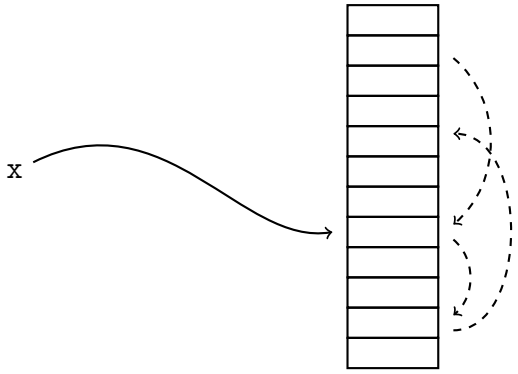
Estruturas de Dados

aula 27: A matemática das tabelas de dispersão

1 Introdução

Na aula passada, nós vimos que a busca sem sucesso na tabela de dispersão com 90% de ocupação executa tempo médio $O(1)$



Porque a busca fica ping-pongando na tabela até bater em uma posição vazia.

E isso acontece após 10 passos, em média — (*i.e.,* $O(1)$).

Legal.

Mas a busca com sucesso acaba antes.

Porque ela encontra o seu elemento antes de bater em uma posição vazia.

Mas qual é o tempo médio da busca com sucesso?

Bom, nós começamos observando que a inserção também precisa encontrar uma posição vazia para colocar o novo elemento.

Daí que, no momento em que a tabela está 90% ocupada, a inserção inspeciona 10 posições da tabela, em média — (*i.e.,* $O(1)$)

E esse também é o tempo da busca por esse elemento.

Porque a busca sempre realiza o mesmo caminho que foi percorrido no momento da inserção.

Mas, elementos que entram na tabela mais cedo encontram a sua posição mais rápido — (*porque a tabela tinha mais posições vazias*)

Logo, o seu tempo de busca vai ser menor.

E isso nos dá a ideia de que o tempo médio da busca com sucesso é menor do que o tempo médio da busca com sucesso.

Sim, mas como é que a gente calcula isso?

Bom, nós vamos calcular isso imaginando o seguinte experimento

- primeiro a gente insere m elementos em uma tabela de tamanho n

- daí, a gente faz a busca por cada um desses elementos
- e soma os tempos de busca de todos os elementos e divide por m,
  para obter o tempo médio da busca com sucesso

Mas o tempo da busca é igual o tempo da inserção.

Logo, a gente pode somar os tempos de inserção.

Certo.

Imagine que já foram inseridos k elementos na tabela, e nós estamos a ponto de inserir mais um.

Nesse momento, a probabilidade de encontrar uma posição vazia é $\frac{n-k}{n}$.

Logo, nesse momento, a inserção vai inspecionar

$$\frac{1}{(n-k)\,/\,n} \quad = \quad \frac{n}{n-k} \qquad \text{posições} \quad \text{(em média)}$$

Legal.

Agora, basta somar isso para k variando de 0 até $m-1$

$$\begin{aligned}
\sum_{k=0}^{m-1} \frac{n}{n-k} \quad &= \quad n \cdot \sum_{k=0}^{m-1} \frac{1}{n-k} \\
&= \quad n \cdot \underbrace{\Big(\frac{1}{n} \;+\; \frac{1}{n-1} \;+\; \frac{1}{n-2} \;+\; \ldots \;+\; \frac{1}{n-m+1} \Big)}_{(*)}
\end{aligned}$$

Daí, para estimar a soma entre parênteses, a gente faz a seguinte esperteza

$$\begin{aligned}
(*) \quad &= \quad \Big(\tfrac{1}{n} \;+\; \tfrac{1}{n-1} \;+\; \tfrac{1}{n-2} \;+\; \ldots \;+\; \tfrac{1}{2} \;+\; 1 \Big) \\
&\quad\; - \Big(\tfrac{1}{m-n} \;+\; \tfrac{1}{m-n-1} \;+\; \ldots \;+\; \tfrac{1}{2} \;+\; 1 \Big)
\end{aligned}$$

E daí, para estimar essas somas, a gente faz mais uma esperteza

$$\underset{\simeq 1}{\boxed{1}} + \underset{\simeq 1}{\boxed{\tfrac{1}{2} \;+\; \tfrac{1}{3}}} + \underset{\simeq 1}{\boxed{\tfrac{1}{4} \;+\; \tfrac{1}{5} \;+\; \tfrac{1}{6} \;+\; \tfrac{1}{7}}} + \underset{\simeq 1}{\boxed{\tfrac{1}{8} \;+\; \ldots \;+\; \tfrac{1}{15}}} + \ldots + \tfrac{1}{n}$$

$$= \quad O(\log n)$$

Quer dizer, a soma dos tempos de inserção é (aproximadamente)

$$n \cdot \Big(\log n \;-\; \log (n-m) \Big)$$

Quando $m = 9n/10$ (i.e., 90% de ocupação), isso dá

$$\begin{aligned} n \cdot \Big(\log n \ - \ \log (n/10) \Big) \quad &= \quad n \cdot \Big(\log n \ - \ \log n \ + \ \log 10 \Big) \\ &\simeq \quad 3,32 \cdot n \end{aligned}$$

E dividindo isso por n, nós descobrimos que

→ *A busca com sucesso inspeciona 3,32 posições em média*
— (na tabela com 90% de ocupação)

1.1 O pior caso

Certo.

