universidade de aveiro deti departamento de electrónica, telecomunicações e informática

Relatório do 2º Projeto de Algoritmo e Estrutura de Dados (AED)

Projeto: **Multi-ordered trees**

Grupo: Guilherme Dias (nº 103128), 50%
Tomás Almeida (nº 103300), 50%

Data de entrega: 1 de Fevereiro 2022

Docentes: Tomás Oliveira e Silva (TP) e Pedro Lavrador (P)

Índice

* -> Anexo inclui o código multi_ordered_tree, bem como o código matlab usado para a criação de todos os gráficos, e alguns testes de código.

Introdução

Este trabalho tem como objetivo encontrar uma solução para guardar/processar dados, e processá-los de forma rápida e eficaz. Implementámos uma solução baseada em binary trees, que permite criar vários branches com dados de cada pessoa (Primeiro e último nome, código postal, número de telemóvel e número da segurança social), e possibilita o acesso, sort e search desses dados de forma eficiente.

Binary Trees

Foram criadas 4 funções para cumprir os objetivos do trabalho:

- tree_insert(), que permite adicionar à árvore novos ramos de dados (insertion routine).

- tree_node_t *find(), que permite percorrer e procurar por dados na árvore (search routine).

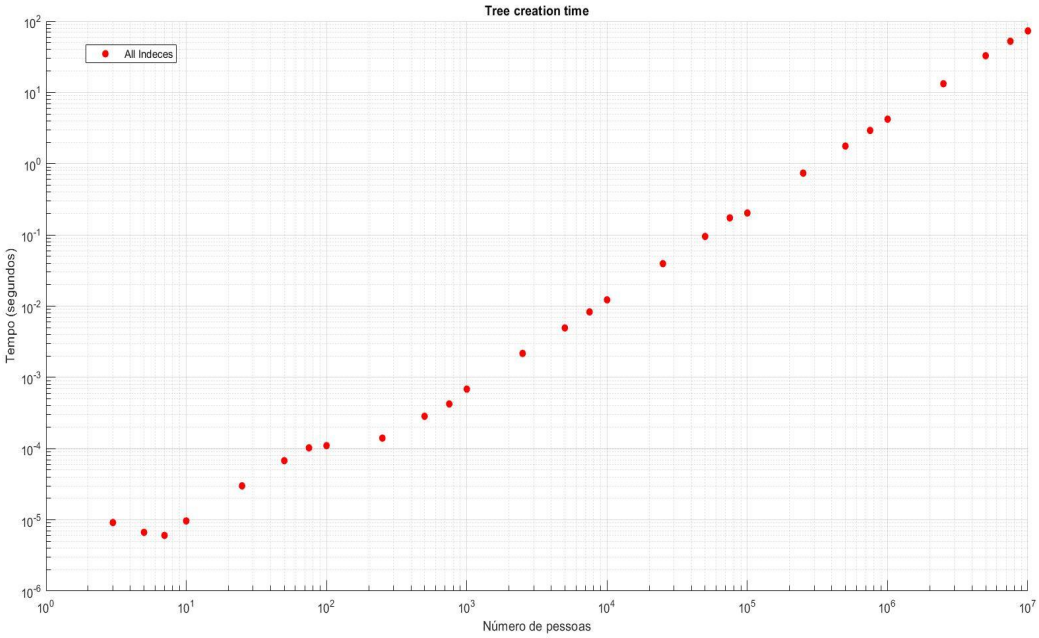
- tree_depth(), que mede recursivamente a profundidade máxima da árvore.

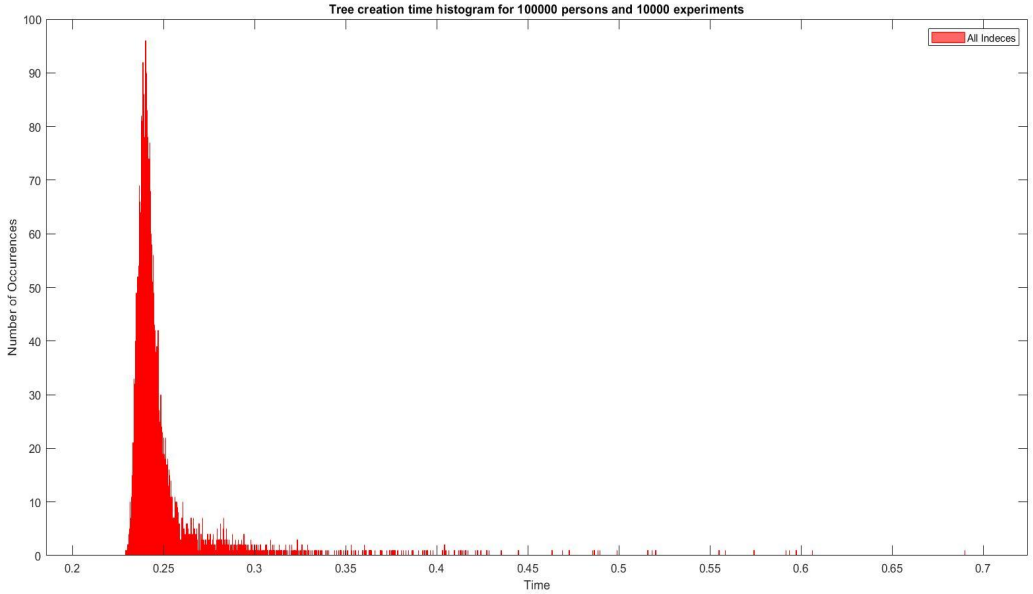
- list(), que permite listar os dados das pessoas no terminal (transverse the tree).

