Algoritmos de Procura e de Ordenação I

02/10/2024

Sumário

- Recap
- Ordens de complexidade : Análise experimental
- Análise do Caso Médio : Abordagem estruturada
- Linear Search – Procura sequencial num array não ordenado
- Linear Search – Procura sequencial num array ordenado
- Binary Search – Procura binária num array ordenado
- Exercícios / Tarefas
- Sugestão de leitura

Recapitulação

Ordens de Complexidade – Notação habitual

notation	provides	example	shorthand for	used to
Big Theta	asymptotic order of growth	$\Theta(N^2)$	$\frac{1}{2} N^2$ $10 N^2$ $5 N^2 + 22 N \log N + 3N$ $\vdots$	classify algorithms
Big Oh	$\Theta(N^2)$ and smaller	$O(N^2)$	$10 N^2$ $100 N$ $22 N \log N + 3 N$ $\vdots$	develop upper bounds
Big Omega	$\Theta(N^2)$ and larger	$\Omega(N^2)$	$\frac{1}{2} N^2$ N^5 $N^3 + 22 N \log N + 3 N$ $\vdots$	develop lower bounds

[Sedgewick & Wayne]

Best case, Worst case, Average Case

$$B(n) = \min_{I \in D_n} t(I) \qquad W(n) = \max_{I \in D_n} t(I)$$

$$A(n) = \sum_{I \in D_n} p(I) \times t(I)$$

Procura da 1ª ocorrência do maior elemento

```
int searchMax( int a[], int n ) {
        int indexMax = 0;
        for( int i=1; i<n; i++ ) {
                if( a[i] > a[indexMax] ) {
                        indexMax = i;
                }
        }
        return indexMax;
}
```

(n – 1) comps.

Atribuições a indexMax:

$B(n) = 1$

$W(n) = n$

$A(n) \approx n/2$

Ordens de Complexidade – Análise experimental

Ordens de Complexidade / Classes de Eficiência

order of growth	name	typical code framework	description	example	$T(2N) / T(N)$
1	**constant**	`a = b + c;`	statement	add two numbers	1
$\log N$	**logarithmic**	`while (N > 1) { N = N / 2; ... }`	divide in half	binary search	~ 1
N	**linear**	`for (int i = 0; i < N; i++) { ... }`	loop	find the maximum	2
$N \log N$	**linearithmic**	[see mergesort lecture]	divide and conquer	mergesort	~ 2
N^2	**quadratic**	`for (int i = 0; i < N; i++) for (int j = 0; j < N; j++) { ... }`	double loop	check all pairs	4
N^3	**cubic**	`for (int i = 0; i < N; i++) for (int j = 0; j < N; j++) for (int k = 0; k < N; k++) { ... }`	triple loop	check all triples	8
2^N	**exponential**	[see combinatorial search lecture]	exhaustive search	check all subsets	$T(N)$

[Sedgewick & Wayne]

Exemplo – Ordem de complexidade ?

n	1	2	4	8	16	32	64	128	256
$M(n)$	1	3	10	36	136	528	2080	8256	32896

- M(n) : número de operações efetuadas para instância de tamanho n
- Ordem de complexidade ?
- Expressão para o número de operações ?

Outro exemplo – Ordem de complexidade ?

n	1	2	3	4	5	6	7	8	9	10
$M(n)$	1	3	7	15	31	63	127	255	511	1023

- M(n) : número de operações efetuadas para instância de tamanho n
- Ordem de complexidade ?
- Expressão para o número de operações ?

Tempo de execução – Estimativas

order of growth of time				for a program that takes a few hours for input of size N	
description	function	2x factor	10x factor	predicted time for 10*N*	predicted time for10*N* on a 10x faster computer
linear	N	2	10	a day	a few hours
linearithmic	$N \log N$	2	10	a day	a few hours
quadratic	N^2	4	100	a few weeks	a day
cubic	N^3	8	1,000	several months	a few weeks
exponential	2^N	2^N	2^{9N}	never	never

Predictions on the basis of order-of-growth function

[Sedgewick & Wayne]

Linear Search
– Array não ordenado

Procura sequencial – Nº de Comparações ?

```
int search( int a[], int n, int x ) {
    for( int i=0; i<n; i++ ) {
        if( a[i] == x ) {
            return i;
        }
    }
    return -1;
}
```

$B(n) = 1$

$W(n) = n$

$A(n) = ?$

Caso Médio – Abordagem estruturada

- Estabelecer um cenário
- Identificar a operação básica a contar
- Identificar os casos / configurações possíveis
  - Quantos são ?
- Contar o nº de operações realizadas para cada um dos casos
- Atribuir uma probabilidade a cada um dos casos possíveis
- Construir uma tabela
- Calcular o nº de operações para o caso médio

Cenário – Valor procurado existe no array

- O valor procurado pertence ao array
- Contar o número de comparações necessárias para o encontrar
- Valor encontrado na 1ª posição, ou na 2ª posição, ou …
- Necessária 1 comparação, ou 2 comparações, ou …
- Situações equiprováveis

- Tabela ?

