Busca binária

Como encontrar um dado CPF numa longa lista de números de CPF? Se a lista não estiver ordenada, não há o que fazer senão percorrer a lista toda. Se a lista estiver ordenada, entretanto, é possível fazer algo bem melhor. Esse problema é um caso particular do seguinte

> PROBLEMA DA BUSCA EM VETOR ORDENADO: Dado um inteiro x e um vetor inteiro crescente $A[1 \,..\, n]$, encontrar j tal que $A[j-1] < x \leq A[j]$.

O problema faz sentido para todo n natural, até mesmo para $n = 0$. Toda instância do problema tem solução e toda solução j pertence ao intervalo $1 \,..\, n+1$. Se $x > A[n]$, por exemplo, então $n+1$ é a resposta correta. (Veja o exemplo de *Busca em vetor ordenado* em outra página.)

Solução iterativa

O seguinte algoritmo iterativo é uma solução muito eficiente do problema da busca em vetor ordenado:

```
BUSCA-BINÁRIA-ITERATIVA (A, n, x)
1   p := 0
2   r := n+1
3   enquanto p < r−1
4       q := ⌊(p+r)/2⌋
5       se A[q] < x
6           p := q
7       senão r := q
8   devolva r
```

No início de cada repetição do “enquanto”, imediatamente antes da comparação de p com $r-1$, vale a relação

$$A[p] \;<\; x \;\leq\; A[r].$$

(Imagine que o vetor tem um elemento $A[0]$ com valor $-\infty$ e um elemento $A[n+1]$ com valor $+\infty$.) Como essa relação vale no início de cada iteração, dizemos que ela é invariante. (Note a semelhança entre esta relação e o objetivo que estamos perseguindo.) É óbvio que r é a resposta correta quando $p = r-1$.

Em cada iteração temos $p < q < r$ no fim da linha 4. Logo, tanto $r-q$ quanto $q-p$ são (estritamente) menores que $r-p$. Portanto, a sequência de valores da expressão

$r-p$ é estritamente decrescente. Assim, o número de iterações é finito.

Exercícios 1

1. Diga *o que* o algoritmo Busca-Binária faz.
2. Prove o invariante principal do algoritmo Busca-Binária-Iterativa.
3. Prove que $p < q < r$ em cada iteração do algoritmo Busca-Binária-Iterativa. Deduza daí que a execução do algoritmo termina depois de um número finito de iterações.

Desempenho da solução iterativa

O consumo de tempo de Busca-Binária-Iterativa é proporcional a número de repetições do bloco de linhas 4 a 7. Qual é esse número?

No início da primeira iteração, $r-p$ vale aproximadamente n. No início da segunda, vale aproximadamente $n/2$. No início da terceira, $n/4$. No início da $(k+1)$-ésima, $n/2^k$. Quando k atinge ou ultrapassa $\lg n$, o valor da expressão $n/2^k$ fica menor ou igual a 1 e a execução do algoritmo termina. Logo, o número de iterações é aproximadamente

$$\lg n .$$

(Isso é muito menos que as n iterações de uma busca sequencial.) Se cada iteração consome 1 unidade de tempo então uma busca em n elementos consome $\lg n$ unidades de tempo, uma busca em $8n$ elementos consumirá apenas $3 + \lg n$ unidades de tempo, e uma busca em $1024n$ elementos consumirá apenas $10 + \lg n$ unidades de tempo.

Exercícios 2

1. Prove que a comparação na linha 5 é executada pelo menos $\lfloor \lg n \rfloor$ vezes e no máximo $\lfloor \lg n \rfloor + 1$ vezes.

