Semana 7: *Hashing* - Redimensionamento de Tabelas Hash e Funções de Sondagem

Prof. Dr. Juliano Henrique Foleis

Estude com atenção os vídeos e as leituras sugeridas abaixo. Os exercícios servem para ajudar na fixação do conteúdo e foram escolhidos para complementar o material básico apresentado nos vídeos e nas leituras. Quando o exercício pede que crie ou modifique algum algoritmo, sugiro que implemente-o em linguagem C para ver funcionando na prática.

Vídeos

Redimensionamento de Tabelas Hash com Endereçamento Aberto (Rehashing)

Funções de Sondagem (Sondagem Quadrática e Sondagem por Hash Duplo)

Leitura Sugerida

FEOFILOFF, Paulo. Estruturas de Dados. *Hashing* - **Seção:** Redimensionamento da tabela de sondagem linear (Link)

Exercícios

1. Clone (ou atualize!) o repositório da disciplina no github. A implementação da tabela hash com endereçamento aberto está nos arquivos *hashing/hashtable_ea.c* e *hashing/hashtable_ea.h*.

a. Implemente as funções em branco conforme mostrado no vídeo. (Já foi um exercício da semana passada.)

b. Implemente a função *static void THEA_Redimensionar(THEA* TH, int m_novo)* e as alterações nas funções THEA_Criar, THEA_Inserir e THEA_Remover, conforme apresentado no vídeo.

c. Altere a função THEA_Redimensionar para que faça o a diminuição da tabela (caso *m_novo* < *TH->m*) somente se for possível, ou seja, se *TH->n* <= *m_novo*.

d. Conforme discutido no vídeo, caso muitas remoções sejam realizadas na tabela, o fator de carga pode diminuir de tal maneira que haja desperdício excessivo de espaço. Para evitar o desperdício, insira uma verificação na função *THEA_Remover*, que invoque *THEA_Redimensionar* assim que $\alpha < 0.2$. Redimensione a tabela para possuir apenas a metade das posições que antes do redimensionamento.

OBS: Nem sempre esse redimensionamento para economizar espaço é recomendado, uma vez que a operação de redimensionamento pode prejudicar o desempenho do programa caso ela seja realizada com muita frequência.

2. No vídeo apresentei o redimensionamento da tabela hash que usa tratamento de colisões por endereçamento aberto. Tabelas hash que utilizam tratamento de colisões por encadeamento também podem ser redimensionadas utilizando um algoritmo de redimensionamento semelhante. A idéia criar um novo vetor de listas encadeadas de forma que M seja o próximo primo após $2M$ e inserir todos os elementos que estavam na tabela original na nova tabela.

Um detalhe importante é que agora α necessariamente é maior que 1, uma vez que as colisões são tratadas usando uma lista encadeada para cada posição da tabela hash. Sabemos que sob a hipótese de hashing uniforme o tamanho médio das listas é aproximadamente α. Logo, caso queiramos que sejam feitas em média k varreduras até encontrar a chave, podemos realizar o redimensionamento quando $\alpha > k$.

i. Altere as funções *THED_Criar*, *THED_Inserir* e *THED_Remover* e a estrutura *THED* (em *hashing/hashtable_ed.h* e *hashing/hashtable_ed.c*) para contabilizar a quantidade de elementos (n) na tabela hash.

ii. Implemente a função *static void THED_Redimensionar(THED* TH, int m_novo)* em *hashing/hashtable_ed.c* conforme descrito acima. **DICA:** Use a função THEA_Redimensionar(THEA* TH, int m_novo)* em *hashing/hashtable_ea.c* como referência.

iii. Altere a função *THED_Criar* para receber e gravar o parâmetro k na estrutura THED. O parâmetro k foi explicado no enunciado.

iv. Altere a função *THED_Inserir* para verificar a condição necessária para o redimensionamento, conforme explicado no enunciado. Caso seja, chame a função *THED_Redimensionar* para realizar a operação.

Atividade para Entregar

Não há atividade para entregar.

BONS ESTUDOS!