Código tree_insert()

-Permite adicionar à árvore novos ramos de dados, ou criar uma nova árvore. Esta função utiliza uma outra que compara nós, e que vai ajudar a decidir se o nó a adicionar será adicionado ao left ou ao right branch.

```
61  //
62  // tree insertion routine (place your code here)
63  //
64
65  void tree_insert(tree_node_t **link, tree_node_t *node, int main_idx){
66
67    if(*link == NULL)
68      *link = node;
69    else if(compare_tree_nodes(*link,node,main_idx) > 0)
70      tree_insert(&((*link)->left[main_idx]),node,main_idx);
71    else
72      tree_insert(&((*link)->right[main_idx]),node,main_idx);
73  }
74
```

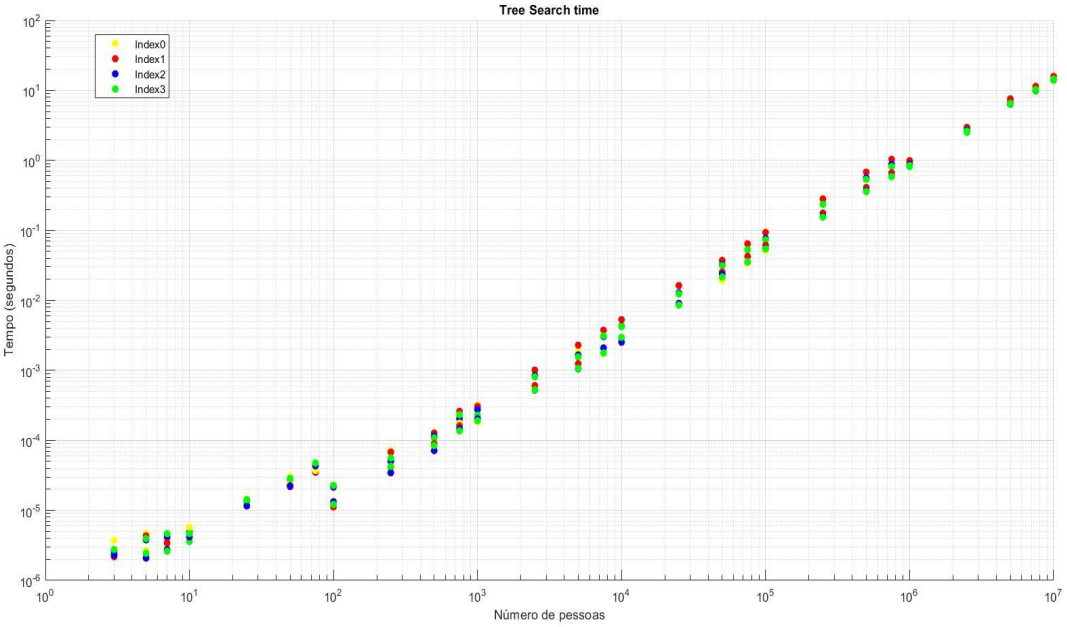

Tree creation time
All Indeces
Tempo (segundos)
Número de pessoas

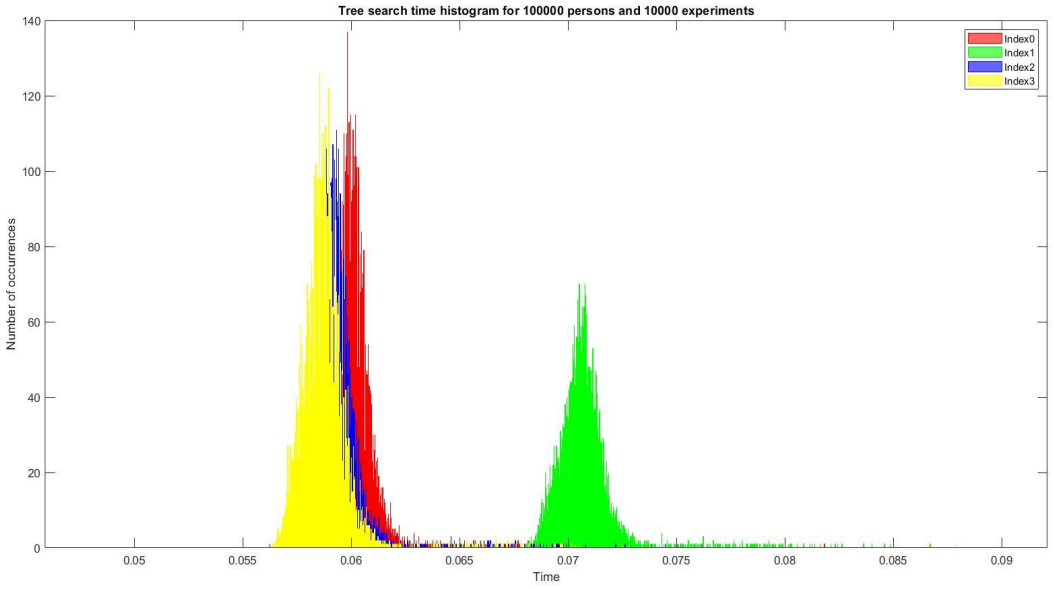

Tree creation time histogram for 100000 persons and 10000 experiments
All Indeces
Number of Occurrences
Time

Código tree_node_t *find()

- Permite percorrer e procurar por dados na árvore, utiliza também a função de comparação para decidir se deve prosseguir pelos right ou left branches, em busca do(s) node(s) que se pretende encontrar.

```
76  //
77  // tree search routine (place your code here)
78  //
79
80  tree_node_t *find(tree_node_t *link, tree_node_t *node, int main_idx) {
81    if(link == NULL){
82      return NULL;
83    }
84    int compare = compare_tree_nodes(link,node,main_idx);
85    if(compare > 0)
86      return find(link->left[main_idx], node, main_idx);
87    else if (compare < 0)
88      return find(link->right[main_idx], node, main_idx);
89    else
90      return link;
91  }
92
```

