Estrutura de Dados II

Lista de Exercicios de Tabelas Hash

Aluno: Nathan Martins

1. Enviado em código fonte.
2. Das técnicas de endereçamento aberto, o duplo hash é que proporciona o melhor espalhamento das no vetor, aplicando outra função de hash (que jamais deve ser igual a primeira) quando ocorre colisões, evitando clusters, por isso é considerada ligeiramente melhor que os outros métodos de endereçamento aberto. A exploração linear tende a formar grandes clusters, já que ela vai buscando células vazias na adjacências, quando há colisão, se o vetor é preenchido em mais de ⅔, o seu desempenho degrada significamente, aumentando o tempo de busca e inserção de elementos, nesse caso deve-se considerar transferir dados para um vetor maior. O encadeamento separado se vale de listas encadeadas, ele pega itens com a mesma chave e ao invés de procurar espaço vazio no vetor ele insere, na sua posição original dentro de uma lista encadeada. A inserção é rápida, já que não depende da exploração, onde quer que seja sua chave, ali o elemento será inserido. Usa mais memória por conta dos ponteiros, e o código é mais extenso, já que precisa ser implementado todo uma estrutura para listas encadeadas mas o acesso é rápido e a remoção é simples. È especialmente útil quando não se tem ideia de quanto elementos serão inseridos, já que cresce dinamicamente.
3.
    a. Esse problema ocorre quando temos clusters e removemos algum item da tabela, no momento da busca o algoritmo percorre o cluster tentando encontrar o elemento, porém quando ele encontra um espaço vazio ele para a busca, pois entende-se que o cluster acabou e o elemento não existe na lista, mas pode não ser o caso, um elemento pode ter sido removido mas a célula ainda faz parte do cluster, para resolvermos isso, nós não excluímos uma célula quando precisamos remover um elemento, nos a marcamos para o algoritmo saiba que ali tinha um elemento e deve continuar na sua busca.
    b.
```
for(i=0;i<MAXN. i++){
        if(chave%MAXN == hash[i].chave){
                printf(“Chave %d \n Nome %s \n Idade %d \n\n”, hash[i].chave, hash[i].nome, hash[i].idade);
        }
}
```

4.

Nº do Item	Busca	Valor do Hash	Tamanho do Passo	Celulas na Sequencia	Status da busca
1	38	4	2	6,8,10,12	Localizado
2	78	10	2	12,14,16,1	Não
3	24	7	1		Localizado
4	7	7	3	10,13	Não
5	51	0	4		Localizado
6	30	13	5		Não
7	19	2	1	3,4,5,6,7,8	Localizado
8	8	8	2	10,12,14,16,1	Não
9	32	15	3		Localizado
10	46	12	4	16,3,7,11	Localizado
11	69	1	1		Não
12	85	0	5	5,10,15,3	Localizado

5.

a.

Nº do Item	Chave	Valor do Hash	Tamanho do Passo	Celulas na Sequencia
1	5	5	5	
2	33	0	2	
3	19	8	1	
4	25	3	5	
5	27	5	3	8,0,3,6
6	41	8	4	1
7	15	4	5	
8	36	3	4	7

9	11	0	4	4,8,1,5,9

33	41		25	15	5	27	36	19	11	

b.

Nº do Item	Chave	Valor do Hash	Tamanho do Passo	Celulas na Sequencia
1	22	5	3	
2	61	10	4	
3	18	1	2	
4	75	7	5	
5	5	5	5	10, 15
6	1	1	4	5,9
7	32	15	3	1,4
8	5	5	5	10,15,3
9	12	12	3	
10	72	4	3	7,10,13
11	2	2	3	
12	27	10	3	13,16
13	41	7	4	11
14	67	16	3	2,5,8
15	25	8	5	13,1,6
16	Del(72)	4	3	7,10,13
17	30	13	5	
18	43	9	2	11,13,15,0
19	Del(25)	8	5	13,1,6
20	Del(1)	1	4	5,9
21	55	4	5	9

43	18	2	5	32	22	*	75	67	55	61	41	12	30		5	27