SCC0601– Introdução a Ciência da Computação II ICMC–USP

LISTA DE EXERCÍCIOS III
BUSCA, HASHING E PROGRAMAÇÃO DINÂMICA

Professor: Diego Raphael Amancio **PAE:** Marcos Vinícius C. Alves

Além das transparências utilizadas em aula, considere também a bibliografia da disciplina como suporte para resolução dos exercicios.

1. Implemente em C uma função que, supondo que haja recuperação recorrente de registros, realiza busca sequencial em um arranjo não ordenado de 10.000 elementos e utiliza o método da transposição. Neste método um registro recuperado com sucesso é trocado com o registro anterior. Faça o seguinte experimento com esse método:
   - insira 10.000 chaves aleatórias entre 1 e 100.000 no arranjo
   - marque o tempo para realizar a busca sequencial convencional de 50 chaves diferentes (ex. os números de 1 a 51), buscando cada chave 1000 vezes.
   - marque o tempo para realizar a busca sequencial com o método da transposição, das mesmas 50 chaves diferentes, buscando cada chave 1000 vezes.
2. Implemente em C uma função que realiza busca sequencial indexada. Para isso, utilize um arranjo original com o número de itens $n = 10.000$ e o número de índices $k = 50$. Faça o seguinte experimento com esse método:
   - insira 10.000 chaves aleatórias entre 1 e 100.000 no arranjo, e ordene o arranjo utilizando algum método de ordenação (excluindo o *bubblesort e variantes*).
   - marque o tempo para realizar a busca sequencial convencional no arranjo original, de 50 chaves diferentes, buscando cada chave 1.000 vezes.
   - marque o tempo para realizar a busca indexada, das mesmas 50 chaves, buscando cada chave 1.000 vezes.
3. Implemente em C uma função que realiza busca binária e uma função que realiza busca interpolada. Faça o seguinte experimento com tais métodos:
   - insira 100 chaves aleatórias entre 1 e 1.000 no arranjo, e ordene o arranjo utilizando algum método de ordenação (excluindo o *bubblesort e variantes*).
   - faça a busca binária e por interpolação e imprima quantas iterações foram necessárias para encontrar a chave em cada um dos métodos (o número de iterações é o número de vezes que a posição *meio* é (re)calculada.
4. Escrever o código de uma função C que faz uma busca sequencial por uma chave em um vetor de n elementos, **com sentinela.**
5. No exercício anterior, qual a vantagem do uso do sentinela em relação à busca sem sentinela?
6. Escrever o código de uma função C que faz busca sequencial indexada por um elemento em uma tabela com índice primário, e insere-o na tabela caso não seja encontrado (o índice precisa ser atualizado).

7. Escrever o código de uma função C que faz busca sequencial indexada por um elemento em uma tabela com índice primário, e remove-o da tabela caso seja encontrado (o índice precisa ser atualizado).

8. Escrever o código da versão recursiva da busca binária em vetor.

9. Defina uma função *hash* qualquer (suponha que você está lidando com números inteiros positivos lidos do usuário, até que -1 seja dado como entrada). Utilize *hashing* estático aberto (com listas encadeadas). Implemente o código da função de inserção. A cada inserção, imprima a posição da tabela em que o elemento foi inserido. Ao fim da execução, imprima todos elementos da tabela *hash* e libere toda a memória utilizada.

10. Considere as técnicas de pesquisa sequencial, pesquisa binária e a pesquisa baseada em *hashing*.
    a) Descreva as vantagens e desvantagens de cada uma das técnicas acima, colocando em que situações você usaria cada uma delas.
    b) Dê a ordem do pior caso e do caso esperado de tempo de execução para cada método.
    c) Qual é a eficiência de utilização de memória (relação entre o espaço necessário para dados e o espaço total necessário) para cada método?

11. Quais as características de uma boa função *hash*?

12. Um dos métodos utilizado para se organizar dados é pelo uso de tabelas *hash*.
    a) Em que situações a tabela *hash* deve ser utilizada?
    b) Descrev a dois mecanismos diferentes para resolver o problema de colisões de várias chaves em uma mesma posição da tabela. Quais são as vantagens e desvantagens de cada mecanismo?

13. Demonstre a inserção das chaves $5, 28, 19, 20, 33, 12, 17, 10$ em uma tabela *hash*, em que as colisões são resolvidas por encadeamento. Considere que a tabela tem tamanho 9, e a função *hash* $h(k) = k \mod 9$.

14. Substitua X X X X X X X X X X X X pelas 12 primeiras letras do seu nome, desprezando branco e letras repetidas, nas duas partes desta questão. Para quem não tiver doze letras diferentes no nome, completar com as letras PQRSTUWXYZ, nesta ordem, sem repetir letras já usadas, até completar 12 letras.
    a) Desenhe o conteúdo da tabela *hash* resultante da inserção de registros com as chaves X X X X X X X X X X X X , nesta ordem, em uma tabela inicialmente vazia de tamanho 7 (sete), usando listas encadeadas (hashing estático aberto). Use a função *hash* $h(k) = k \mod 7$ para a k-ésima letra do alfabeto.
    b) Desenhe o conteúdo da tabela *hash* resultante da inserção de registros com as chaves X X X X X X X X X X X X , nesta ordem, em uma tabela inicialmente vazia de tamanho 13 (treze), usando *overflow* progressivo e sondagem linear (*hashing* estático fechado) para resolver as colisões. Use a função *hash* $h(k) = k \mod 13$ para a k-ésima letra do alfabeto.