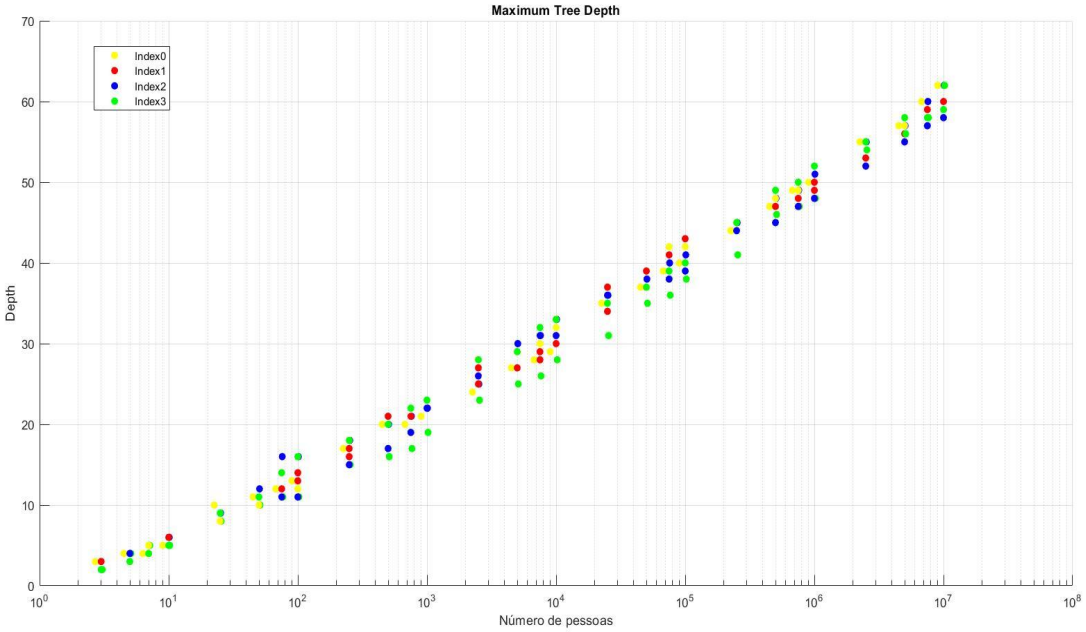

Tree Search time
Index0
Index1
Index2
Index3
Tempo (segundos)
Número de pessoas

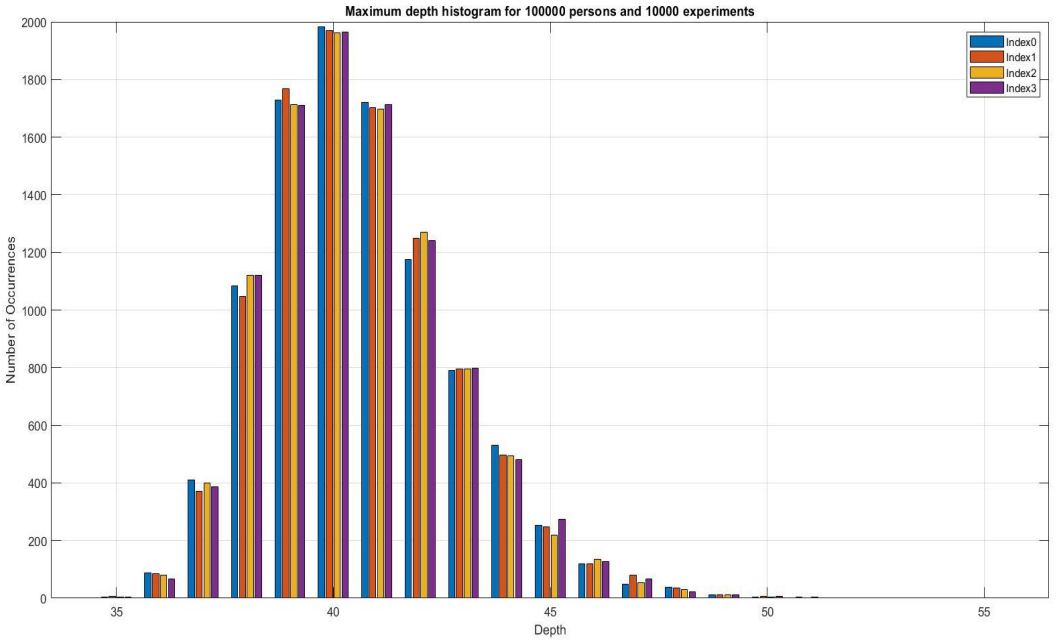

Tree search time histogram for 100000 persons and 10000 experiments
Index0
Index1
Index2
Index3
Number of occurrences
Time

Código tree_depth()

-Mede recursivamente a profundidade máxima da árvore.

```
94   //
95   // tree depdth
96   //
97
98   int tree_depth(tree_node_t *link, int main_idx){
99
100    if (link == NULL)
101      return 0;
102
103    //Recursively calculates the depth
104
105    int left_depth = tree_depth(link->left[main_idx], main_idx);
106    int right_depth = tree_depth(link->right[main_idx], main_idx);
107
108
109    if (left_depth > right_depth){
110      return left_depth +1;
111    }
112
113    return right_depth+1 ;
114  }
115
```


Maximum Tree Depth
Index0
Index1
Index2
Index3
Depth
Número de pessoas


Maximum depth histogram for 100000 persons and 10000 experiments
Index0
Index1
Index2
Index3
Number of Occurrences
Depth

Conclusão

Após a realização deste trabalho, ficamos a saber mais sobre Binary Trees, como percorrê-las, e como podem ser úteis no que toca ao armazenamento e processamento de dados.

Também é de notar a evolução no que toca ao conhecimento da linguagem C, e também em relação à gestão de tempo na realização de trabalhos de grupo.

Bibliografia

https://www.mathworks.com/help/matlab/ref/scatter.html

https://www.geeksforgeeks.org/binary-tree-data-structure/

https://www.programiz.com/dsa/binary-tree

https://www.mathworks.com/help/matlab/ref/matlab.graphics.chart.primitive.histogram.html

Anexo

Código multi_ordered_tree :

```c
//
// AED, January 2022
//
// Solution of the second practical assignement (multi-ordered tree)
//
// Place your student numbers and names here
//Tomás Almeida - 103300/ Guilherme Dias - 103128

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "AED_2021_A02.h"


//
// the custom tree node structure
//
// we want to maintain three ordered trees (using the same nodes!), so we need three left and three right pointers
