UFSJ - Ciências da Computação

Laboratório de Programação 2

Roteiro 12

Nome: Geraldo Arthur Detomi

1.1) Reimplemente os algoritmos de busca sequencial e busca binaria e teste-os comparando os valores finais de numero de comparacoes e tempo de execucao.

roteiro_12/tempo.h

```c
#ifndef TEMPO_H
#define TEMPO_H

#include <stdio.h>
#include <sys/resource.h>
#include <sys/time.h>

// Estrutura para armazenar os tempos de execução
typedef struct Temporizador {
  struct timeval start_tv, end_tv;
  struct rusage start_usage, end_usage;
} Temporizador;

// Inicia a contagem do temporizador
void iniciarTemporizador(Temporizador *t);

// Finaliza a contagem do temporizador
void finalizarTemporizador(Temporizador *t);

// Calcula o tempo real decorrido em segundos
double calcularTempoReal(Temporizador *t);

// Calcula o tempo de CPU em modo usuário em segundos
double calcularTempoUsuario(Temporizador *t);

// Calcula o tempo de CPU em modo sistema/kernel em segundos
double calcularTempoSistema(Temporizador *t);

// Imprime todos os tempos medidos de forma formatada
void imprimirTempos(Temporizador *t);

#endif
```

roteiro_12/tempo.c

```c
#include "tempo.h"

// Inicia a medição do tempo, armazenando os valores atuais
void iniciarTemporizador(Temporizador *t) {
  gettimeofday(&t->start_tv, NULL);
  getrusage(RUSAGE_SELF, &t->start_usage);
}

// Finaliza a medição do tempo, armazenando os valores finais
void finalizarTemporizador(Temporizador *t) {
  gettimeofday(&t->end_tv, NULL);
  getrusage(RUSAGE_SELF, &t->end_usage);
}

double calcularTempo(struct timeval inicio, struct timeval fim) {
  time_t seg = fim.tv_sec - inicio.tv_sec;
  suseconds_t microseg = fim.tv_usec - inicio.tv_usec;

  // Ajusta caso microsegundos do fim sejam menores que os do início
  if (microseg < 0) {
    seg -= 1;
    microseg += 1000000;
  }

  return (double)seg + (double)microseg / 1e6;
}

// Calcula diferença entre tempos reais (start e end) em segundos
double calcularTempoReal(Temporizador *t) {
  return calcularTempo(t->start_tv, t->end_tv);
```

```
}

// Calcula diferença entre tempos de sistema (start e end) em segundos
double calcularTempoSistema(Temporizador *t) {
  return calcularTempo(t->start_usage.ru_stime, t->end_usage.ru_stime);
}

// Imprime todos os tempos de execução de forma legível
void imprimirTempos(Temporizador *t) {
  printf("Tempo de execução:\n");
  printf("Tempo Real: %.8f segundos\n", calcularTempoReal(t));
  printf("Tempo Sistema: %.8f segundos\n", calcularTempoSistema(t));
}
```

roteiro_12/1-1.c

```c
#include "tempo.h"

#include <stdio.h>
#include <stdlib.h>

void liberar_vetor(int **v) {
  free(*v);
  *v = NULL;
}

int *copiaVetor(int *v, int n) {
  int i;
  int *v2;
  v2 = (int *)malloc(n * sizeof(int));
  for (i = 0; i < n; i++)
    v2[i] = v[i];
  return v2;
}

void imprimeVetor(int *v, int n) {
  int i, prim = 1;
  printf("[");
  for (i = 0; i < n; i++)
    if (prim) {
      printf("%d", v[i]);
      prim = 0;
    } else
      printf(", %d", v[i]);
  printf("]\n");
}

int buscaSequencial(int *v, int n, int elem, int *comp) {
  int i;
  for (i = 0; i < n; i++) {
    (*comp)++;
    if (v[i] == elem)
      return i; // Elemento encontrado
  }
  return -1; // Elemento encontrado
}

int it_buscaBinaria(int *v, int ini, int fim, int elem, int *comp) {
  int meio;
  while (ini <= fim) {
    meio = (ini + fim) / 2;
    (*comp)++;
    if (elem == v[meio])
      return meio;
    else if (elem < v[meio])
      fim = meio - 1;
    else
      ini = meio + 1;
  }
  return -1;
}

int compara_crescente(const void *a, const void *b) {
  int x = *(int *)a;
  int y = *(int *)b;

  if (x < y)
    return -1;
  if (x > y)
    return 1;

  return 0;
```

```c
}

enum algoritmos_para_teste {
  PESQUISA_SEQUENCIAL = 0,
  PESQUISA_BINARIA,
};

#define QTD_ALGORITMOS_TESTE 2

int main() {
  int n;

  printf("Digite o tamanho do vetor:");
  scanf("%d", &n);

  int *array_original = (int *)malloc(n * sizeof(int));

  for (int i = 0; i < n; i++) {
    scanf("%d", &array_original[i]);
  }

  qsort(array_original, n, sizeof(int), compara_crescente);

  int value;
  printf("Digite um valor para buscar no vetor:\n");
  scanf("%d", &value);

  printf("Vetor final da busca:\n");
  imprimeVetor(array_original, n);

  for (int i = 0; i < QTD_ALGORITMOS_TESTE; i++) {
    int comp = 0, index;

    Temporizador tempo_teste;
    iniciarTemporizador(&tempo_teste);

    printf("Elemento buscado = %d\n", value);

    switch (i) {
    case PESQUISA_SEQUENCIAL:

      printf("---------- Busca Sequencial ----------\n");

      index = buscaSequencial(array_original, n, value, &comp);

      if (index != -1) {
        printf("Elemento encontrado na indice %d\n", index);
      } else {
        printf("Elemento não encontrado\n");
      }

      break;
    case PESQUISA_BINARIA:
      printf("---------- Busca binária ----------\n");

      index = it_buscaBinaria(array_original, 0, n - 1, value, &comp);

      if (index != -1) {
        printf("Elemento encontrado na posição %d\n", index);
      } else {
        printf("Elemento não encontrado\n");
      }

      break;
    }

    finalizarTemporizador(&tempo_teste);
    printf("Quantidade de comparações realizadas: %d\n", comp);
    imprimirTempos(&tempo_teste);
    printf("--------------------------------\n");
  }

  liberar_vetor(&array_original);
  return 0;

  return 0;
}
```