15. *Hashing* Estático Fechado:

    a) *Overflow* progressivo com sondagem linear. Desenhe o conteúdo da tabela *hash* resultante da inserção de registros com as chaves QUESTAOFCIL, nesta ordem, em uma tabela inicialmente vazia de tamanho 13 (treze) usando *overflow* progressivo com sondagem linear para a escolha de localizações alternativas. Use a função *hash* $h(k) = k \mod 13$ para a k-ésima letra do alfabeto.

    b) *Hashing* Duplo. Desenhe o conteúdo da tabela *hash* resultante da inserção de registros com as chaves QUESTAOFCIL, nesta ordem, em uma tabela inicialmente vazia de tamanho 13 (treze) usando *hashing* duplo. Use a função *hash* $h_1 = k \mod 13$ para calcular o endereço primário e $j = 1 + (k \mod 11)$ para resolver colisões, ou seja, para a escolha de localizações alternativas. Logo $h_i(k) = (h_{i-1}(k) + j) \mod 13$, para $2 \leq i \leq M$, onde M é o tamanho da tabela.

16. Demonstre a inserção das chaves $5, 28, 19, 20, 33, 12, 17, 10$ em uma tabela *hash*, em que as colisões são resolvidas por encadeamento. A tabela tem tamanho 9, e a função *hash* $h(k) = k \mod 9$.

17. Suponha uma tabela *hash* de tamanho 10 com endereçamento aberto para armazenar chaves no intervalo $[1, 999]$. Insira as seguintes chaves nesta tabela: $371, 121, 173, 203, 11, 24$, nessa ordem, considerando diferentes métodos de resolução de colisões:

    a) Sondagem linear, função *hash*: $h(k) = k\%M + i$

    b) Sondagem quadrática, função *hash*: $h(k) = k\%M + i^2$

    c) Sondagem quadrática, função *hash*: $h(k) = k\%M + 2i + i^2$

    d) *Hash* duplo, função *hash*: $h_1(k) = k\%M$, função *hash2*: $h_1(k) = 7 - (k\%7)$

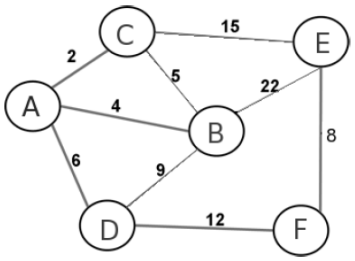
18. O uso de tabelas *hash* com encadeamento facilita a operação de remoção, pois apenas é preciso remover o elemento da lista ligada. No caso de endereçamento aberto a remoção deve ser feita de maneira diferente.

    a) qual operação é realizada sobre a chave removida?

    b) em caso de existirem muitos elementos marcados como deletados na tabela, perde-se a eficiência nas buscas; como é possível tratar esse problema?

19. Descreva as características de problemas que podem ser resolvidos com Programação Dinâmica (PD). Explique sucintamente as ideias básicas da PD. Enuncie os passos para o desenvolvimento de um algoritmo de PD, explicando cada um deles.

20. Suponha que, no gráfico abaixo, as letras representam cidades e os caminhos entre as cidades estão marcados com a distância entre as cidades conectadas. Imagine um problema de encontrar o menor caminho entre duas cidades.



O paradigma da programação dinâmica resolve sub-problemas menores e os armazena para posterior reuso, enquanto o paradigma de algoritmos gulosos escolhe sempre a solução que parece mais promissora num dado instante. Como se comportaria e qual seria o provável resultado se os seguintes algoritmos tivessem que obter o menor caminho entre a cidade **B** e a cidade **F**?

a) Programação dinâmica (caminhe pelas estradas armazenando possíveis soluções e descarte sub-soluções quando encontrar melhores opções).

b) Algoritmo guloso utilizando a heurística do vizinho mais próximo (ou seja, caminhe sempre pela próxima estrada cuja distância seja mínima, em direção à cidade destino).

21. Um problema muito famoso é o do escalonamento de grade horária, que consiste em, dadas as seguintes variáveis:

    - janelas de tempo
    - salas
    - disciplinas (das quais participam alunos e professores)

    E então, montar um horário viável, ou seja, no qual as salas sejam ocupadas pelas disciplinas em diferentes janelas de tempo de forma que tanto alunos quanto professores participar de todas as disciplinas que estejam inscritos. Discuta esse problema do ponto de vista dos seguintes paradigmas, descrevendo as vantagens e desvantagens de cada abordagem:

    - Tentativa e erro (*backtracking*)
    - Programação dinâmica

Este material consiste de uma compilação de exercícios retirados das referências citadas abaixo. Agradeço aos autores pela disponibilidade do material.

Referências

[1] J. a. Luís, *Listas de Exercícios – SCC201.* ICMC–USP, Agosto 2009.

[2] A. R. da Cruz, *Listas de Exercícios – SIN213.* UFV–CRP, Setembro 2011.

[3] R. Campello, *Listas de Exercícios – SCC5763.* ICMC–USP, Abril 2014.

[4] M. Ponti, *Listas de Exercícios – SCC201/501.* ICMC–USP, Novembro 2010.