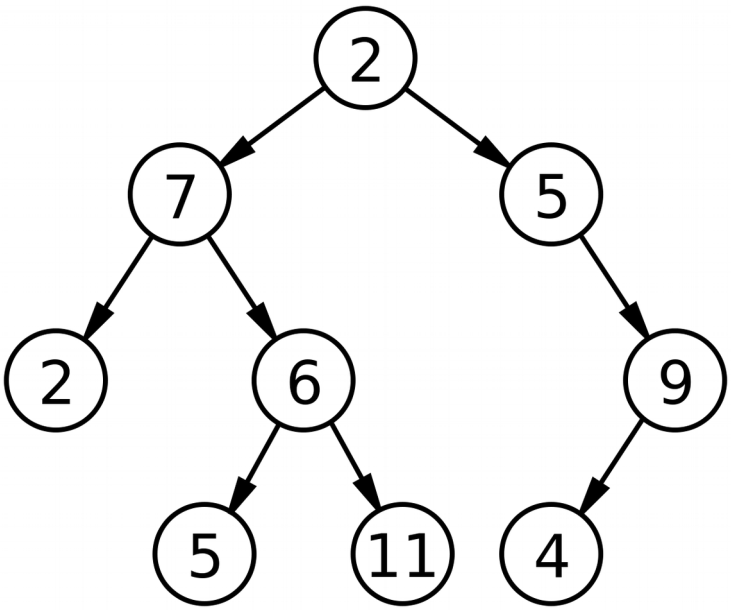
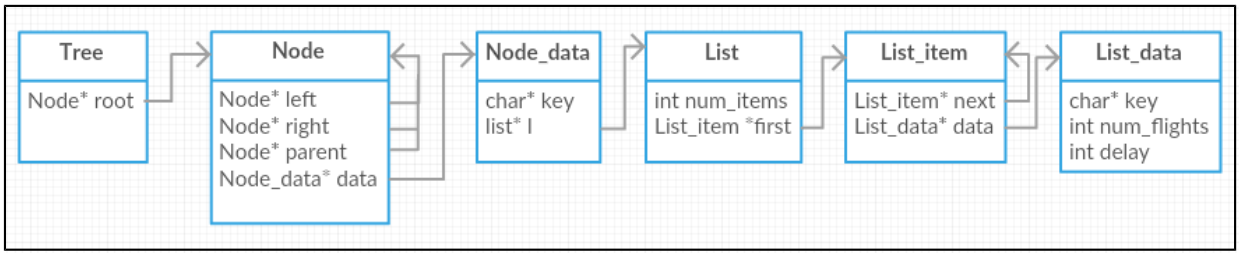
Guardat i processat de dades de vols

Sistemes Operatius II - Pràctica 2

Xavier de Juan Pulido
David de la Osa Bañales



Comenteu els camps membres de l'estructura del node de l'arbre. Indiqueu quines funcions heu modificat (i com les heu modificat) per tal de permetre que l'arbre binari indexi per cadenes de caràcters. Feu el mateix pels camps membres de la llista indexada.



Camps membres de l'estructura de l'arbre:

Per representar l'arbre tenim definida una estructura 'rb_tree' la qual conté un node, el node root, un punter que apunta al primer element de l'arbre.
Tenim una estructura 'node' per representar els nodes de l'arbre que conté: un node 'parent', el qual seria el node anterior, un node 'left' i un node 'right', els quals serien els nodes fills d'un node.

A més per guardar les dades d'aquest node tenim definida una estructura 'node_data', la qual conté la informació que creiem important guardar: una 'key' del tipus 'char *', per tal d'identificar el node, i una llista del tipus 'linked-list' per guardar totes les destinacions. És a dir, en el camp 'key' guardem l'aeroport d'origen, el qual ve representat per 3 chars (per exemple 'IND') i a l'altre camp guardem una llista dels aeroports destí i aquí guardem la informació dels vols.

Camps membres de la llista indexada:

La llista es tracta d'una estructura definida tal que conté el número de ítems, 'num_items', que seria el nombre de destins de l'aeroport origen indicat a les dades de l'arbre, i també hi guardem un punter de tipus 'list_item' que apunta al primer element de la llista, 'first'.

Hem definit els 'list_item' com una estructura per representar cada destí que conté: un punter de tipus 'list_item' que apunta al següent element de la llista i una estructura 'list_data' per guardar la informació de l'element actual.

Aquesta estructura 'list_data' ve definida per 3 elements: una 'key' del tipus 'char *' per indicar de quin destí es tracta (per exemple 'BWI', igual que la 'key' de les dades del node de l'arbre) i, a més, guardem el número de vols que té el destí en concret i una suma dels retards de tots els vols a aquesta destinació.

Modificacions a l'arbre perquè indexi per cadenes de caràcters:

En primer lloc, hem modificat el tipus de clau pel qual indexa l'arbre (RBTREE_KEY_TYPE) de int a char*.

En el moment en què canviem el tipus de la clau hem hagut de modificar altres funcions ja que en el moment en què utilitzaven el valor de la clau era de tipus int i no pas char *.
Aquestes funcions són:

1) **compare_key1_less_than_key2(key1,key2) i compare_key1_equal_to_key2(key1, key2):** aquestes dues funcions comparen si key1 és menor a key2 i si key1 és igual a key2 respectivament. Hem modificat aquestes dues funcions tal que es comparin les quals amb strcmp(key1,key2), d'aquesta manera si es retorna un valor negatiu a strcmp, tenim que key1 és menor a key2 pel que fa al seu contingut, i si la funció strcmp retorna 0 indica que són iguals

2) **free_node_data(data):** com hem modificat la clau perquè sigui del tipus char * i hem afegit com a element de node_data la llista indexada, hem d'alliberar memòria per ambdós elements. Hem afegit alliberament de memòria de la clau i executem delete_list perquè s'alliberi la memòria dels seus elements membres i finalment alliberem memòria per l'element llista de node_data.

Això és tot el que hem hagut de modificar del codi d'arbre binari donat. Com utilitzem codi de la llista indexada a l'arbre hem hagut de fer un include de la linked-list per tal de poder utilitzar-la als fitxers de l'arbre. A red-black-tree.h hem afegit #include "linked-list.h".

Modificacions de la llista indexada:

Per dur a terme la pràctica també hem hagut de fer unes modificacions a la llista indexada.
Igual que a l'arbre hem modificat el RBTREE_KEY_TYPE de int a char *. Hem afegit algun camp a les dades del node els quals he explicat anteriorment.

Al modificar el tipus de dada de la key hem hagut de fer les modificacions següents:

1) **compare_key1_equal_key2(key1,key2):** igual que abans hem hagut de modificar aquest comportament utilitzant la funció strcmp(key1,key2) per comparar el contingut de les keys, si la funció strcmp retorna 0 indica que són iguals.

2) **free_list_data(data):** igual que abans hem hagut d'afegir 'free(data→key)' per alliberar la memòria a la qual apuntava key per ser un char *.

Fins aquí les modificacions que hem hagut de fer sobre el codi de 'linked-list'.

En ambdós codis: arbre binari i llista indexada hem hagut d'incloure la llibreria string.h per tal de poder utilitzar la funció strcmp.

Indiqueu algun dels resultats que obteniu amb el fitxer que se us proporciona amb la pràctica. Proveu, per exemple, amb els aeroports de DEN (el corresponent a Denver) o ATL (Atlanta)

```
davids-MacBook-Pro:src dave$ ./main dades/dades.csv aeroports/aeroports.csv DEN
1)Media de retardos para DEN
Retardos para el aeropuerto DEN
        TPA --  55.667 minutos
        SLC --  27.286 minutos
        SEA --  -0.545 minutos
        PHX --  18.438 minutos
        OKC --  7.500 minutos
        OAK --  13.538 minutos
        MDW --  17.438 minutos
        MCO --  13.000 minutos
        MCI --  13.000 minutos
        LAS --  17.632 minutos
        HOU --  23.778 minutos
        BWI --  16.000 minutos
        BNA --  24.833 minutos
        AUS --  -4.333 minutos
        AMA --  13.833 minutos
        ABQ --  13.889 minutos

2)Aeropuerto con más destinos
Aeropuerto con mas destinos: LAS, destinos 54
davids-MacBook-Pro:src dave$ ./main dades/dades.csv aeroports/aeroports.csv ATL
1)Media de retardos para ATL
Retardos para el aeropuerto ATL

2)Aeropuerto con más destinos
Aeropuerto con mas destinos: LAS, destinos 54
davids-MacBook-Pro:src dave$
```

Passant-li com a paràmetre l'aeroport de DEN i ATL obtenim el següent output:

En tots el casos que hem provat el programa retorna el que s'espera d'aquest.