// so, when inserting a new node we need to do it three times (one for each index), so we will end upo with 3 three roots
//

typedef struct tree_node_s
{
  char name[MAX_NAME_SIZE + 1];                       // index 0 data item
  char zip_code[MAX_ZIP_CODE_SIZE + 1];               // index 1 data item
  char telephone_number[MAX_TELEPHONE_NUMBER_SIZE + 1]; // index 2 data item
  char ssn[MAX_SSN_NUMBER_SIZE+1];                      // index 3 data item
  struct tree_node_s *left[4];                        // left pointers (one for each index) ---- left means smaller
  struct tree_node_s *right[4];                       // right pointers (one for each index) --- right means larger
  long depth;                                         // depth of the tree
}
tree_node_t;
```

```c
//
// the node comparison function (do not change this)
//

int compare_tree_nodes(tree_node_t *node1,tree_node_t *node2,int main_idx)
{
  int i,c;

  for(i = 0;i < 3;i++)
  {
    if(main_idx == 0)
      c = strcmp(node1->name,node2->name);
    else if(main_idx == 1)
      c = strcmp(node1->zip_code,node2->zip_code);
    else if (main_idx == 2)
      c = strcmp(node1->telephone_number,node2->telephone_number);
    else
      c = strcmp(node1->ssn,node2->ssn);
    if(c != 0)
      return c; // different on this index, so return
    main_idx = (main_idx == 3) ? 0 : main_idx + 1; // advance to the next index
  }
  return 0;
}


//
// tree insertion routine (place your code here)
//

void tree_insert(tree_node_t **link, tree_node_t *node, int main_idx){

  if(*link == NULL)
    *link = node;
  else if(compare_tree_nodes(*link,node,main_idx) > 0)
    tree_insert(&((*link)->left[main_idx]),node,main_idx);
  else
    tree_insert(&((*link)->right[main_idx]),node,main_idx);
}


//
// tree search routine (place your code here)
//

tree_node_t *find(tree_node_t *link, tree_node_t *node, int main_idx) {
  if(link == NULL){
    return NULL;
  }
  int compare = compare_tree_nodes(link,node,main_idx);
  if(compare > 0)
    return find(link->left[main_idx], node, main_idx);
  else if (compare < 0)
    return find(link->right[main_idx], node, main_idx);
  else
    return link;
}