Solução recursiva do problema

A solução recursiva do problema de busca começa com um algoritmo-embrulho (= *wrapper function*), ou algoritmo-interface, que repassa o serviço para o algoritmo recursivo propriamente dito.

```
Busca-Binária (A, n, x)
    devolva BB-r (A, 0, n+1, x)
```

```
BB-r (A, p, r, x)
    se p = r−1
        devolva r e pare
    q := ⌊(p+r)/2⌋
    se A[q] < x
```

```
5        devolva BB-R (A, q, r, x)
6    senão devolva BB-R (A, p, q, x)
```

O sufixo "-R" é uma abreviatura de "recursivo".) O algoritmo BB-R recebe um vetor crescente $A[p+1\,..\,r-1]$ e um número x tal que $A[p] < x \le A[r]$ (portanto $p < r$) e devolve um índice j no intervalo $p+1\,..\,r$ tal que $A[j-1] < x \le A[j]$. (Imagine que $A[p] = -\infty$ e $A[r] = +\infty$.)

A prova de que o algoritmo de fato produz um tal j procede por indução em $r-p$. Se $r-p = 1$ então o algoritmo devolve r e esta é a resposta correta pois $p = r-1$ e $A[p] < x \le A[r]$. Suponha agora que $r-p > 1$. Nossa hipótese de indução é que o algoritmo produz a resposta correta quando invocado com argumentos (A, p', r', x) tais que $r'-p' < r-p$. Em particular, o algoritmo produz a resposta correta quando invocado com argumentos (A, q, r, x) e (A, p, q, x).

- Se $A[q] < x$ então o algoritmo devolve a solução da instância (A, q, r, x). Mas esta é também a solução da instância (A, p, r, x).
- Se $A[q] \ge x$ então o algoritmo devolve a solução da instância (A, p, q, x). Mas esta é também a solução da instância (A, p, r, x).

Desempenho da solução recursiva

O consumo de tempo do algoritmo BB-R é função do número de elementos, digamos n, do vetor $A[p+1\,..\,r-1]$. É claro que $n = r-p-1$. Digamos que o algoritmo consome $T(n)$ unidades de tempo no pior caso. Para calcular $T(n)$, considere o consumo de tempo de cada linha do algoritmo (supondo que cada linha "simples" consome uma unidade de tempo):

BB-R (A, p, r, x)	
1 se $p = r-1$	1
2 devolva r e pare	1
3 $q := \lfloor (p+r)/2 \rfloor$	1
4 se $A[q] < x$	1
5 devolva BB-R (A, q, r, x)	$T(\lfloor n/2 \rfloor)$
6 senão devolva BB-R (A, p, q, x)	$T(\lceil n/2 \rceil - 1)$

(Para entender o "$\lfloor n/2 \rfloor$" e o "$\lceil n/2 \rceil - 1$" nas linhas 5 e 6, veja o exercício abaixo.) Segue daí que o tempo de pior caso, $T(n)$, é definido pela recorrência

$$T(n) \;=\; \max\big(T(\lfloor n/2 \rfloor),\, T(\lceil n/2 \rceil - 1)\big) + 3$$

para $n = 1, 2, 3$, etc., com valor inicial $T(0) = 2$. (Puxa! que complicado!) A intuição sugere que $T(n)$ cresce com n. Suporemos então que

$$T(n) \;=\; T(\lfloor n/2 \rfloor) + 3 \qquad (*)$$

para $n = 1, 2, 3$, etc. Eis alguns valores:

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$T(n)$	2	5	8	8	11	11	11	11	14	14	14	14	14	14	14	14	17	17

Resta extrair daí uma fórmula fechada para $T(n)$.