Mas isso é apenas o tempo médio.

E isso não significa que todas as buscas vão levar esse tempo.

Quer dizer, a busca por alguns elementos vai ser mais rápida.

E a busca por outros elementos vai levar mais tempo.

Daí, faz sentido perguntar

- *Qual é o pior caso ?*

(. . .)

2 Análise de encadeamento externo

(. . .)

- caixas e bolas

(. . .)

3 Espertezas (quase perfeitas)

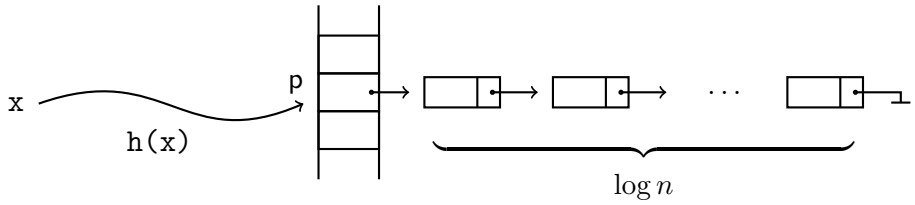
Certo.

Então o pior caso da busca é $O(\log n)$, tanto no esquema de endereçamento aberto como no esquema de encadeamento externo.

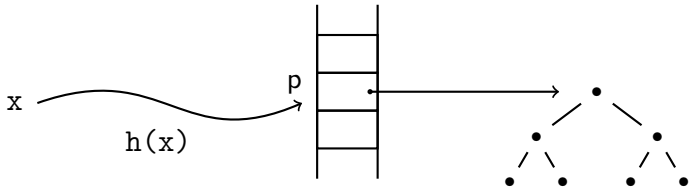
E a nossa tarefa agora é tentar melhorar isso.

Nós vamos concentrar a nossa atenção no esquema de encadeamento externo.

Nesse caso, o pior caso corresponde à maior lista encadeada externa



E aqui, nem é preciso pensar muito para imaginar que a lista encadeada pode ser substituída por uma árvore binária de busca (balanceada)



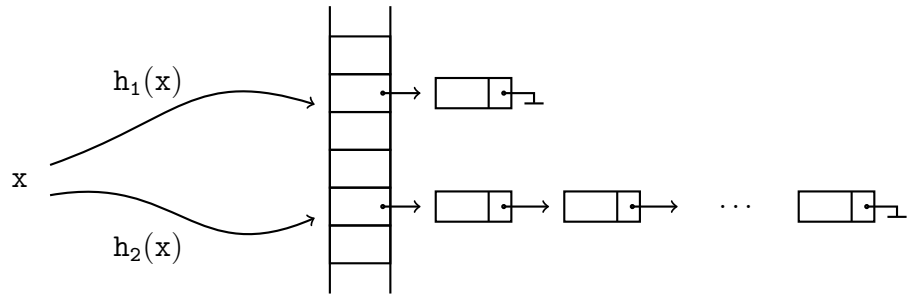
E apenas isso já é suficiente para reduzir o pior caso da busca para $O(\log \log n)$
— (*o que já é uma coisa bem pequenininha ...*)

Mas, essa não é a única maneira de fazer as coisas.

Quer dizer, outra ideia consiste em utilizar duas funções de dispersão

$$\mathsf{h}_1(\cdot) \qquad\qquad \mathsf{h}_2(\cdot)$$

Daí, no momento em que um novo elemento é inserido na tabela, nós utilizamos as duas funções



Quer dizer, a ideia aqui é que o novo elemento `x` é inserido na posição que tem a menor lista encadeada.

E a boa notícia é que esse esquema também reduz o pior caso da busca para $O(\log \log n)$.

Infelizmente, a análise desse esquema é complicada demais par apresentar aqui.

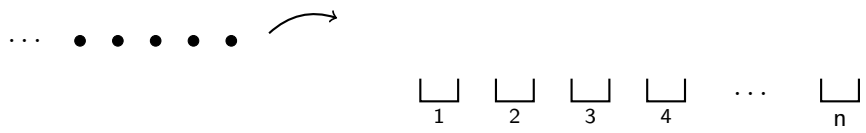
Mas, existe uma ideia ainda melhor do que essa.

4 A tabela de dispersão perfeita

Agora nós vamos reduzir o pior caso da busca para $O(1)$.

E para fazer isso, nós voltamos ao experimento das caixas e bolas.

Quer dizer, imagine que nós atiramos bolas aleatoriamente sobre n caixas



Daí, em algum momento vai acontecer de uma bola cair em uma caixa que já estava ocupada — (*i.e., uma colisão*)

A pergunta é

- *Qual é o momento em que isso ocorre pela primeira vez ?*