MC202 — Estruturas de Dados
Lista de Exercícios 3
Tabelas de Espalhamento

Primeiro semestre de 2017 - Turmas B e C
Professor: Emilio Francesquini
francesquini@ic.unicamp.br

Legenda:
$\star$ – Fácil
$\star\star$ – Médio
$\star\star\star$ – Difícil

1. $\star$ [CLRS] Suponha que tenhamos um conjunto de elementos S representado por uma tabela de endereçamento direto T com m entradas. Descreva como encontrar o elemento máximo de S. Qual é o tempo que esta operação gastará no pior caso?

2. $\star\star$ [CLRS] Sugira como implementar uma tabela de endereçamento direto na qual as chaves dos elementos armazenados não precisam ser distintas e onde os elementos também podem possuir dados satélites. Todas as operações de dicionário (inserção, remoção e busca) devem ter um tempo de execução constante, ou seja, não devem depender do tamanho da tabela. (Não se esqueça que a operação de remoção recebe como parâmetro um ponteiro para o elemento a ser removido, e não a sua chave).

3. $\star$ Descreva com as suas próprias palavras o que é uma função de espalhamento (*hash function*). Quais características ela possui? Ela é uma função injetora, sobrejetora ou bijetora?

4. $\star$ Seja $\mathscr{H}$ uma função de hash. É possível criar uma função $\mathscr{G}$ tal que $\forall x\ \mathscr{G}(\mathscr{H}(x)) = x$? Justifique a sua resposta.

5. $\star\star$ Sistemas computacionais modernos utilizam uma função de hash para armazenar as senhas dos seus usuários. O mecanismo mais comum utiliza uma função de hash $\mathscr{H}$ criptograficamente segura[1] e a senha P do usuário da seguinte maneira:

   - Para cada usuário sorteia-se um conjunto de caracteres aleatórios S chamado de "sal"[2].
   - A senha P do usuário é concatenada (*salgada*[3]) com o sal S resultando no texto PS. Por exemplo, se a senha P do usuário *Juca* era "GostoDeComerJacaComAçucar" e o sal "UHWP*@$", então PS = "GostoDeComerJacaComAçucarUHWP*@$"

[1] Grosseiramente podemos dizer que uma função de hash é criptograficamente segura quando a complexidade de encontrar x e y tais que $\mathscr{H}(x) = \mathscr{H}(y)$ é equivalente a testar todas as possíveis entradas de $\mathscr{H}$.

[2] Veja `https://en.wikipedia.org/wiki/Salt_(cryptography)`

[3] Há quem defenda que o tempero só fica completo depois de adicionarmos sal e pimenta à senha do usuário. Veja: `https://en.wikipedia.org/wiki/Pepper_(cryptography)`

- Em um arquivo são armazenados o nome do usuário, $\mathscr{H}(PS)$ e o sal utilizado. Por exemplo:

Usuário	**Sal**	$\mathscr{H}(PS)$
Juca	UHWP*@$	98312798478374124294
Alemão	Pklc#1P	3123543824621721334
...	...	...

- Toda vez que o usuário for se logar, ele digita o seu nome de usuário e a sua senha P. O sistema utiliza a tabela acima para encontrar o sal S específico daquele usuário e salgar a senha informada (ou seja, concatena P e S). Em seguida $\mathscr{H}(PS)$ é calculado. Se o valor calculado corresponder ao valor armazenado na tabela o acesso é liberado e caso contrário o acesso é negado.

Tendo em vista o mecanismo descrito acima, responda:

(a) Por que não guardar a senha do usuário diretamente no arquivo?

(b) Por que quando esquecemos a senha, os sites mais seguros não nos mandam a senha por e-mail? Por que eles nos forçam a definir uma nova senha?

(c) Por que não guardar apenas o $\mathscr{H}(P)$ na tabela? Por que é importante salgar a senha?

(d) Qual é a importância de se utilizar um sal aleatório? E se utilizássemos como sal uma numeração sequencial?

(e) Por que o sal e a função de hash não precisam ser mantidas em segredo?