```

```
94  //
95  // tree depdth
96  //
97
98  int tree_depth(tree_node_t *link, int main_idx){
99
100   if (link == NULL)
101     return 0;
102
103   //Recursively calculates the depth
104
105   int left_depth = tree_depth(link->left[main_idx], main_idx);
106   int right_depth = tree_depth(link->right[main_idx], main_idx);
107
108
109   if (left_depth > right_depth){
110     return left_depth +1;
111   }
112
113   return right_depth+1 ;
114 }
115
116
117 int counter = 1;
118 void list(tree_node_t *link, int main_idx){
119
120   if(link != NULL){
121     list(link->left[main_idx],main_idx);
122     printf("Person #%d\n",counter++);
123     printf("    name --------------------- %s\n", link->name);
124     printf("    zip code ----------------- %s\n", link->zip_code);
125     printf("    telephone number --------- %s\n", link->telephone_number);
126     printf("    social security number --- %s\n", link->ssn);
127     list(link->right[main_idx],main_idx);
128   }
129 }
130
131
132
133  void *find_zip_code(tree_node_t *link, char *zip){
134
135    if(link != NULL){
136      if (strcmp(link->zip_code, zip) == 0){
137        find_zip_code(link->left[1],zip);
138        printf("Person #%d\n",counter++);
139        printf("    name --------------------- %s\n", link->name);
140        printf("    zip code ----------------- %s\n", link->zip_code);
141        printf("    telephone number --------- %s\n", link->telephone_number);
142        printf("    social security number --- %s\n", link->ssn);
143        find_zip_code(link->right[1],zip);
144      }
145      else{
146        find_zip_code(link->left[1],zip);
147        find_zip_code(link->right[1],zip);
148      }
149    }
150  }
151
```

```c
int main(int argc,char **argv)
{
  double dt;

  // process the command line arguments
  if(argc < 4)
  {
    fprintf(stderr,"Usage: %s student_number number_of_persons [options ...]\n",argv[0]);
    fprintf(stderr,"Recognized options:\n");
    fprintf(stderr,"  -list[N]              # list the tree contents, sorted by key index N (the default is index 0)\n");
    // place a description of your own options here
    return 1;
  }
  int student_number = atoi(argv[1]);
  if(student_number < 1 || student_number >= 1000000)
  {
    fprintf(stderr,"Bad student number (%d) --- must be an integer belonging to [1,1000000{\n",student_number);
    return 1;
  }
  int n_persons = atoi(argv[2]);
  if(n_persons < 3 || n_persons > 10000000)
  {
    fprintf(stderr,"Bad number of persons (%d) --- must be an integer belonging to [3,10000000]\n",n_persons);
    return 1;
  }
  // generate all data
  tree_node_t *persons = (tree_node_t *)calloc((size_t)n_persons,sizeof(tree_node_t));
  if(persons == NULL)
  {
    fprintf(stderr,"Output memory!\n");
    return 1;
  }
  aed_srandom(student_number);
  for(int i = 0;i < n_persons;i++)
  {
    random_name(&(persons[i].name[0]));
    random_zip_code(&(persons[i].zip_code[0]));
    random_telephone_number(&(persons[i].telephone_number[0]));
    random_ssn(&(persons[i].ssn[0]));
    for(int j = 0;j < 4;j++)
      persons[i].left[j] = persons[i].right[j] = NULL; // make sure the pointers are initially NULL
  }
  // create the ordered binary trees
  dt = cpu_time();
  tree_node_t *roots[4]; // three indices, three roots
  for(int main_index = 0;main_index < 4;main_index++)
    roots[main_index] = NULL;
  for(int i = 0;i < n_persons;i++)
    for(int main_index = 0;main_index < 4;main_index++)
      tree_insert(&roots[main_index], &persons[i], main_index) ; // place your code here to insert &(persons[i]) in the t
  dt = cpu_time() - dt;
  printf("Tree creation time (%d persons): %.3es\n",n_persons,dt);
  // search the tree
  for(int main_index = 0;main_index < 4;main_index++)
  {
    dt = cpu_time();
    for(int i = 0;i < n_persons;i++)
    {
      tree_node_t n = persons[i]; // make a copy of the node data
      if(find(roots[main_index], &n , main_index) != &(persons[i])) // place your code here to find a given person, searc
      {
        fprintf(stderr,"person %d not found using index %d\n",i,main_index);
        return 1;
      }
    }
    dt = cpu_time() - dt;
    printf("Tree search time (%d persons, index %d): %.3es\n",n_persons,main_index,dt);
  }
```

```
225   // compute the largest tree depdth
226   for(int main_index = 0;main_index < 4;main_index++)
227   {
228     dt = cpu_time();
229     int depth = tree_depth(roots[main_index],main_index); // place your code here to compute the depth of the tree with nu
230     dt = cpu_time() - dt;
231     printf("Tree depth for index %d: %d (done in %.3es)\n",main_index,depth,dt);
232   }
233   // process the command line optional arguments
234   for(int i = 3;i < argc;i++)
235   {
236     if(strncmp(argv[i],"-list",5) == 0)
237     { // list all (optional)
238       int main_index = atoi(&(argv[i][5]));
239       if(main_index < 0)
240         main_index = 0;
241       if(main_index > 3)
242         main_index = 3;
243       printf("List of persons:\n");
244       (void)list(roots[main_index], main_index); // place your code here to traverse, in order, the tree with number main
245     } else if (strcmp(argv[i],"-find") == 0){
246       printf("List of people with the zip code '%s' is: \n", argv[i+1]);
247       find_zip_code(roots[i], argv[i+1]);
248
249     }
250     // place your own options here
251   }
252   // clean up --- don't forget to test your program with valgrind, we don't want any memory leaks
253   free(persons);
254   return 0;
255 }
```

