MAC5710 - Estruturas de Dados e suas Aplicações

Departamento de Ciência da Computação

Primeiro semestre de 2009

Lista 8

1. **(Exercício 11.2-2 do CLRS)** Demonstre a inserção das chaves 5, 28, 19, 15, 20, 33, 12, 17, 10 numa tabela de espalhamento com colisões resolvidas por encadeamento. Suponha que a tabela seja de tamanho 9 e que a função de espalhamento seja $h(k) = k \bmod 9$.

2. **(Exercício 11.2-5 do CLRS)** Mostre que se $|U| > nm$, existe um subconjunto de U de tamanho n que consiste de chaves que, todas, entram na mesma posição da tabela de espalhamento, de modo que o pior caso das operações é linear em n.

3. **(Exercício 11.1-4 do CLRS)** Desejamos implementar um dicionário usando endereçamento direto em um vetor enoooorme. No início, as entradas do vetor contem lixo e inicializar o vetor inteiro não é recomendável por causa de seu tamanho. Descreva um esquema para implementar um dicionário por endereçamento direto (isto é, a posição i do vetor marca se i está ou não no conjunto) num vetor enorme. Cada objeto lá guardado deve utilizar espaço $O(1)$; as operações `inserção`, `busca` e `remoção` devem consumir tempo $O(1)$ e a inicialização também deve consumir tempo $O(1)$. (*Dica:* use uma pilha cujo tamanho é o número de chaves armazenadas no dicionário para ajudar a determinar se uma entrada do vetor enorme é válida ou não.)

4. **(Exercício 11.3-1 do CLRS)** Suponha que desejamos percorrer uma lista ligada de comprimento n onde cada elemento contém uma chave k junto com um valor de hash $h(k)$. Cada chave é uma longa cadeia de caracteres. Como podemos tirar vantagem dos valores de hash quando fazemos uma busca por um elemento com uma dada chave?

5. **(Exercício 11.3-4 do CLRS)** Considere uma tabela de espalhamento de tamanho $m = 1000$ e a função de espalhamento $h(k) = \lfloor (m(kA \bmod 1) \rfloor$, onde $A = (\sqrt{5} - 1)/2$. Calcule a posição em que cada uma das chaves 61, 62, 63, 64 e 65 seria armazenada em tal tabela segundo essa função de espalhamento.

6. **(Problema 11-4 do CLRS)** Seja $\mathcal{H}$ uma coleção de funções de hash na qual cada h em $\mathcal{H}$ mapeia o universo U de chaves em $\{0, 1, \ldots, m-1\}$. Dizemos que $\mathcal{H}$ é *k-universal* se, para cada seqüência fixa de k chaves distintas $\langle x^{(1)}, \ldots, x^{(k)} \rangle$ e cada h escolhido aleatoriamente de $\mathcal{H}$, a seqüência $\langle h(x^{(1)}), \ldots, h(x^{(k)}) \rangle$ tem a mesma probabilidade de ser qualquer uma das m^k seqüências de comprimento k cujos elementos estão em $\{0, 1, \ldots, m-1\}$.

   (a) Mostre que se $\mathcal{H}$ é 2-universal então $\mathcal{H}$ é universal.

   (b) Seja U o conjunto de n-uplas de valores de $\mathbf{Z}_p$ e seja $B = \mathbf{Z}_p$, onde p é primo. Para cada n-upla $a = \langle a_0, \ldots, a_{n-1} \rangle$ de valores de $\mathbf{Z}_p$ e para cada b em $\mathbf{Z}_p$, defina a função $h_{a,b} : \mathcal{H} \to B$ sobre a n-upla $x = \langle x_1, \ldots, x_k \rangle$ por
   $$h_{a,b}(x) = (\textstyle\sum_{j=0}^{n-1} a_j x_j + b) \bmod p.$$
   Seja $\mathcal{H}$ a família $\{h_{a,b}\}$. Mostre que a coleção $\mathcal{H}_{a,b}$ é 2-universal.

   (c) Suponha que Alice e Bob concordam secretamente sobre uma função de hash $h_{a,b}$ de uma família 2-universal $\mathcal{H}$ de funções de hash. Mais tarde, Alice envia pela internet uma mensagem m a Bob na qual $m \in U$. Ela autentica a mensagem para Bob enviando também $t = h_{a,b}(m)$ e Bob verifica se o par (m, t) que ele recebe satisfaz $t = h_{a,b}(m)$. Suponha que um adversário intercepte (m, t) em trânsito e tente iludir Bob substituindo o par (m, t) por um par (m', t') diferente. Mostre que a probabilidade de o adversário ter sucesso na tentativa de fazer Bob aceitar (m', t') é no máximo $1/p$, independente de quanta capacidade de computação o adversário tenha.