Cenário – Valor procurado existe no array

Casos possíveis		Nº de comparações	Probabilidade
I_0	É o 1º elemento	1	1/n
I_1	É o 2º elemento	2	1/n
I_2	É o 3º elemento	3	1/n
…	….	…	…
I_{n-1}	É o último elemento	n	1/n

$$A(n) = \sum_{i=0}^{n-1} \frac{1}{n} \times (i+1) = \frac{1}{n} \sum_{i=0}^{n-1} (i+1) = \frac{1}{n} \times \frac{n \times (n+1)}{2} = \frac{n+1}{2} \approx \boldsymbol{\frac{n}{2}}$$

Cenário – Valor procurado pode não existir

- O valor procurado pertence ao array com probabilidade **p**
- O valor procurado não pertence ao array com probabilidade **(1 – p)**
- Quantos casos são ?
- Contar o número de comparações para cada um dos casos
- Valor encontrado na 1ª posição, ou na 2ª posição, ou …
- Necessária 1 comparação, ou 2 comparações, ou …
- Que probabilidade atribuir a cada caso ?

- Tabela ?

Cenário – Valor procurado pode não existir

Casos possíveis		Nº de comparações	Probabilidade
I_0	É o 1º elemento	1	p / n
I_1	É o 2º elemento	2	p / n
I_2	É o 3º elemento	3	p / n
…	….	…	…
I_{n-1}	É o último elemento	n	p / n
I_n	Não encontrado !	n	(1 – p)

$$A(n) = \sum_{i=0}^{n-1} \frac{p}{n} \times (i+1) \;+\; (1-p) \times n = \frac{p \times (n+1)}{2} + (1-p) \times n$$

Cenário – Valor procurado pode não existir

$$A(n) = \frac{p \times (n+1)}{2} + (1 - p) \times n$$

- Se **p = 1**, então A(n) = (n + 1) / 2 **≈ n / 2** comparações – OK
- Se **p = 50%**, então A(n) = (n + 1) / 4 + n / 2 **≈ 3 x n / 4**
- Se **p = 25%**, então A(n) = (n + 1) / 8 + 3 x n / 4 **≈ 7 x n / 8**

Linear Search
– Array ordenado

Procura sequencial – Nº de Comparações ?

```
int search( int a[], int n, int x ) {
    int stop = 0; int i;
    for( i=0; i<n; i++ ) {
        if( x <= a[i] ) {
            stop = 1; break;
        }
    }
    if( stop && x == a[i] ) return i;        // Ordem da conjunção !!
    return -1;
}
```

Melhor Caso e Pior Caso

- Valor procurado pode pertencer ou não pertencer ao array
- **B(n) = 2** - Quando ?
- **W(n) = (n + 1)** - Quando ?

Melhor Caso e Pior Caso

- Valor procurado pode pertencer ou não pertencer ao array

- **B(n) = 2** - Valor procurado é igual ao 1º elemento

  OU

  - Valor procurado é menor do que o 1º elemento

- **W(n) = (n + 1)** - Valor procurado é igual ao último elemento

  OU

  - Valor procurado está entre o penúltimo e o último

Caso Médio – Casos possíveis

Casos possíveis		Nº de comparações	Probabilidade
Sucesso 0	É o 1º elemento		
Sucesso 1	É o 2º elemento		
…	…		
Sucesso (n – 1)	É o último elemento		
Insucesso 0	Menor do que o 1º		
Insucesso 1	Entre o 1º e o 2º		
…	…		
Insucesso (n – 1)	Entre o penúltimo e o último		
Insucesso n	Maior do que o último		

- Quais são os casos possíveis ? Quantas comparações em cada um ?