Testes código:

Sort options:

-list0 : Sort pelo nome.

-list1 : Sort pelo código postal.

-list2 : Sort pelo número de telefone.

-list3 : Sort pelo número de segurança social.

```
guidias@legion-y540-15irh-pg0:~/GitHub_Projects/AEDProject2$ ./multi_ordered_tree 2021 5 -list0
Tree creation time (5 persons): 1.965e-06s
Tree search time (5 persons, index 0): 1.234e-06s
Tree search time (5 persons, index 1): 1.096e-06s
Tree search time (5 persons, index 2): 1.153e-06s
Tree search time (5 persons, index 3): 1.317e-06s
Tree depth for index 0: 4 (done in 5.600e-07s)
Tree depth for index 1: 4 (done in 4.470e-07s)
Tree depth for index 2: 4 (done in 4.590e-07s)
Tree depth for index 3: 4 (done in 5.240e-07s)
List of persons:
Person #1
    name -------------------- Aaron Morgan
    zip code ---------------- 44256 Medina (Medina county)
    telephone number -------- 2034 151 114
    social security number --- 13501054
Person #2
    name -------------------- Anna Thomas
    zip code ---------------- 10468 Bronx (Bronx county)
    telephone number -------- 2065 792 213
    social security number --- 45445945
Person #3
    name -------------------- Cindy Buchanan
    zip code ---------------- 926 San Juan (San Juan county)
    telephone number -------- 7140 229 408
    social security number --- 57562519
Person #4
    name -------------------- Luke Hall
    zip code ---------------- 11215 Brooklyn (Kings county)
    telephone number -------- 7362 997 722
    social security number --- 13129026
Person #5
    name -------------------- Marvin Sanchez
    zip code ---------------- 94122 San Francisco (San Francisco county)
    telephone number -------- 6655 181 342
    social security number --- 53564340

guidias@legion-y540-15irh-pg0:~/GitHub_Projects/AEDProject2$ ./multi_ordered_tree 2021 5 -list1
Tree creation time (5 persons): 1.887e-06s
Tree search time (5 persons, index 0): 1.101e-06s
Tree search time (5 persons, index 1): 8.590e-07s
Tree search time (5 persons, index 2): 8.480e-07s
Tree search time (5 persons, index 3): 8.380e-07s
Tree depth for index 0: 4 (done in 4.380e-07s)
Tree depth for index 1: 4 (done in 3.530e-07s)
Tree depth for index 2: 4 (done in 3.160e-07s)
Tree depth for index 3: 4 (done in 3.340e-07s)
List of persons:
Person #1
    name -------------------- Anna Thomas
    zip code ---------------- 10468 Bronx (Bronx county)
    telephone number -------- 2065 792 213
    social security number --- 45445945
Person #2
    name -------------------- Luke Hall
    zip code ---------------- 11215 Brooklyn (Kings county)
    telephone number -------- 7362 997 722
    social security number --- 13129026
Person #3
    name -------------------- Aaron Morgan
    zip code ---------------- 44256 Medina (Medina county)
    telephone number -------- 2034 151 114
    social security number --- 13501054
Person #4
    name -------------------- Cindy Buchanan
    zip code ---------------- 926 San Juan (San Juan county)
    telephone number -------- 7140 229 408
    social security number --- 57562519
Person #5
    name -------------------- Marvin Sanchez
    zip code ---------------- 94122 San Francisco (San Francisco county)
    telephone number -------- 6655 181 342
    social security number --- 53564340
```

```
guidias@legion-y540-15irh-pg0:~/GitHub_Projects/AEDProject2$ ./multi_ordered_tree 2021 5 -list2
Tree creation time (5 persons): 2.106e-06s
Tree search time (5 persons, index 0): 1.573e-06s
Tree search time (5 persons, index 1): 1.212e-06s
Tree search time (5 persons, index 2): 1.089e-06s
Tree search time (5 persons, index 3): 9.790e-07s
Tree depth for index 0: 4 (done in 3.910e-07s)
Tree depth for index 1: 4 (done in 3.260e-07s)
Tree depth for index 2: 4 (done in 3.470e-07s)
Tree depth for index 3: 4 (done in 3.400e-07s)
List of persons:
Person #1
    name -------------------- Aaron Morgan
    zip code ---------------- 44256 Medina (Medina county)
    telephone number -------- 2034 151 114
    social security number --- 13501054
Person #2
    name -------------------- Anna Thomas
    zip code ---------------- 10468 Bronx (Bronx county)
    telephone number -------- 2065 792 213
    social security number --- 45445945
Person #3
    name -------------------- Marvin Sanchez
    zip code ---------------- 94122 San Francisco (San Francisco county)
    telephone number -------- 6655 181 342
    social security number --- 53564340
Person #4
    name -------------------- Cindy Buchanan
    zip code ---------------- 926 San Juan (San Juan county)
    telephone number -------- 7140 229 408
    social security number --- 57562519
Person #5
    name -------------------- Luke Hall
    zip code ---------------- 11215 Brooklyn (Kings county)
    telephone number -------- 7362 997 722
    social security number --- 13129026

guidias@legion-y540-15irh-pg0:~/GitHub_Projects/AEDProject2$ ./multi_ordered_tree 2021 5 -list3
Tree creation time (5 persons): 1.930e-06s
Tree search time (5 persons, index 0): 1.206e-06s
Tree search time (5 persons, index 1): 9.170e-07s
Tree search time (5 persons, index 2): 7.650e-07s
Tree search time (5 persons, index 3): 8.160e-07s
Tree depth for index 0: 4 (done in 4.120e-07s)
Tree depth for index 1: 4 (done in 3.460e-07s)
Tree depth for index 2: 4 (done in 2.960e-07s)
Tree depth for index 3: 4 (done in 3.590e-07s)
List of persons:
Person #1
    name -------------------- Luke Hall
    zip code ---------------- 11215 Brooklyn (Kings county)
    telephone number -------- 7362 997 722
    social security number --- 13129026
Person #2
    name -------------------- Aaron Morgan
    zip code ---------------- 44256 Medina (Medina county)
    telephone number -------- 2034 151 114
    social security number --- 13501054
Person #3
    name -------------------- Anna Thomas
    zip code ---------------- 10468 Bronx (Bronx county)
    telephone number -------- 2065 792 213
    social security number --- 45445945
Person #4
    name -------------------- Marvin Sanchez
    zip code ---------------- 94122 San Francisco (San Francisco county)
    telephone number -------- 6655 181 342
    social security number --- 53564340
Person #5
    name -------------------- Cindy Buchanan
    zip code ---------------- 926 San Juan (San Juan county)
    telephone number -------- 7140 229 408
    social security number --- 57562519
```

Find:

```
guidias@legion-y540-15irh-pg0:~/GitHub_Projects/AEDProject2$ ./multi_ordered_tree 2021  10000 -find '23452 Virginia Beach (Virginia Beach City coun
ty)'
Tree creation time (10000 persons): 1.864e-02s
Tree search time (10000 persons, index 0): 3.536e-03s
Tree search time (10000 persons, index 1): 4.276e-03s
Tree search time (10000 persons, index 2): 3.702e-03s
Tree search time (10000 persons, index 3): 4.062e-03s
Tree depth for index 0: 30 (done in 2.705e-04s)
Tree depth for index 1: 29 (done in 2.322e-04s)
Tree depth for index 2: 35 (done in 2.362e-04s)
Tree depth for index 3: 34 (done in 2.364e-04s)
List of people with the zip code '23452 Virginia Beach (Virginia Beach City county)' is:
Person #1
    name -------------------- Felicia Meyer
    zip code ---------------- 23452 Virginia Beach (Virginia Beach City county)
    telephone number -------- 1693 086 148
    social security number --- 22038645
Person #2
    name -------------------- Jennifer Andersen
    zip code ---------------- 23452 Virginia Beach (Virginia Beach City county)
    telephone number -------- 1872 007 619
    social security number --- 01723154
Person #3
    name -------------------- Patricia Freeman
    zip code ---------------- 23452 Virginia Beach (Virginia Beach City county)
    telephone number -------- 2300 641 311
    social security number --- 65988115
Person #4
    name -------------------- David Gilmore
    zip code ---------------- 23452 Virginia Beach (Virginia Beach City county)
    telephone number -------- 2319 976 987
    social security number --- 32398684
Person #5
    name -------------------- Richard Robinson
    zip code ---------------- 23452 Virginia Beach (Virginia Beach City county)
    telephone number -------- 2595 124 891
    social security number --- 37458305
Person #6
    name -------------------- Mary Tran
    zip code ---------------- 23452 Virginia Beach (Virginia Beach City county)
    telephone number -------- 2795 475 468
    social security number --- 66577379
```