7. Projete uma implementação de um tipo abstrato de dados que suporte as seguintes operações:

   - INSERE(x): a inserção deve ser efetuada mesmo que x já esteja na estrutura de dados. Em outras palavras, a ED deve permitir duplicatas de um elemento.
   - REMOVE(y): remove um elemento *qualquer* da ED e o devolve em y. Novamente, a uma chamada, qualquer elemento do conjunto pode ser devolvido em y. Se existirem várias cópias do elemento removido na ED, apenas uma delas deve ser removida por vez.

   Esse tipo abstrato de dados é chamado de *pool*. Ele é útil para, por exemplo, armazenar tarefas. Novas tarefas são geradas e inseridas no *pool* de tarefas, e quanto um trabalhador fica disponível para começar uma nova tarefa, ele pega uma tarefa qualquer do *pool* de tarefas disponíveis. A sua implementação deve ser tal que as duas operações consumam tempo $O(1)$.

8. Construa uma seqüência infinita de cadeias de caracteres sobre um alfabeto finito, onde o tamanho total dos rótulos nas arestas da árvore de sufixos para essas cadeias cresce mais rapidamente que $O(m)$, onde m denota o comprimento da cadeia em questão. Ou seja, mostre uma família de cadeias que comprove que seria impossível ter uma implementação linear da construção da árvore de sufixos se os rótulos das arestas fossem armazenados explicitamente.

9. Dadas duas cadeias de caracteres, $S[1\,..\,m]$ e $T[1\,..\,n]$, determinar uma cadeia de caracteres $X[1\,..\,t]$ de comprimento máximo que é subcadeia tanto de S quanto de T. Descreva um algoritmo que resolva esse problema em tempo $O(m+n)$.

10. Dada uma cadeia de caracteres $S[1\,..\,m]$ que representa uma cadeia circular, determine a linearização de S que é mínima lexicograficamente. Descreva um algoritmo que resolva esse problema em tempo $O(m)$.

11. A relação entre as árvores de sufixos para uma cadeia e a cadeia reversa não é óbvia. No entanto, há uma relação significante entre essas duas árvores. Tente descobrir tal relação.

12. Suponha que precisamos manter dinamicamente a árvore de sufixos de uma cadeia de caracteres que está crescendo ou contraindo. Discuta como fazer isso eficientemente se a cadeia está crescendo (contraindo) na sua extremidade esquerda, e como fazer isso se a cadeia está crescendo (contraindo) na extremidade direita.

13. Na mesma situação do exercício anterior, considere agora que as alterações na cadeia são no seu interior (não nas extremidades). Você consegue pensar em uma maneira eficiente de lidar com este caso? Se não consegue, discorra sobre as dificuldades que encontrou.

14. Considere uma árvore de sufixos generalizada, construída para um conjunto de k cadeias de caracteres. Cadeias adicionais podem ser adicionadas ou removidas do conjunto. Esse é um problema que ocorre de fato na manutenção de dados de seqüências biológicas. Discuta o problema de manter uma árvore de sufixos generalizada para um tal conjunto dinâmico de cadeias de caracteres.

15. Para cada par de cadeias de caracteres, podemos calcular o comprimento da subcadeia mais longa desse par em tempo linear no comprimento total das duas cadeias, usando uma árvore de sufixos. Agora imagine que temos k cadeias, de comprimento total n, e que queremos calcular o menor comprimento de uma subcadeia de comprimento máximo entre duas destas cadeias. Ou seja, para cada par, temos o comprimento máximo de um prefixo comum, e queremos determinar o par para qual tal comprimento é o menor possível. Há um jeito óbvio de resolver esse problema em tempo $O(k^2 n)$ olhando para cada um dos $k(k-1)/2$ pares e juntando a informação para todos eles. Mostre como resolver esse problema em tempo $O(k(n+k))$, usando uma árvore de sufixo.

16. Para cada par de cadeias de caracteres, podemos calcular o comprimento do prefixo mais longo entre esse par em tempo linear no comprimento total das duas cadeias, usando uma árvore de sufixos. Agora imagine que temos k cadeias, de comprimento total n, e que queremos calcular o menor comprimento de um prefixo máximo entre duas destas cadeias. Ou seja, para cada par, temos o comprimento máximo de um prefixo comum, e queremos determinar o par para qual tal comprimento é o menor possível. Há um jeito óbvio de resolver esse problema em tempo $O(k^2 + kn)$ olhando para cada um dos $k(k-1)/2$ pares e juntando a informação para todos eles. Mostre como resolver esse problema em tempo $O(n)$ (independente do valor de k). Considere agora a variante desse problema em que queremos determinar o maior comprimento de um prefixo máximo entre duas das cadeias dadas. O que você pode dizer sobre essa variante?