Saída do terminal:

```
arthurdetomi at arthurdetomi-System-Product-Name in ~/Documents/UFSJ-Graduacao/UFSJ-2025_1/Lab_Prog_2/rot
.:: ./1-1.out < input/1000-misturado.txt
Digite o tamanho do vetor:Digite um valor para buscar no vetor:
Vetor final da busca:
[1, 2, 3, 4, 4, 8, 11, 12, 13, 13, 15, 16, 19, 20, 20, 24, 26, 27, 28, 33, 34, 36, 36, 37, 37, 39, 43, 44,
 92, 93, 99, 102, 103, 104, 106, 109, 112, 115, 116, 116, 119, 122, 130, 130, 135, 137, 140, 147, 147, 148,
9, 192, 192, 195, 196, 198, 200, 200, 201, 204, 209, 210, 211, 211, 214, 217, 219, 219, 220, 222, 223, 224,
5, 258, 260, 261, 265, 266, 268, 269, 270, 273, 275, 276, 276, 280, 280, 282, 286, 295, 295, 298, 299, 305,
8, 351, 353, 354, 357, 358, 359, 363, 363, 364, 365, 366, 370, 370, 373, 374, 379, 382, 382, 386, 387, 387,
6, 417, 418, 418, 424, 425, 431, 432, 433, 435, 438, 443, 448, 449, 450, 451, 453, 454, 456, 456, 460, 463,
3, 484, 487, 487, 493, 494, 497, 498, 499, 499, 503, 509, 510, 513, 517, 520, 523, 525, 526, 526, 530, 534,
4, 584, 585, 589, 590, 590, 591, 592, 593, 594, 594, 595, 596, 598, 598, 598, 600, 604, 606, 608, 609, 610,
6, 667, 669, 670, 671, 672, 672, 673, 674, 679, 681, 681, 689, 690, 691, 693, 694, 695, 696, 700, 700, 700,
0, 734, 734, 738, 740, 741, 746, 747, 750, 752, 753, 755, 755, 768, 770, 772, 773, 775, 776, 777, 777, 778,
6, 827, 828, 830, 830, 836, 837, 838, 842, 846, 847, 851, 852, 853, 853, 855, 855, 857, 861, 862, 864, 865,
5, 897, 902, 902, 903, 905, 906, 907, 907, 909, 909, 910, 912, 915, 915, 916, 918, 919, 919, 919, 931, 932,
8, 962, 962, 963, 963, 964, 964, 967, 968, 968, 975, 976, 980, 984, 985, 990, 990, 991, 992, 992, 992, 994,
 1032, 1034, 1035, 1039, 1040, 1041, 1051, 1052, 1055, 1058, 1058, 1059, 1059, 1060, 1061, 1062, 1063, 1066
8, 1102, 1103, 1106, 1107, 1110, 1110, 1112, 1113, 1114, 1116, 1120, 1121, 1122, 1128, 1131, 1131, 1131, 11
165, 1170, 1171, 1172, 1175, 1177, 1179, 1179, 1179, 1180, 1181, 1181, 1182, 1182, 1182, 1183, 1185, 1188,
 1215, 1215, 1216, 1219, 1220, 1222, 1222, 1222, 1225, 1226, 1226, 1226, 1226, 1230, 1231, 1234, 1236, 1236
3, 1274, 1275, 1276, 1281, 1281, 1288, 1291, 1292, 1294, 1296, 1300, 1301, 1303, 1307, 1309, 1311, 1317, 13
340, 1341, 1341, 1345, 1345, 1347, 1353, 1358, 1365, 1366, 1368, 1369, 1369, 1372, 1372, 1374, 1375, 1382,
 1416, 1417, 1422, 1423, 1427, 1428, 1428, 1430, 1430, 1430, 1432, 1434, 1435, 1436, 1437, 1443, 1445, 1447
5, 1480, 1481, 1481, 1482, 1483, 1485, 1486, 1491, 1494, 1496, 1497, 1504, 1504, 1507, 1507, 1507, 1508, 15
524, 1524, 1527, 1528, 1534, 1536, 1538, 1539, 1539, 1541, 1546, 1547, 1549, 1558, 1558, 1560, 1561, 1561,
 1585, 1586, 1588, 1589, 1589, 1590, 1599, 1601, 1601, 1607, 1609, 1610, 1611, 1615, 1619, 1622, 1623, 1625
3, 1664, 1669, 1672, 1672, 1673, 1673, 1673, 1675, 1676, 1679, 1682, 1684, 1689, 1689, 1689, 1691, 1692, 16
709, 1715, 1720, 1723, 1726, 1727, 1729, 1732, 1737, 1740, 1740, 1744, 1745, 1748, 1749, 1754, 1756, 1760,
 1787, 1791, 1795, 1797, 1798, 1803, 1803, 1804, 1805, 1806, 1806, 1807, 1808, 1814, 1815, 1816, 1819, 1829
0, 1853, 1857, 1861, 1862, 1871, 1873, 1874, 1874, 1875, 1876, 1877, 1879, 1880, 1881, 1883, 1886, 1887, 18
917, 1919, 1921, 1922, 1923, 1928, 1932, 1933, 1934, 1937, 1942, 1943, 1944, 1944, 1945, 1945, 1947, 1947,
 1968, 1974, 1974, 1975, 1976, 1976, 1976, 1977, 1979, 1980, 1981, 1983, 1984, 1985, 1985, 1986, 1989, 1993
Elemento buscado = 1997
---------- Busca Sequencial ----------
Elemento encontrado na indice 997
Quantidade de comparações realizadas: 998
Tempo de execução:
Tempo Real: 0.00001200 segundos
Tempo Sistema: 0.00001200 segundos
--------------------------------
Elemento buscado = 1997
---------- Busca binária ----------
Elemento encontrado na posição 998
Quantidade de comparações realizadas: 9
Tempo de execução:
Tempo Real: 0.00000800 segundos
Tempo Sistema: 0.00000800 segundos
--------------------------------
```

1-2) Em relacao ao algoritmo de busca binaria, quais alteracoes devem ser realizadas para que a busca seja feita em um vetor ordenado de forma decrescente? Aplique as mudancas nas versoes recursiva e iterativa do metodo e teste o seu codigo.

roteiro_12/1-2.c

```c
#include "tempo.h"

#include <stdio.h>
#include <stdlib.h>

void liberar_vetor(int **v) {
  free(*v);
  *v = NULL;
}

int *copiaVetor(int *v, int n) {
  int i;
  int *v2;
  v2 = (int *)malloc(n * sizeof(int));
  for (i = 0; i < n; i++)
    v2[i] = v[i];
  return v2;
}

void imprimeVetor(int *v, int n) {
  int i, prim = 1;
  printf("[");
  for (i = 0; i < n; i++)
    if (prim) {
      printf("%d", v[i]);
      prim = 0;
    } else
```

```c
      printf(", %d", v[i]);
  printf("]\n");
}

int buscaSequencial(int *v, int n, int elem, int *comp) {
  int i;
  for (i = 0; i < n; i++) {
    (*comp)++;
    if (v[i] == elem)
      return i; // Elemento encontrado
  }
  return -1; // Elemento encontrado
}

int it_buscaBinaria(int *v, int ini, int fim, int elem, int *comp) {
  int meio;
  while (ini <= fim) {
    meio = (ini + fim) / 2;
    (*comp)++;
    if (elem == v[meio])
      return meio;
    else if (elem < v[meio])
      ini = meio + 1;
    else
      fim = meio - 1;
  }
  return -1;
}

int rec_buscaBinaria(int *v, int ini, int fim, int elem, int *comp) {
  if (ini > fim)
    return -1;
  int meio = (ini + fim) / 2;
  (*comp)++;
  if (v[meio] == elem)
    return meio;
  else if (elem < v[meio])
    return rec_buscaBinaria(v, meio + 1, fim, elem, comp);
  else
    return rec_buscaBinaria(v, ini, meio - 1, elem, comp);
}

int compara_decrescente(const void *a, const void *b) {
  int x = *(int *)a;
  int y = *(int *)b;

  if (x < y)
    return 1;
  if (x > y)
    return -1;

  return 0;
}

enum algoritmos_para_teste {
  PESQUISA_SEQUENCIAL = 0,
  PESQUISA_BINARIA,
};

#define QTD_ALGORITMOS_TESTE 2

int main() {
  int n;

  printf("Digite o tamanho do vetor:");
  scanf("%d", &n);

  int *array_original = (int *)malloc(n * sizeof(int));

  for (int i = 0; i < n; i++) {
    scanf("%d", &array_original[i]);
  }

  qsort(array_original, n, sizeof(int), compara_decrescente);

  int value;
  printf("Digite um valor para buscar no vetor:\n");
  scanf("%d", &value);

  printf("Vetor final da busca:\n");
  imprimeVetor(array_original, n);
```

```
109
110   for (int i = 0; i < QTD_ALGORITMOS_TESTE; i++) {
111     int comp = 0, index;
112
113     Temporizador tempo_teste;
114     iniciarTemporizador(&tempo_teste);
115
116     printf("Elemento buscado = %d\n", value);
117
118     switch (i) {
119     case PESQUISA_SEQUENCIAL:
120
121       printf("---------- Busca Sequencial ----------\n");
122
123       index = buscaSequencial(array_original, n, value, &comp);
124
125       if (index != -1) {
126         printf("Elemento encontrado no indice %d\n", index);
127       } else {
128         printf("Elemento não encontrado\n");
129       }
130
131       break;
132     case PESQUISA_BINARIA:
133       printf("---------- Busca binária ----------\n");
134
135       index = it_buscaBinaria(array_original, 0, n - 1, value, &comp);
136
137       if (index != -1) {
138         printf("Elemento encontrado no indice %d\n", index);
139       } else {
140         printf("Elemento não encontrado\n");
141       }
142
143       break;
144     }
145
146     finalizarTemporizador(&tempo_teste);
147     printf("Quantidade de comparações realizadas: %d\n", comp);
148     imprimirTempos(&tempo_teste);
149     printf("--------------------------------\n");
150   }
151
152   liberar_vetor(&array_original);
153   return 0;
154
155   return 0;
156 }
```

