ACH2024 - Algoritmos e Estruturas de Dados II

Atenção: Embora na lista abaixo haja predominância de exercícios de hashing e de processamento de textos, essa predominância não necessariamente se refletirá no conteúdo da prova.

11.4-1

Consider inserting the keys $10, 22, 31, 4, 15, 28, 17, 88, 59$ into a hash table of length $m = 11$ using open addressing with the auxiliary hash function $h'(k) = k \bmod m$. Illustrate the result of inserting these keys using linear probing, using quadratic probing with $c_1 = 1$ and $c_2 = 3$, and using double hashing with $h_2(k) = 1 + (k \bmod (m - 1))$.

11.4-2

Write pseudocode for HASH-DELETE as outlined in the text, and modify HASH-INSERT to handle the special value DELETED.

1. Considere as técnicas de pesquisa seqüencial, pesquisa binária e a pesquisa baseada em *hashing*.

a) Descreva as vantagens e desvantagens de cada uma dessas técnicas, indicando 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?

2. Suponha uma lista ordenada

11. Um dos métodos utilizados para se organizar dados é pelo uso de tabelas *hash*.

a) Em que situações a tabela *hash* deve ser utilizada?

b) Descreva 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?

12. Em uma tabela *hash* com cem entradas, as **colisões** são resolvidas usando listas encadeadas. Para reduzir o tempo de pesquisa, decidiu-se que cada lista seria organizada como uma árvore binária de pesquisa. A função utilizada é $h(k) = k \bmod 100$. Infelizmente, as chaves inseridas seguem o padrão $k_i = 50i$, onde k_i corresponde à i-ésima chave inserida.

a) Mostre a situação da tabela após a inserção de k_i, com $i = 1, 2, \cdots, 13$. (Faça o desenho.)

b) Depois que mil chaves são inseridas de acordo com o padrão acima, inicia-se a inserção de chaves escolhidas de forma randômica (isto é, não seguem o padrão das chaves já inseridas). Assim, responda:

i) Qual é a ordem do pior caso (isto é, o maior número de comparações) para se inserir uma chave?

ii) Qual é o número esperado de comparações para se inserir uma chave? (Assuma que cada uma das cem entradas da tabela é igualmente provável de ser endereçada pela função h.)

13. *Hashing*:

Substitua XXXXXXXXXXXX pelas 12 primeiras letras do seu nome, desprezando brancos e letras repetidas, nas duas partes dessa questão. Para quem não tiver doze letras diferentes no nome, completar com as letras PQRSTUVWXYZ, nesta ordem, até completar 12 letras. Por exemplo, eu deveria escolher N I V O Z A P Q R S T U. A segunda letra I de NIVIO não entra porque ela já apareceu antes, e assim por diante (Árabe, 1992).

a) Desenhe o conteúdo da tabela *hash* resultante da inserção de registros com as chaves XXXXXXXXXXXX, nesta ordem, em uma tabela inicialmente vazia de tamanho 7 (sete), usando listas encadeadas. Use a função *hash* $h(k) = k \bmod 7$ para a k-ésima letra do alfabeto.

b) Desenhe o conteúdo da tabela *hash* resultante da inserção de registros com as chaves XXXXXXXXXXXX, nesta ordem, em uma tabela inicialmente vazia de tamanho 13 (treze), usando *endereçamento aberto* e *hashing linear* para resolver as **colisões**. Use a função *hash* $h(k) = k$ mod 13 para a k-ésima letra do alfabeto.

14. *Hashing* - Endereçamento aberto

a) *Hashing Linear.* Desenhe o conteúdo da tabela *hash* resultante da inserção de registros com as chaves Q U E S T A O F C I L, nesta ordem, em uma tabela inicialmente vazia de tamanho 13 (treze) usando endereçamento aberto com *hashing 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 Q U E S T A O F C I L, nesta ordem, em uma tabela inicialmente vazia de tamanho 13 (treze) usando endereçamento aberto com ***hashing duplo***. Use a função *hash* $h_1(k) = k$ **mod** 13 para calcular o endereço primário e $j = 1 + (k$ **mod** 11) para resolver as colisões, ou seja, para a escolha de localizações alternativas. Logo, $h_i(k) = (h_{i-1}(k) + j)$ **mod** 13, para $1 \leq i \leq M - 1$ (Sedgewick, 1988).

15. Considere as seguintes estruturas de dados: *heap*, árvore binária de pesquisa, vetor ordenado, tabela *hash* com solução para **colisões** usando endereçamento aberto, tabela *hash* com solução para colisões usando listas encadeadas.

Para cada um dos problemas a seguir, sugira a estrutura de dados mais apropriada dentre as listadas anteriormente, de forma a minimizar tempo esperado e espaço necessário. Indique o tempo esperado e o espaço necessário em cada escolha e por que a estrutura de dados escolhida é superior às outras.

a) inserir/retirar/encontrar um elemento dado;

b) inserir/retirar/encontrar o elemento de valor mais próximo ao solicitado;

c) coletar um conjunto de registros, processar o maior elemento, coletar mais registros, processar o maior elemento, e assim por diante;

d) mesma situação descrita no item anterior adicionada da operação extra de ajuntar ("*merge*") duas estruturas.

R-9.7 Mostre o resultado do Exercício R-9.5, assumindo que as colisões são tratadas por teste quadrático, até o ponto em que o método falha.

R-9.8 Qual o resultado do Exercício R-9.5, assumindo que as colisões são tratadas por *hashing* duplo usando uma função de hash secundária $h'(k) = 7 - (k \bmod 7)$?

R-9.9 Forneça uma descrição em pseudocódigo da inserção em uma tabela de hash que usa teste quadrático para resolver colisões, assumindo que se usa o truque de substituir elementos deletados com um objeto indicando "item desativado".

R-9.10 Forneça uma descrição de Java dos métodos values( ) e entries( ) que poderiam ser incluídas na implementação da tabela de hash apresentada nos Trechos de código 9.3 – 9.5

R-9.11 Explique como modificar a classe HashTableMap, apresentada nos Trechos de códigos 9.3 − 9.5, para implementar o TAD dicionário em vez do TAD mapa.

R-9.12 Mostre o resultado de fazer rehash na tabela de hash, mostrada na Figura 9.4, para uma tabela de tamanho 19, usando a nova função de hash $h(k) = 2k \bmod 19$.

R-9.13 Discuta porque uma tabela de hash não é adaptada para implementar um dicionário ordenado.

R-9.14 Qual é o pior tempo para inserir n elementos em uma tabela de hash inicialmente vazia, com colisões sendo resolvidas por encadeamento? Qual seria o melhor caso?

R-9.15 Desenhe a skip list resultante da execução da seguinte seqüência de operações sobre a skip list da Figura 9.12: remove(38), insert(48,x), insert(24,y), remove(55). Registre as jogadas de cara e coroa.

R-9.16 Apresente a descrição de um pseudocódigo da operação de remoção em uma skip list.

R-9.17 Qual é o tempo de execução esperado dos métodos para manutenção de um conjunto de máximos inserindo-se n pares tal que cada par tenha o menor custo e desempenho que um anterior a ele? O que estará contido em um dicionário ordenado ao final desta série de operações? Se um par tem o menor curso e maior desempenho qual será o anterior a ele?

R-9.18 Argumente por que os localizadores não são realmente necessários para um dicionário implementado com uma boa tabela de hash.

R-12.1 Quantos prefixos não-vazios da cadeia $P =$ `"aaabbaaa"` são também sufixos de P?

R-12.2 Desenhe uma figura ilustrando as comparações feitas pelo algoritmo de procura de padrões baseado em força bruta, para o caso em que o texto é `"aaabaadaabaaa"` e o padrão é `"aabaaa"`.

R-12.3 Repita o problema anterior para o algoritmo BM de procura de padrões, não contando as comparações feitas para calcular a função last(c).

R-12.4 Repita o problema anterior para o algoritmo KMP de procura de padrões, não contando as comparações feitas para calcular a função de falha.

R-12.5 Calcule uma tabela representando a função last usada no algoritmo BM para o padrão

`"the quick brown fox jumped over a lazy cat"`

assumindo o seguinte alfabeto (que começa com um espaço em branco):

$\Sigma = \{$,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z$\}$.

R-12.6 Assumindo que os caracteres no alfabeto Σ podem ser enumerados e podem indexar arranjos, forneça um método de tempo $O(m + |\Sigma|)$ para construir a função last a partir de um padrão P de comprimento m.

R-12.7 Calcule uma tabela representando a função de falha KMP para o padrão `"cgtacgttcgtac"`.

R-12.8 Desenhe um trie-padrão para a seguinte seqüência de cadeias de caracteres:

{`abab,baba,ccccc,bbaaaa,caa,bbaacc,cbcc,cbca`}.

R-12.9 Desenhe um trie comprimido para o conjunto de cadeias de caracteres do Exercício R-12.8.

R-12.10 Desenhe a representação compacta para o trie de sufixos para a cadeia

`"minimize minime"`.

R-12.11 Qual o mais longo prefixo da cadeia `"cgtacgttcgtacg"` que também é um sufixo desta cadeia?

R-12.12 Desenhe o arranjo das freqüências e a árvore de Huffman para a seguinte cadeia:

`"dogs do not spot hot pots or cats"`.

R-12.13 Mostre o arranjo L para a maior subseqüência comum para as duas cadeias

X = `"skullandbones"`
Y = `"lullabybabies"`.

Qual é uma maior subseqüência comum entre essas cadeias de caracteres?

Capítulo 12 Livro Goodrich & Tamassia (Xerox):

C-12.1 Dê um exemplo de um texto T de comprimento n e um padrão P de comprimento m que force o algoritmo de procura de padrões por força bruta a um tempo de execução $\Omega(mn)$.

C-12.2 Justifique por que o método KMPFailureFunction (Trecho de código 11.5) precisa de tempo $O(m)$ em um padrão de comprimento m.

C-12.3 Mostre como modificar o algoritmo de procura de padrões KMP de forma que ele ache **todas** as ocorrências de um padrão P que aparece como substring em T, ainda sendo executado em tempo $O(m+n)$. (Assegure-se de encontrar até as ocorrências que se sobrepõem.)