Solução da recorrência. Felizmente não precisamos de uma fórmula fechada exata para a função $T(n)$ definida pela recorrência (*). Basta obter uma boa cota superior. O exemplo B da página *Recorrências* sugere que $T(n) = \mathrm{O}(\lg n)$. Poderíamos recorrer ao Teorema Mestre para uma confirmação, mas prefiro mostrar diretamente que

$$T(n) \le 8 \lg n$$

para $n = 2, 3, 4$, etc. (Não há mal em ignorar os casos em que n vale 0 ou 1.) A desigualdade é evidentemente verdadeira quando $n = 2$ e $n = 3$. Agora suponha que $n > 3$. Observe que $\lfloor n/2 \rfloor \ge 2$ e adote a hipótese de indução $T(\lfloor n/2 \rfloor) \le 8 \lg \lfloor n/2 \rfloor$. Então

$$\begin{aligned} T(n) &= 8 \lg \lfloor n/2 \rfloor \ + \ 3 \\ &\le 8 \lg (n/2) \ + \ 3 \\ &= 8 (\lg n - 1) \ + \ 3 \\ &= 8 \lg n - 8 \ + \ 3 \\ &< 8 \lg n \,. \end{aligned}$$

Viva! Isso prova o que queríamos.

Não é difícil provar, de maneira semelhante, que $T(n) \ge \lg n$ para $n = 1, 2, 3$, etc.

Exercícios 3

1. ★ Mostre que para qualquer número natural n tem-se $\lfloor n/2 \rfloor = \lceil (n-1)/2 \rceil$ e $\lceil n/2 \rceil - 1 = \lfloor (n-1)/2 \rfloor$.
2. Seja T a função definida pela recorrência (*). Prove que $T(n) \le 37 \lg n + 8$ para $n = 1, 2, 3$, etc. (Note que ignoramos o ponto $n = 0$.)
3. Seja T a função definida pela recorrência (*). Prove que $T(n) \ge \lg n$ para $n = 1, 2, 3$, etc. (Segue daí que $T(n) = \Omega(\lg n)$.)
4. ★ Seja n o número $r-p-1$. Seja $N(n)$ o número de execuções da comparação "$A[q] < x$" na linha 4 de BB-R. (A função N é relevante porque o consumo de tempo de BB-R é proporcional a $N(n)$. Assim, N tem o mesmo papel que a função T acima.) Escreva a recorrência que define $N(n)$. Mostre que $N(n) \le 8 \lg n$ para $n = 2, 3, 4$, etc. Mostre que $N(n) \ge \lg n$ para $n = 2, 3, 4$, etc.
5. Seja F a função definida sobre os números naturais pela recorrência $F(n) = F(\lfloor n/2 \rfloor) + 1$ para $n = 1, 2, 3, \ldots$, com valor inicial $F(0) = 0$. Prove que $F(n) \le \lg n + 1$ para $n = 1, 2, 3, \ldots$ (Sugestão: faça indução em n.)
6. Seja F a função definida sobre os números naturais pela recorrência $F(n) = F(\lfloor n/2 \rfloor) + n$ (para $n = 1, 2, 3, \ldots$), com valor inicial $F(0) = 0$. Dê uma boa cota superior para F. Repita o exercício para a recorrência $F(n) = F(\lfloor n/2 \rfloor) + \lg n$.
7. Um vetor $A[1 \mathinner{..} n]$ de números inteiros é *semi-compacto* se $A[i+1] - A[i] \le 1$ para $i = 1, 2, \ldots, n-1$. Escreva um algoritmo que receba um vetor semi-compacto $A[1 \mathinner{..} n]$ e um inteiro x tais que $A[1] \le x \le A[n]$ e devolva um índice i no intervalo $1 \mathinner{..} n$ tal que $A[i] = x$. Seu algoritmo deve consumir $\mathrm{O}(\lg n)$ unidades de tempo.
8. É dado um número inteiro s e um vetor crescente $A[1 \mathinner{..} n]$ de números inteiros. Quero saber se existem dois elementos do vetor cuja soma é exatamente s. Dê um algoritmo que resolva o problema em tempo $\mathrm{O}(n \lg n)$.
9. Discuta a busca ternária.

---

www.ime.usp.br/~pf/analise_de_algoritmos/
Atualizado em 2020-09-16

Departamento de Ciência da Computação
Instituto de Matemática e Estatística da USP