6. $\star$ [CLRS] Suponha que utilizamos a função de hash $\mathscr{H}$ para calcular o hash de n chaves distintas de elementos armazenados na tabela de hash T com m entradas e tratamento de colisões por encadeamento. Suponha também que a função $\mathscr{H}$ promove uma distribuição uniforme das chaves entre $[0, n)$. Qual é o número esperado de colisões em cada uma das entradas (*buckets*) da tabela de hash T?

7. $\star$ [CLRS] Mostre o que ocorre quando inserimos as chaves 5; 28; 19; 15; 20; 33; 12; 17; 10 em uma tabela de hash com tratamento de colisões por encadeamento. Considere que a tabela tem 9 entradas ($m = 9$) e que a função de hash $\mathscr{H}(k) = k \ mod \ 9$.

8. $\star$ [CLRS] Considere o problema de encontrar um elemento dentro de uma lista ligada com n elementos. Nesta lista cada elemento contém uma chave k além do valor do hash de k, ou seja, $\mathscr{H}(k)$. Considere também que cada uma das chaves é na verdade um texto muito longo. Descreva um mecanismo que aproveite os valores de hash já calculados para acelerar a busca por um elemento com chave k nesta lista.

9. $\star\star$ [CLRS] Considere uma versão do método da divisão na qual $\mathscr{H}(k) = k \ mod \ m$, $m = 2^p - 1$ e a chave k é uma cadeia de caracteres (string) interpretada como um número na base 2^p. Mostre que se pudermos derivar a string x a partir da string y através de uma permutação de seus caracteres, então $\mathscr{H}(x) = \mathscr{H}(y)$. Dê um exemplo de uma aplicação onde este comportamento da função de hash seria inconveniente.

10. ⋆ [CLRS] Considere uma tabela de hash de tamanho $m = 1000$, que utiliza uma função de hash $\mathscr{H}(k) = \lfloor m\ (kA\ (mod\ 1) \rfloor$, onde $A = \frac{\sqrt{5}-1}{2}$. Calcule as localizações da tabela de hash nas quais as chaves 61, 62, 63, 64, e 65 serão mapeadas.

11. ⋆ [CLRS] Considere a inserção das chaves 10; 22; 31; 4; 15; 28; 17; 88; 59 em uma tabela de hash com endereçamento aberto que utiliza a função de hash com $\mathscr{H}(k) = k$ com $m = 11$. Examine os casos onde é utilizado o método de sondagem linear, o método de sondagem quadrática com $c_1 = 1$ e $c_2 = 3$, e usando espalhamento duplo onde $\mathscr{H}_1(k) = k$ e $\mathscr{H}_2(k) = 1 + (k\ mod\ (m-1))$

12. ⋆ Considere a remoção de elementos em uma tabela de hash com endereçamento aberto discutida no slide 51 das notas de aula. Escreva o pseudocódigo para remover um elemento da tabela. Escreva também o pseudocódigo para a inserção e busca de um elemento na tabela levando em consideração o seu código para a remoção.

13. ⋆ ⋆ ⋆ [CLRS] Considere uma tabela de hash com endereçamento aberto que utiliza uma função de hash que distribui os elementos da tabela de maneira uniforme. Calcule o limite superior para o número de sondagens necessárias para se determinar que um elemento não está presente na tabela e o limite superior para o número de sondagens necessárias quando o elemento está presente. Considere os casos de um fator de carga igual a $\frac{3}{4}$ e de um fator de carga $\frac{7}{8}$, ou seja, quando $n = \frac{3}{4}m$ e $n = \frac{7}{8}m$.

14. ⋆ Discuta as diferenças entre as seguintes tabelas:

    - Tabela de endereçamento direto
    - Tabela de de hash com tratamento de colisão por encadeamento
    - Tabela de de hash com tratamento de colisão por endereçamento aberto com sondagem linear
    - Tabela de de hash com tratamento de colisão por endereçamento aberto com sondagem linear quadrática
    - Tabela de de hash com tratamento de colisão por endereçamento aberto com espalhamento duplo