aeroport a buscar **[ATL]:**

There are no flights from this airport

Valgrind

Utilitzem Valgrind per comprovar que la memòria s'allibera correctament. Veiem que no dóna errors: