SISTEMES OPERATIUS 2

- PRÀCTICA 2 -

UNIVERSITAT DE BARCELONA

Pol Safont
Jaume Campderros

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.

Primer hem definit al header el key tipe com char *.

```
#define RBTREE_KEY_TYPE char*
// or #define RBTREE_KEY_TYPE char[4] ???
#include "linked-list.h"
```

A l'estructura del node hem creat una variable int per contar el nombre de destins i un punter a la llista.

```
typedef struct node_data_
{
    // The variable used to index the tree has to be called "key".
    // The type may be any you want (float, char *, etc)
    RBTREE_KEY_TYPE key;

    // This is the additional information that will be stored
    // within the structure. You may adapt it to your needs:
    // add or remove fields as you need.

    // int counter of flights
    int num_destins;

    // pointer to linked list TODO
    list *l;

} node_data;
```

En quant al fitxer .c i les funcions per comparar les dues claus, s'han modificat amb strcmp per que pugui comparar cadenes de caràcters.

```
static int compare_key1_less_than_key2(RBTREE_KEY_TYPE key1, RBTREE_KEY_TYPE key2)
{
    int rc;

    rc = 0;

    if (strcmp(key1,key2) < 0)
        rc = 1;

    return rc;
}
```

L’altre funció de comparar s’ha modificat de la mateixa manera.

En quant a l’estructura de la linked list hem fet pràcticament les mateixes modificacions, per començar hem cambiat el tipus al .h.

```
#define LIST_KEY_TYPE char*
```

Seguidament a la estructura, hem agregat un nou camp per contar el delay.

```
typedef struct list_data_ {
  // The variable used to index the list has to be called "key".
  LIST_KEY_TYPE key;

  // This is the additional information that will be stored
  // within the structure. This additional information is associated
  // to the key. You may include any field you may need useful.
  int num_times;
  int delay;

} list_data;
```

Per les funcions a l’arxiu .c hem fet els mateixos canvis que al tree. Hem utilitzat la funció strcmp per comparar les cadenes.

A l’hora de alliberar memoria hem modificat les funcions free_node_data i free_list_data.

Per asegurar-nos que tota la memoria dinamica sigui lliberada, en free_node_data per cridem:

```
delete_list(data->l);
free(data->l) ;
free(data->key);
free(data);
```

I en free_list_data cridem:

```
free(data->key);
free(data);
```

Indiqueu algun dels resultats que obteniu amb el fitxer que se us proporciona amb la pràctica.

Resultats obtinguts introduïnt com a origen LAX:

Retards mitjos desde Aeroport LAX:
TUS retard de 29.2 minuts.
STL retard de 18.3 minuts.
SMF retard de 42.0 minuts.
SLC retard de 25.8 minuts.
SJC retard de 41.0 minuts.
SFO retard de 95.7 minuts.
SAT retard de 12.8 minuts.
RNO retard de 43.1 minuts.
PHX retard de 33.9 minuts.
OAK retard de 36.9 minuts.
MDW retard de 10.1 minuts.
MCI retard de 24.0 minuts.
LAS retard de 49.9 minuts.
HOU retard de 41.2 minuts.
ELP retard de 40.6 minuts.
BNA retard de 19.3 minuts.
AUS retard de 15.2 minuts.
ABQ retard de 34.1 minuts.
Aeroport amb major nombre de destins es LAS amb un total de 54 destins

Resultats obtinguts introduïnt com a origen DEN:

Retards mitjos desde Aeroport DEN:
TPA retard de 55.7 minuts.
SLC retard de 27.3 minuts.
SEA retard de -0.5 minuts.
PHX retard de 18.4 minuts.
OKC retard de 7.5 minuts.
OAK retard de 13.5 minuts.
MDW retard de 17.4 minuts.
MCO retard de 13.0 minuts.
MCI retard de 13.0 minuts.
LAS retard de 17.6 minuts.
HOU retard de 23.8 minuts.
BWI retard de 16.0 minuts.
BNA retard de 24.8 minuts.
AUS retard de -4.3 minuts.
AMA retard de 13.8 minuts.
ABQ retard de 13.9 minuts.

Resultats amb ATL.

Podem veure que no hi han trajectes amb origen ATL.

Retards mitjos desde Aeroport ATL: