Universidade Federal de Santa Catarina
Campus Araranguá
Tecnologias da Informação e Comunicação
ARA7125 – Estrutura de Dados I
Prof. Gustavo Mello Machado

Lista de Exercícios III - Aula 01/07/2016

1) Considere a seguinte função hash:
h(key) = key mod 11,
onde *a* mod *b* retorna o resto da divisão de *a* por *b*. Armazene os valores dados na seguinte ordem:
82, 31, 28, 4, 45, 27, 59, 79 e 35
na tabela hash abaixo
```
int table[11];
```
Considere as seguintes soluções de tratamento de colisão
   a) Endereçamento encadeado
   b) Endereçamento aberto

2) Em uma tabela hash com 100 entradas, as colisões eram resolvidas usando listas encadeadas. Para reduzir tempo de pesquisa, decidiu-se que cada lista seria organizada como uma árvore binária de busca. A função utilizada é h(k) = k mod 100. Infelizmente, as chaves inseridas seguem o padrão $k_i = 50i$, onde k_i corresponde a i-ésima chave inserida. Mostre a situação da tabela após a inserção de k_i, com i = 1, 2, ..., 13 (desenhe).

3) Dada a implementação de tabela hash abaixo, onde M é pré-definido como sendo o tamanho da tabela. Observe que esta implementação não trata colisões.

```
typedef char *string;

typedef struct {
  int chave;
  string curso;
} tipoObjeto;

#define CHAVE_NULA            -1
#define CHAVE_MARCA           -2

tipoObjeto objetonulo;
objetonulo.chave = CHAVE_NULA;

tipoObjeto objetomarca;
objetomarca.chave = CHAVE_MARCA;

tipoObjeto tabela[M];

void inicializa (void) {
  int h;
  for (h = 0; h < M; h++) tabela[h] = objetonulo ;
}

void insere (tipoObjeto obj ) {
  int v = obj.chave ;
```

```
  int h = hash(v);
  tabela[h] = obj;
}

void remove (int v) {
  int h = hash(v);
  tabela[h] = objetonulo ;
}

tipoObjeto busca (int v) {
  int h = hash(v);
  return tabela[h];
}
```

Tendo em vista o código acima

a) Escreva uma função hash **`int`** `hash(`**`int`** `k)` apropriada. Qual método você utilizou para escrever a sua função hash?

b) Para o código acima, explique o que acontecerá nos casos em que ocorrerem colisões?

c) Para tratar colisões por endereçamento aberto as funções `insere` e `remove` poderiam ficar assim

```
bool insere (tipoObjeto obj ) {
  int v = obj.chave ;
  int i;
  for (i = 0; i < M; i++) {
    int h = hash(v,i);
    tipoObject o = tabela[h];
    if (o.chave == CHAVE_NULA || o.chave == CHAVE_MARCA) {
      tabela[h] = obj;
      return true;
    }
  }
  return false;
}

void remove (int v) {
  int i;
  for (i = 0; i < M; i++) {
    int h = hash(v,i);
    tipoObject o = tabela[h];
    if (o.chave == v) {
      tabela[h] = objetomarca;
      return;
    } else if (o.chave == CHAVE_NULA) {
      return;
    }
  }
}
```

Escreva a função correspondente para buscar um elemento `tipoObjeto busca (`**`int`** `v)` e a função hash **`int`** `hash(`**`int`** `k,` **`int`** `i)`. Qual escolha de estratégia para endereçamento aberto você escolheu para a sua função hash.

d) É possível reimplementar as funções `insere`, `remove` e `busca`, vistas na questão (3c) de forma recursiva? Se sim, apresente uma solução recursiva para estas funções e explique porque uma solução recursiva nestes casos não seria uma boa ideia?