Código de Histogramas (Matlab):

```
%% Gráfico 2
A2 = load ("./depthIndex0hist.txt");
B2 = load ("./depthIndex1hist.txt");
C2 = load ("./depthIndex2hist.txt");
D2 = load ("./depthIndex3hist.txt");

binc1=(34:56);

o0 = hist(A2,binc1);
o1 = hist(B2,binc1);
o2 = hist(C2,binc1);
o3 = hist(D2,binc1);

figure(1) %Maximum Depth 100k pessoas 10k testes
x = 34:1:56;
y = [o0;o1;o2;o3];
bar(x,y);
grid on;
xlabel('Depth')
ylabel('Number of Occurrences')
legend('Index0','Index1','Index2', 'Index3')
title('Maximum depth histogram for 100000 persons and 10000 experiments');
```

```
%% Gráfico 4

A4 = load ("./creationTimehist.txt");


figure(2) %Creation time histogram 100k pessoas 10k testes
binc2=(0.21:0.0001:0.7);
histogram(A4,binc2,EdgeColor="r",FaceColor="r");
xlabel('Time')
ylabel('Number of Occurrences')
title("Tree creation time histogram for 100000 persons and 10000 experiments");
legend('All Indeces');
```

```
%% Gráfico 6

A6 = load("./searchTime0hist.txt");
B6 = load("./searchTime1hist.txt");
C6 = load("./searchTime2hist.txt");
D6 = load("./searchTime3hist.txt");



figure(3) %Search Time histogram 100k pessoas 10k testes TODO
binc3 = (0.048:0.00001:0.090);
histogram(A6,binc3,EdgeColor="r",FaceColor="r")
hold on;
histogram(B6,binc3,EdgeColor="g",FaceColor="g")
hold on;
histogram(C6,binc3,EdgeColor="b",FaceColor="b")
hold on;
histogram(D6,binc3,EdgeColor="y",FaceColor="y")
title('Tree search time histogram for 100000 persons and 10000 experiments');
legend('Index0','Index1','Index2', 'Index3');
xlabel('Time');
ylabel('Number of occurrences');
```

Código gráficos (Matlab):

```
%% Creation Time
C1 = readmatrix("./creationTime103300.txt");
C2 = readmatrix("./creationTime103128.txt");

figure(1);
scatter(C1(1:28,1),C1(1:28,2),'red','filled');
set(gca,'xscale','log');
set(gca,'yscale','log');

figure(1);
scatter(C2(1:28,1),C2(1:28,2),'red','filled');
grid on;
title("Tree creation time");
xlabel("Número de pessoas");
ylabel("Tempo (segundos)");
set(gca,'xscale','log');
set(gca,'yscale','log');
legend('All Indeces');
```

```
%% Search Time
S1 = readmatrix("./searchTime0.txt");
S2 = readmatrix("./searchTime1.txt");
S3 = readmatrix("./searchTime2.txt");
S4 = readmatrix("./searchTime3.txt");

figure(2);
scatter(S1(:,1),S1(:,2),'yellow','filled')
hold on;
scatter(S2(:,1),S2(:,2),'red','filled');
hold on;
scatter(S3(:,1),S3(:,2),'blue','filled')
hold on;
scatter(S4(:,1),S4(:,2),'green','filled')
set(gca,'xscale','log');
set(gca,'yscale','log');


S5 = readmatrix("./searchTime00.txt");
S6 = readmatrix("./searchTime11.txt");
S7 = readmatrix("./searchTime22.txt");
S8 = readmatrix("./searchTime33.txt");
```

```
figure(2);
scatter(S5(:,1),S5(:,2),'yellow','filled')
hold on;
scatter(S6(:,1),S6(:,2),'red','filled');
hold on;
scatter(S7(:,1),S7(:,2),'blue','filled')
hold on;
scatter(S8(:,1),S8(:,2),'green','filled')
set(gca,'xscale','log');
set(gca,'yscale','log');

grid on;
title("Tree Search time");
xlabel("Número de pessoas");
ylabel("Tempo (segundos)");
legend('Index0','Index1','Index2', 'Index3');
```

```
%% Tree depth

S1 = readmatrix("./depthIndex0.txt");
S2 = readmatrix("./depthIndex1.txt");
S3 = readmatrix("./depthIndex2.txt");
S4 = readmatrix("./depthIndex3.txt");

figure(3);
scatter(S1(:,1)*0.9,S1(:,2),'yellow','filled')
hold on;
scatter(S2(:,1),S2(:,2),'red','filled');
hold on;
scatter(S3(:,1)*1.01,S3(:,2),'blue','filled')
hold on;
scatter(S4(:,1)*1.02,S4(:,2),'green','filled')
set(gca,'xscale','log');



S5 = readmatrix("./depthIndex00.txt");
S6 = readmatrix("./depthIndex11.txt");
S7 = readmatrix("./depthIndex22.txt");
S8 = readmatrix("./depthIndex33.txt");
```

```
figure(3);
scatter(S5(:,1),S5(:,2),'yellow','filled')
hold on;
scatter(S6(:,1),S6(:,2),'red','filled');
hold on;
scatter(S7(:,1),S7(:,2),'blue','filled')
hold on;
scatter(S8(:,1),S8(:,2),'green','filled')
set(gca,'xscale','log');


grid on;
title("Maximum Tree Depth");
xlabel("Número de pessoas");
ylabel("Depth");
legend('Index0','Index1','Index2', 'Index3');
```