Saída do terminal:

```
arthurdetomi at arthurdetomi-System-Product-Name in ~/Documents/UFSJ-Graduacao/UFSJ-2025_1/Lab_Prog_2/rot
∴ ./1-2.out < input/100-misturado.txt
Digite o tamanho do vetor:Digite um valor para buscar no vetor:
Vetor final da busca:
[200, 200, 197, 195, 193, 193, 192, 189, 185, 184, 180, 178, 178, 175, 173, 172, 167, 167, 167, 165, 164, 1
 127, 124, 123, 121, 117, 116, 115, 115, 113, 113, 106, 105, 100, 97, 96, 95, 94, 89, 88, 85, 84, 84, 83, 8
29, 29, 28, 27, 25, 25, 21, 18, 17, 14, 12, 12, 10, 9, 4, 4, 3, 2]
Elemento buscado = 82
---------- Busca Sequencial ----------
Elemento encontrado no indice 61
Quantidade de comparações realizadas: 62
Tempo de execução:
Tempo Real: 0.00000800 segundos
Tempo Sistema: 0.00000800 segundos
--------------------------------
Elemento buscado = 82
---------- Busca binária ----------
Elemento encontrado no indice 61
Quantidade de comparações realizadas: 3
Tempo de execução:
Tempo Real: 0.00000600 segundos
Tempo Sistema: 0.00000600 segundos
--------------------------------
```

1-3) Utilizando o codigo feito na aula prática para a ordenacao de um vetor de struct, adapte esse codigo para que a struct seja de um aluno com os campos....

roteiro_12/1-3.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
```

```c
#include <string.h>

typedef struct {
  char nome[50];
  int matricula;
} Aluno;

void liberar_vetor(Aluno **v) {
  free(*v);
  *v = NULL;
}

int *copiaVetor(int *v, int n) {
  int i;
  int *v2;
  v2 = (int *)malloc(n * sizeof(int));
  for (i = 0; i < n; i++)
    v2[i] = v[i];
  return v2;
}

void imprimeVetor(int *v, int n) {
  int i, prim = 1;
  printf("[");
  for (i = 0; i < n; i++)
    if (prim) {
      printf("%d", v[i]);
      prim = 0;
    } else
      printf(", %d", v[i]);
  printf("]\n");
}

int comp;

void troca(Aluno *a, Aluno *b) {
  Aluno aux = *a;
  *a = *b;
  *b = aux;
}

int particao(Aluno *v, int ini, int fim, int (*compara)(Aluno *, Aluno *)) {
  int i = ini, j = fim;
  Aluno pivo = v[(ini + fim) / 2];
  while (1) {
    while (compara(&(v[i]), &pivo) < 0) {
      i++;
    }
    while (compara(&(v[j]), &pivo) > 0) {
      j--;
    }

    if (i < j) {
      troca(&v[i], &v[j]);
      i++;
      j--;
    } else
      return j;
  }
}

void QuickSort(Aluno *v, int ini, int fim, int (*compara)(Aluno *, Aluno *)) {
  if (ini < fim) {
    int q = particao(v, ini, fim, compara);
    QuickSort(v, ini, q, compara);
    QuickSort(v, q + 1, fim, compara);
  }
}

int rec_buscaBinaria(Aluno *v, int ini, int fim, Aluno elem,
                     int (*compara)(Aluno *, Aluno *)) {
  if (ini > fim)
    return -1;
  int meio = (ini + fim) / 2;
  comp++;
  if (compara(&(v[meio]), &elem) == 0)
    return meio;
  else if (compara(&(v[meio]), &elem) > 0)
    return rec_buscaBinaria(v, ini, meio - 1, elem, compara);
  else
    return rec_buscaBinaria(v, meio + 1, fim, elem, compara);
```

```c
}

int comparaNome(Aluno *a, Aluno *b) { return strcmp(a->nome, b->nome); }

int comparaMatricula(Aluno *a, Aluno *b) { return a->matricula - b->matricula; }

void print_aluno(const Aluno *aluno) {
  printf("--------------------\n");
  printf("Aluno Info:\n");
  printf("Nome: %s\n", aluno->nome);
  printf("Matricula: %d\n", aluno->matricula);
  printf("--------------------\n");
}

int main() {
  int n;

  printf("Digite a quantidade de alunos:\n");
  scanf("%d", &n);

  Aluno *alunos = malloc(n * sizeof(Aluno));

  printf("Cadastro de alunos:\n");
  for (int i = 0; i < n; i++) {
    printf("Nome:\n");
    scanf("%s", alunos[i].nome);
    printf("Matricula:\n");
    scanf("%d", &alunos[i].matricula);
  }

  Aluno al_busca;
  printf("Digite um nome de um aluno para busca:\n");
  scanf("%s", al_busca.nome);

  printf("Nome de aluno buscado : %d\n", al_busca.matricula);

  QuickSort(alunos, 0, n - 1, comparaNome);

  printf("Lista de alunos cadastrados:\n");
  for (int i = 0; i < n; i++) {
    printf("{%s, %d}, ", alunos[i].nome, alunos[i].matricula);
  }
  printf("\n");

  int index = rec_buscaBinaria(alunos, 0, n - 1, al_busca, comparaNome);

  if (index != -1) {
    printf("----- Sucesso ! --------\n");
    printf("Aluno encontrado\n");
    print_aluno(&alunos[index]);
    printf("--------------------\n");
  } else {
    printf("----- Not Found --------\n");
    printf("Aluno não encontrado\n");
    printf("--------------------\n");
  }

  printf("Digite uma matricula para busca:\n");
  scanf("%d", &al_busca.matricula);

  QuickSort(alunos, 0, n - 1, comparaMatricula);

  printf("Matricula buscada : %d\n", al_busca.matricula);

  index = rec_buscaBinaria(alunos, 0, n - 1, al_busca, comparaMatricula);

  if (index != -1) {
    printf("----- Sucesso ! --------\n");
    printf("Aluno encontrado\n");
    print_aluno(&alunos[index]);
    printf("--------------------\n");

  } else {
    printf("----- Not Found --------\n");
    printf("Aluno não encontrado\n");
    printf("--------------------\n");
  }

  liberar_vetor(&alunos);

  return 0;
```

```
165 }
```

Saída do terminal:

```
arthurdetomi at arthurdetomi-System-Product-Name in ~/Documents/UFSJ-Graduacao/UFSJ-2025_1/Lab_Prog_2/roteiro_12 on mainxxx 25-06-28
.:: ./1-3.out < in
Digite a quantidade de alunos:
Cadastro de alunos:
Nome:
Matricula:
Nome:
Matricula:
Nome:
Matricula:
Nome:
Matricula:
Nome:
Matricula:
Nome:
Matricula:
Nome:
Matricula:
Nome:
Matricula:
Nome:
Matricula:
Nome:
Matricula:
Digite um nome de um aluno para busca:
Nome de aluno buscado : Lara
Lista de alunos cadastrados:
{Ana, 1002}, {Beatriz, 1004}, {Bruna, 1010}, {Carlos, 1003}, {Daniel, 1005}, {Felipe, 1007}, {Joao, 1001}, {Lara, 1008}, {Mariana, 100
----- Sucesso ! --------
Aluno encontrado
--------------------
Aluno Info:
Nome: Lara
Matricula: 1008
--------------------
--------------------
Digite uma matricula para busca:
Matricula buscada : 1003
----- Sucesso ! --------
Aluno encontrado
--------------------
Aluno Info:
Nome: Carlos
Matricula: 1003
--------------------
--------------------
```

2-1) Reimplemente o TAD: Tabela Hash

roteiro_12/Hash.h

```c
#ifndef HASH_H
#define HASH_H

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

enum opcoes_func_sondagem { LINEAR, QUADRATICA, DUPLO };

#define QTD_SONDAGENS 3

typedef struct {
  int **tabela;
  int tam, qtd;
} Hash;

Hash *criaHash(int t) {
  Hash *h;
  h = (Hash *)malloc(sizeof(Hash));
  if (h != NULL) {
    h->tam = t;
    h->qtd = 0;
    h->tabela = (int **)malloc(t * sizeof(int *));
    if (h->tabela == NULL)
      return NULL;
    int i;
    for (i = 0; i < t; i++)
      h->tabela[i] = NULL;
  }
  return h;
}

```

```c
void destroiHash(Hash *h) {
  if (h != NULL) {
    int i;
    for (i = 0; i < h->tam; i++)
      if (h->tabela[i] != NULL)
        free(h->tabela[i]);
    free(h->tabela);
    free(h);
  }
}

int chaveDivisao(int chave, int tam) { return (chave & 0x7FFFFFFF) % tam; }

int chaveMultiplicacao(int chave, int tam) {
  float A = 0.6180339887; // constante: 0 < A < 1
  float val = chave * A;
  val = val - (int)val;
  return (int)(tam * val);
}

int chaveDobra(int chave, int tam) {
  int pos, n_bits = 30;

  int p = 1;
  int r = p << n_bits;
  while ((chave & r) != r) {
    n_bits--;
    r = p << n_bits;
  }

  n_bits++;
  pos = chave;
  while (pos > tam) {
    int metade_bits = n_bits / 2;
    int parte1 = pos >> metade_bits;
    parte1 = parte1 << metade_bits;
    int parte2 = pos ^ parte1;
    parte1 = pos >> metade_bits;
    pos = parte1 ^ parte2;
    n_bits = n_bits / 2;
  }
  return pos;
}

int valorString(char *str) {
  int i, valor = 1;
  int tam = strlen(str);
  for (i = 0; i < tam; i++)
    valor = 31 * valor + (i + 1) * ((int)str[i]);
  return valor;
}

int insereHash_semTratar(Hash *h, int elem) {
  if (h == NULL)
    return 0;
  int pos = chaveDivisao(elem, h->tam);

  if (h->tabela[pos] == NULL) {
    int *novo = (int *)malloc(sizeof(int));
    if (novo == NULL)
      return 0;
    *novo = elem;
    h->tabela[pos] = novo;
    h->qtd++;
  } else
    *(h->tabela[pos]) = elem;
  return 1;
}

int buscaHash_semTratar(Hash *h, int elem, int *p) {
  if (h == NULL)
    return 0;
  int pos = chaveDivisao(elem, h->tam);
  if (h->tabela[pos] == NULL)
    return 0;
  if (*(h->tabela[pos]) == elem) {
    *p = *(h->tabela[pos]);
    return 1;
  }
  return 0;
}
```

```c

int sondagemLinear(int pos, int i, int tam) {
  return ((pos + i) & 0x7FFFFFFF) % tam;
}

int sondagemQuadratica(int pos, int i, int tam) {
  pos = pos + 2 * i + 5 * i * i;
  return (pos & 0x7FFFFFFF) % tam;
}

int sondagemDuploHash(int H1, int chave, int i, int tam) {
  int H2 = chaveDivisao(chave, tam - 1) + 1;
  return ((H1 + i * H2) & 0x7FFFFFFF) % tam;
}

int insereHash_EnderAberto(Hash *h, int elem, int (*func_chave)(int, int),
                           enum opcoes_func_sondagem op) {
  if (h == NULL)
    return 0;
  int i, pos, newPos;
  pos = func_chave(elem, h->tam);
  for (i = 0; i < h->tam; i++) {
    switch (op) {
    case LINEAR:
      newPos = sondagemLinear(pos, i, h->tam);
      break;
    case QUADRATICA:
      newPos = sondagemQuadratica(pos, i, h->tam);
      break;
    case DUPLO:
      newPos = sondagemDuploHash(pos, elem, i, h->tam);
      break;
    }

    if (h->tabela[newPos] == NULL) {
      int *novo = (int *)malloc(sizeof(int));
      if (novo == NULL)
        return 0;
      *novo = elem;
      h->tabela[newPos] = novo;
      h->qtd++;
      return 1;
    }
  }
  return 0;
}

int buscaHash_EnderAberto(Hash *h, int elem, int *p) {
  if (h == NULL)
    return 0;
  int i, pos, newPos;
  pos = chaveDivisao(elem, h->tam);
  for (i = 0; i < h->tam; i++) {
    newPos = sondagemLinear(pos, i, h->tam);
    // newPos = sondagemQuadratica(pos, i, h->tam);
    // newPos = sondagemDuploHash(pos, elem, i, h->tam);
    if (h->tabela[newPos] == NULL)
      return 0;
    if (*(h->tabela[newPos]) == elem) {
      *p = *(h->tabela[newPos]);
      return 1;
    }
  }
  return 0;
}

void imprimeHash(Hash *h) {
  if (h == NULL)
    return;
  int i;
  for (i = 0; i < h->tam; i++) {
    printf("%d: ", i);
    if (h->tabela[i] == NULL)
      printf("NULL\n");
    else
      printf("%d\n", *(h->tabela[i]));
  }
}

#endif
```

roteiro_12/2-1.c

```c
#include "Hash.h"

#include <math.h>

void liberar_vetor(int **v) {
  free(*v);
  *v = NULL;
}

int *copiaVetor(int *v, int n) {
  int i;
  int *v2;
  v2 = (int *)malloc(n * sizeof(int));
  for (i = 0; i < n; i++)
    v2[i] = v[i];
  return v2;
}

void imprimeVetor(int *v, int n) {
  int i, prim = 1;
  printf("[");
  for (i = 0; i < n; i++)
    if (prim) {
      printf("%d", v[i]);
      prim = 0;
    } else
      printf(", %d", v[i]);
  printf("]\n");
}

enum opcoes_func_hash { DIVISAO = 0, MULT, DOBRA };

#define QTD_OPCOES 3

int main() {
  int n;

  printf("Digite o tamanho do vetor:");
  scanf("%d", &n);

  int *array = (int *)malloc(n * sizeof(int));

  for (int i = 0; i < n; i++) {
    scanf("%d", &array[i]);
  }

  printf("-------------------------------------\n");
  imprimeVetor(array, n);
  printf("-------------------------------------\n");

  int (*funcoes_chave[])(int, int) = {&chaveDivisao, &chaveMultiplicacao,
                                      &chaveDobra};
  char *funcoes_nome[] = {"Divisão", "Multiplicação", "Dobra"};

  for (int fn_chave_idx = 0; fn_chave_idx < QTD_OPCOES; fn_chave_idx++) {
    const int INC = 3;

    Hash *hash_table = criaHash(n + INC);

    for (int j = 0; j < n; j++) {
      insereHash_EnderAberto(hash_table, array[j], funcoes_chave[fn_chave_idx],
                             LINEAR);
    }

    printf("-------------------------------------\n");
    printf("Função hash aplicada : %s\n", funcoes_nome[fn_chave_idx]);
    imprimeHash(hash_table);
    printf("-------------------------------------\n");

    destroiHash(hash_table);
  }

  liberar_vetor(&array);
  return 0;
}
```

Saída do terminal:

```
arthurdetomi at arthurdetomi-System-Product-Name in ~/Document
.:: ./2-1.out < input/10-misturado.txt
Digite o tamanho do vetor:-----------------------------------
[47, 132, 88, 191, 5, 73, 64, 103, 39, 26]
-----------------------------------
-----------------------------------
Função hash aplicada : Divisão
0: 103
1: 39
2: 132
3: 26
4: NULL
5: 5
6: NULL
7: NULL
8: 47
9: 191
10: 88
11: 73
12: 64
-----------------------------------
-----------------------------------
Função hash aplicada : Multiplicação
0: 47
1: 191
2: 5
3: 73
4: 39
5: 88
6: 26
7: 132
8: 64
9: 103
10: NULL
11: NULL
12: NULL
-----------------------------------
-----------------------------------
Função hash aplicada : Dobra
0: 103
1: NULL
2: 47
3: 39
4: 191
5: 5
6: 26
7: NULL
8: 73
9: 64
10: NULL
11: 88
12: 132
-----------------------------------
```

2-2) Dessa vez, altere o TAD anterior...

roteiro_12/2-2.c

```c
#include "Hash.h"

#include <math.h>

void liberar_vetor(int **v) {
  free(*v);
  *v = NULL;
}

int *copiaVetor(int *v, int n) {
  int i;
  int *v2;
  v2 = (int *)malloc(n * sizeof(int));
  for (i = 0; i < n; i++)
    v2[i] = v[i];
  return v2;
}

void imprimeVetor(int *v, int n) {
  int i, prim = 1;
  printf("[");
  for (i = 0; i < n; i++)
    if (prim) {
      printf("%d", v[i]);
      prim = 0;
    } else
      printf(", %d", v[i]);
  printf("]\n");
}
```

```c

enum opcoes_func_hash { DIVISAO = 0, MULT, DOBRA };

int main() {
  int n;

  printf("Digite o tamanho do vetor:");
  scanf("%d", &n);

  int *array = (int *)malloc(n * sizeof(int));

  for (int i = 0; i < n; i++) {
    scanf("%d", &array[i]);
  }

  printf("-------------------------------------\n");
  imprimeVetor(array, n);
  printf("-------------------------------------\n");

  int (*funcoes_chave[])(int, int) = {&chaveDivisao, &chaveMultiplicacao,
                                      &chaveDobra};
  char *funcoes_chave_nome[] = {"Divisão", "Multiplicação", "Dobra"};
  char *funcoes_sondagem_nomes[] = {"Sondagem Linear", "Sondagem Quadratica",
                                    "Sondagem Duplo Hash"};

  for (int fn_sd_idx = 0; fn_sd_idx < QTD_SONDAGENS; fn_sd_idx++) {
    const int INC = 3;

    Hash *hash_table = criaHash(n + INC);

    for (int j = 0; j < n; j++) {
      insereHash_EnderAberto(hash_table, array[j], funcoes_chave[DIVISAO],
                             fn_sd_idx);
    }

    printf("-------------------------------------\n");
    printf("Função hash aplicada : %s\n", funcoes_chave_nome[DIVISAO]);
    printf("Função de sondagem aplicada: %s\n",
           funcoes_sondagem_nomes[fn_sd_idx]);
    imprimeHash(hash_table);
    printf("-------------------------------------\n");

    destroiHash(hash_table);
  }

  liberar_vetor(&array);
  return 0;
}
```

Saída do terminal:

```
└─.:: ./2-2.out < input/10-misturado.txt
Digite o tamanho do vetor:-----------------------------------
[47, 132, 88, 191, 5, 73, 64, 103, 39, 26]
-----------------------------------
-----------------------------------
Função hash aplicada : Divisão
Função de sondagem aplicada: Sondagem Linear
0: 103
1: 39
2: 132
3: 26
4: NULL
5: 5
6: NULL
7: NULL
8: 47
9: 191
10: 88
11: 73
12: 64
-----------------------------------
-----------------------------------
Função hash aplicada : Divisão
Função de sondagem aplicada: Sondagem Quadratica
0: 39
1: NULL
2: 132
3: NULL
4: NULL
5: 5
6: 73
7: 26
8: 47
9: 191
10: 88
11: 103
12: 64
-----------------------------------
-----------------------------------
Função hash aplicada : Divisão
Função de sondagem aplicada: Sondagem Duplo Hash
0: 39
1: NULL
2: 132
3: 26
4: 64
5: 5
6: NULL
7: 103
8: 47
9: 191
10: 88
11: NULL
12: 73
-----------------------------------
```

2-3) TAD: Tabela Hash com encadeamento...

roteiro_12/LSE.h

```c
#ifndef LISTASE_H
#define LISTASE_H

#include <stdio.h>
#include <stdlib.h>

typedef struct NO {
  int info;
  struct NO *prox;
} NO;

typedef struct NO *Lista;

Lista *criaLista() {
  Lista *li;
  li = (Lista *)malloc(sizeof(Lista));
  if (li != NULL) {
    *li = NULL;
  }
  return li;
}

int listaVazia(Lista *li) {
  if (li == NULL)
    return 1;
  if (*li == NULL)
    return 1; // True - Vazia!
```

```c
  return 0;   // False - tem elemento!
}

NO *alocarNO() { return (NO *)malloc(sizeof(NO)); }

void liberarNO(NO *q) { free(q); }

int listaBuscaElem(Lista *li, int elem, int *p) {
  if (li == NULL)
    return 0;
  NO *aux = *li;
  while (aux != NULL) {
    if (aux->info == elem) {
      *p = aux->info;
      return 1;
    }
    aux = aux->prox;
  }
  return 0;
}

int insereIni(Lista *li, int elem) {
  if (li == NULL)
    return 0;
  NO *novo = alocarNO();
  if (novo == NULL)
    return 0;
  novo->info = elem;
  novo->prox = *li;
  *li = novo;
  return 1;
}

int insereFim(Lista *li, int elem) {
  if (li == NULL)
    return 0;
  NO *novo = alocarNO();
  if (novo == NULL)
    return 0;
  novo->info = elem;
  novo->prox = NULL;
  if (listaVazia(li)) {
    *li = novo;
  } else {
    NO *aux = *li;
    while (aux->prox != NULL)
      aux = aux->prox;
    aux->prox = novo;
  }
  return 1;
}

int removeIni(Lista *li) {
  if (li == NULL)
    return 0;
  if (listaVazia(li))
    return 0;
  NO *aux = *li;
  *li = aux->prox;
  liberarNO(aux);
  return 1;
}

int removeFim(Lista *li) {
  if (li == NULL)
    return 0;
  if (listaVazia(li))
    return 0;
  NO *ant, *aux = *li;
  while (aux->prox != NULL) {
    ant = aux;
    aux = aux->prox;
  }
  if (aux == *li)
    *li = aux->prox;
  else
    ant->prox = aux->prox;
  liberarNO(aux);
  return 1;
}

```