Caso Médio – Nº de comparações efetuadas

Casos possíveis		Nº de comparações	Probabilidade
Sucesso 0	É o 1º elemento	2	
Sucesso 1	É o 2º elemento	3	
…	…	…	
Sucesso (n – 1)	É o último elemento	n + 1	
Insucesso 0	Menor do que o 1º	2	
Insucesso 1	Entre o 1º e o 2º	3	
…	…	…	
Insucesso (n – 1)	Entre o penúltimo e o último	n + 1	
Insucesso n	Maior do que o último	n	

- Que probabilidade associar a cada caso ?

Caso Médio – **p** = probabilidade de sucesso

Casos possíveis		Nº de comparações	Probabilidade
Sucesso 0	É o 1º elemento	2	p / n
Sucesso 1	É o 2º elemento	3	p / n
…	…	…	…
Sucesso (n – 1)	É o último elemento	n + 1	p / n
Insucesso 0	Menor do que o 1º	2	(1 – p) / (n +1)
Insucesso 1	Entre o 1º e o 2º	3	(1 – p) / (n +1)
…	…	…	…
Insucesso (n – 1)	Entre o penúltimo e o último	n + 1	(1 – p) / (n +1)
Insucesso n	Maior do que o último	n	(1 – p) / (n +1)

- A(n) = ?

Caso Médio – p = probabilidade de sucesso

$$A(n) = \frac{p}{n} \times \left(\sum_{i=0}^{n-1} (i+2) \right) + \frac{1-p}{n+1} \times \left(\sum_{i=0}^{n-1} (i+2) \right) + \frac{1-p}{n+1} \times n$$

$$A(n) = \frac{p}{n} \times \frac{n \times (n+3)}{2} + \frac{1-p}{n+1} \times \frac{n \times (n+3)}{2} + \frac{1-p}{n+1} \times n$$

$$A(n) = \frac{n}{2} + 2 - \frac{p}{2} + \frac{2 \times (p-1)}{n+1} \approx \frac{\boldsymbol{n}}{\mathbf{2}} + \mathbf{2} - \frac{\boldsymbol{p}}{\mathbf{2}}$$

- Comparar com o algoritmo para o array não ordenado
- Melhor ou pior ?

Binary Search
– Array ordenado

Procura binária

0	1	2	3	4	5
2	**4**	**6**	**8**	**10**	**12**

- O valor 2 pertence ao array ?

- Qual é o primeiro elemento consultado ?

Procura binária – 1º elemento consultado

0	1	2	3	4	5
2	**4**	**6**	**8**	**10**	**12**

- O valor 2 pertence ao array ?

0	1	2	3	4	5
2	**4**	**6**	**8**	**10**	**12**

?

Procura binária – Restringir a procura

0	1	2	3	4	5
2	**4**	**6**	**8**	**10**	**12**

- O valor 2 pertence ao array ?

0	1	2	3	4	5
2	**4**	**6**	**8**	**10**	**12**

0	1
2	**4**

Qual é o próximo elemento consultado?

Procura binária – 2º elemento consultado

0	1	2	3	4	5
2	**4**	**6**	**8**	**10**	**12**

- O valor 2 pertence ao array ?

0	1	2	3	4	5
2	**4**	**6**	**8**	**10**	**12**

0	1
2	**4**

Encontrado !!

Procura binária

- O valor 13 pertence ao array ?

0	1	2	3	4	5
2	**4**	**6**	**8**	**10**	**12**

Procura binária – 1º elemento consultado

- O valor 13 pertence ao array ?

0	1	2	3	4	5
2	**4**	**6**	**8**	**10**	**12**

Procura binária – Restringir a procura

- O valor 13 pertence ao array ?

0	1	2	3	4	5
2	**4**	**6**	**8**	**10**	**12**

3	4	5
8	**10**	**12**

Procura binária – 2º elemento consultado

- O valor 13 pertence ao array ?

0	1	2	3	4	5
2	**4**	**6**	**8**	**10**	**12**

3	4	5
8	**10**	**12**

Procura binária – Restringir a procura

- O valor 13 pertence ao array ?

0	1	2	3	4	5
2	**4**	**6**	**8**	**10**	**12**

3	4	5
8	**10**	**12**

5
12

Procura binária – Fim da procura

- O valor 13 pertence ao array ?