C-12.9 Forneça um algoritmo eficiente para deletar uma cadeia de caracteres de um trie-padrão e analise seu tempo de execução.

C-12.11 Descreva um algoritmo para construir a representação compacta de um trie de sufixos e analise seu tempo de execução.

Exercícios adicionais diversos

1. Simule passo a passo a execução do *multiway merge-sort* (3 vias) sobre o arquivo abaixo.

   63, 39, 45, 80, 88, 86, 14, 21, 29, 17, 48, 69, 83, 79, 27, 32, 28, 38, 43, 22, 30, 44, 19, 62, 54, 21, 41, 13, 17, 71, 42, 12

   assumindo que as séries iniciais são formadas com tamanho $k = 1$.

2. Considere o seguinte heap de 1000 bytes, onde os blocos em branco estão em uso, e os blocos rotulados estão ligados em uma lista de blocos disponíveis em ordem alfabética. Os números indicam o endereço inicial de cada bloco.

0	100	200	400	500	575	700	850	900	999
a		b		c	d		e	f	

   Suponha que as seguintes requisições são feitas:

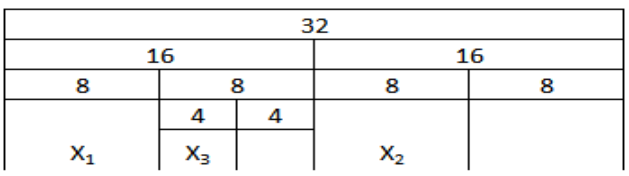
   (i) Alocar um bloco de 120 bytes; (ii) Retornar o bloco ref. aos bytes 700 - 849 e colocá-lo na frente da lista de blocos disponíveis. (iii) Alocar um bloco de 100 bytes;

   Apresente a configuração do heap após a execução dessas requisições (apresentando a lista de disponíveis em ordem alfabética), assumindo que os blocos livres são selecionados pelas estratégias:

   3a) First-fit

   3b) Best-fit

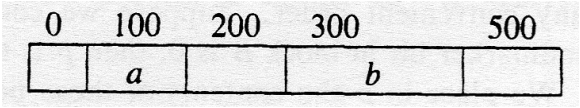
3. Considere um heap de tamanho 32, organizado como um sistema buddy exponencial, representado na figura abaixo, onde X_1, X_2, X_3 são variáveis alocadas.



Represente graficamente a configuração FINAL do heap após a realização das seguintes requisições (assumindo que, após cada liberação, os blocos amigos vazios são reagrupados):

(i) $X_4 \leftarrow$ alocação de 3 bytes; (ii) $X_5 \leftarrow$ alocação de 4 bytes; (iii) Liberação de X_3; (iv) Liberação de X_4. (v) Liberação de X_2.

4. Considere o heap abaixo no qual regiões em branco estão em uso e regiões rotuladas estão vazias.

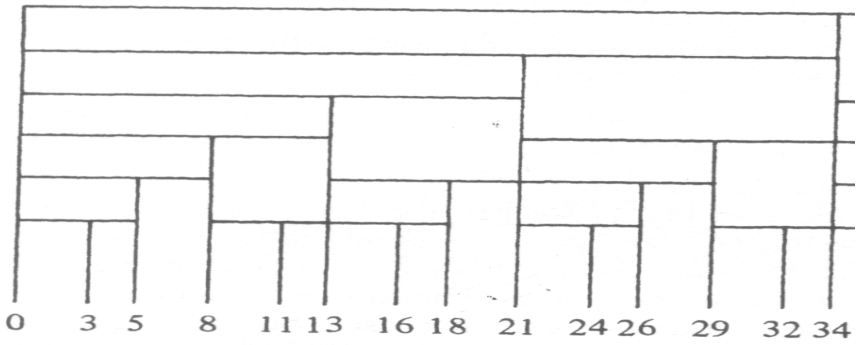


Apresente sequências de requisições que podem ser satisfeitas se usarmos

a) first fit;

b) best fit.

5. Considere um heap de tamanho 34, organizado como um sistema buddy de Fibonacci. (A figura abaixo apresenta um sistema desse tipo com todos os blocos alocados)



Assumindo que inicialmente o heap esteja vazio, represente graficamente a configuração FINAL do heap após a realização das seguintes requisições (assumindo que, após cada liberação, os blocos amigos vazios são reagrupados):

(i) $X_1 \leftarrow$ alocação de 7 bytes; (ii) $X_2 \leftarrow$ alocação de 7 bytes; (iii) $X_3 \leftarrow$ alocação de 3 bytes (iv) $X_4 \leftarrow$ alocação de 5 bytes; (v) $X_5 \leftarrow$ alocação de 5 bytes; (vi) Liberação de X_3; (vii) $X_6 \leftarrow$ alocação de 4 bytes;

6. Simule a execução do algoritmo KMP sobre o texto abaixo, indicando sequencialmente cada comparação realizada:

"MINIMIZARMIMETIZARMICAMIMICAMISA"

O padrão a ser buscado é a string "MIMICA".

Suponha que a função de falha já tenha sido computada.