```c
void imprimeLista(Lista *li) {
  if (li == NULL)
    return;
  if (listaVazia(li)) {
    printf("Lista Vazia!\n");
    return;
  }
  // printf("Elementos:\n");
  NO *aux = *li;
  while (aux != NULL) {
    printf("%d ", aux->info);
    aux = aux->prox;
  }
  printf("\n");
}

void destroiLista(Lista *li) {
  if (li != NULL) {
    NO *aux;
    while ((*li) != NULL) {
      aux = *li;
      *li = (*li)->prox;
      liberarNO(aux);
    }
    free(li);
  }
}

#endif

```

roteiro_12/HashLSE.h

```c
#ifndef HASH_H
#define HASH_H

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

typedef struct {
  Lista **tabela;
  int tam, qtd;
} Hash;

Hash *criaHash(int t) {
  Hash *h;
  h = (Hash *)malloc(sizeof(Hash));
  if (h != NULL) {
    h->tam = t;
    h->qtd = 0;
    h->tabela = (Lista **)malloc(t * sizeof(Lista *));
    if (h->tabela == NULL)
      return NULL;
    int i;
    for (i = 0; i < t; i++)
      h->tabela[i] = NULL;
  }
  return h;
}

void destroiHash(Hash *h) {
  if (h != NULL) {
    int i;
    for (i = 0; i < h->tam; i++)
      if (h->tabela[i] != NULL)
        destroiLista(h->tabela[i]);
    free(h->tabela);
    free(h);
  }
}

int chaveDivisao(int chave, int tam) { return (chave & 0x7FFFFFFF) % tam; }

int chaveMultiplicacao(int chave, int tam) {
  float A = 0.6180339887; // constante: 0 < A < 1
  float val = chave * A;
  val = val - (int)val;
  return (int)(tam * val);
}

```

```c
int chaveDobra(int chave, int tam) {
  int pos, n_bits = 30;

  int p = 1;
  int r = p << n_bits;
  while ((chave & r) != r) {
    n_bits--;
    r = p << n_bits;
  }

  n_bits++;
  pos = chave;
  while (pos > tam) {
    int metade_bits = n_bits / 2;
    int parte1 = pos >> metade_bits;
    parte1 = parte1 << metade_bits;
    int parte2 = pos ^ parte1;
    parte1 = pos >> metade_bits;
    pos = parte1 ^ parte2;
    n_bits = n_bits / 2;
  }
  return pos;
}

int valorString(char *str) {
  int i, valor = 1;
  int tam = strlen(str);
  for (i = 0; i < tam; i++)
    valor = 31 * valor + (i + 1) * ((int)str[i]);
  return valor;
}

int insereHashLSE(Hash *h, int elem) {
  if (h == NULL)
    return 0;
  int pos = chaveDivisao(elem, h->tam);
  if (h->tabela[pos] == NULL)
    h->tabela[pos] = criaLista();
  insereIni(h->tabela[pos], elem);
  h->qtd++;
  return 1;
}

int buscaHashLSE(Hash *h, int elem, int *p) {
  if (h == NULL)
    return 0;
  int pos = chaveDivisao(elem, h->tam);
  if (h->tabela[pos] == NULL)
    return 0;
  return listaBuscaElem(h->tabela[pos], elem, p);
}

void imprimeHash(Hash *h) {
  if (h == NULL)
    return;
  int i;
  for (i = 0; i < h->tam; i++) {
    printf("%d: ", i);
    if (h->tabela[i] == NULL)
      printf("NULL\n");
    else
      imprimeLista(h->tabela[i]);
  }
}

#endif
```

roteiro_12/2-3.c

```c
#include "HashLSE.h"

void liberar_vetor(int **v) {
  free(*v);
  *v = NULL;
}

int *copiaVetor(int *v, int n) {
  int i;
  int *v2;
  v2 = (int *)malloc(n * sizeof(int));
  for (i = 0; i < n; i++)
    v2[i] = v[i];
```

```c
  return v2;
}

void imprimeVetor(int *v, int n) {
  int i, prim = 1;
  printf("[");
  for (i = 0; i < n; i++)
    if (prim) {
      printf("%d", v[i]);
      prim = 0;
    } else
      printf(", %d", v[i]);
  printf("]\n");
}

int main() {
  int n;

  printf("Digite o tamanho do vetor:");
  scanf("%d", &n);

  int *array = (int *)malloc(n * sizeof(int));

  for (int i = 0; i < n; i++) {
    scanf("%d", &array[i]);
  }

  printf("-------------------------------------\n");
  imprimeVetor(array, n);
  printf("-------------------------------------\n");

  Hash *hash_table = criaHash(n);
  for (int i = 0; i < n; i++) {
    insereHashLSE(hash_table, array[i]);
  }

  printf("--------- Mostrando Tabela --------\n");
  imprimeHash(hash_table);
  printf("-------------------------------------\n");

  liberar_vetor(&array);
  return 0;
}
```

Saída do terminal:

```
arthurdetomi at arthurdetomi-System-Product-Name in ~/Documents
∴ ./2-3.out < input/10-misturado.txt
Digite o tamanho do vetor:-------------------------------------
[47, 132, 88, 191, 5, 73, 64, 103, 39, 26]
-------------------------------------
--------- Mostrando Tabela --------
0: NULL
1: 191
2: 132
3: 103 73
4: 64
5: 5
6: 26
7: 47
8: 88
9: 39
-------------------------------------
```