0	1	2	3	4	5
2	**4**	**6**	**8**	**10**	**12**

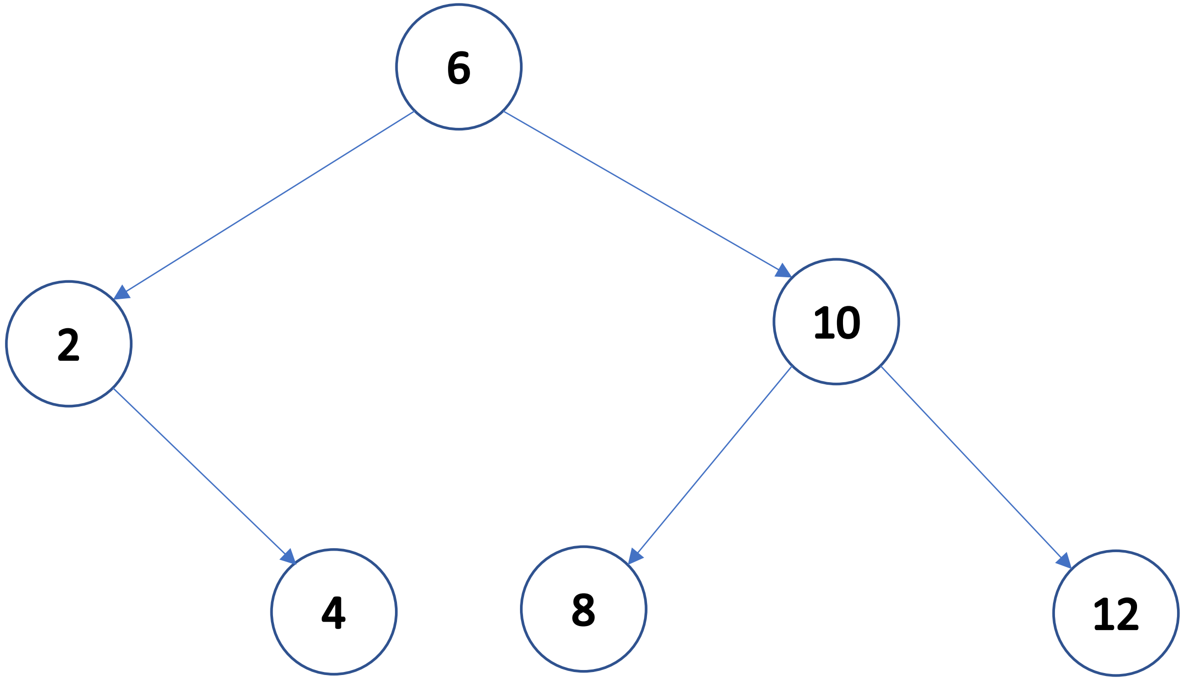
3	4	5
8	**10**	**12**

5
12

Não encontrado !!

Representação do array como árvore binária

- A representação gráfica como árvore binária auxilia a compreensão do funcionamento do algoritmo



Exercícios / Tarefas

Exercício 1 – Verdadeiro ou Falso

- Ao efetuar, usando o algoritmo clássico, o produto de matrizes ***A*** **×** ***B*** **=** ***C***, em que a matriz *A* tem 10 linhas e 20 colunas e a matriz *B* tem 20 linhas e 10 colunas, são efetuadas ***2000*** **multiplicações**.

- Ao efetuar, usando o algoritmo clássico, os produtos de matrizes ***A*** **× (*B*** **×** ***C*)** **=** ***D***, em que a matriz *A* tem 2 linhas e 3 colunas, a matriz *B* tem 3 linhas e 4 colunas e a matriz *C* tem 4 linhas e 5 colunas, são efetuadas, **no total, *120* multiplicações**.

Exercício 2 – Escolha múltipla

A tabela apresenta o número de operações básicas efetuadas, por um determinado algoritmo, para sucessivos valores de n.

n	1	2	4	8	16	32	64	128	256	512	1024
$M(n)$	3	4	5	6	7	8	9	10	11	12	13

Trata-se de um algoritmo com ordem de complexidade:

a) logarítmica.

b) linear.

c) quadrática.

d) Nenhuma está correta.

Exercício 3 – Escolha múltipla

A tabela apresenta o número de operações básicas efetuadas, por um determinado algoritmo, para sucessivos valores de *n*.

n	1	2	4	8	16	32	64	128	256
M(n)	4	12	40	144	544	2112	8320	33024	131584

Trata-se de um algoritmo com ordem de complexidade:

a) logarítmica.

b) linear.

c) quadrática.

d) Nenhuma está correta.

Exercício 4 – Escolha múltipla

A tabela apresenta o número de operações básicas efetuadas, por um determinado algoritmo, para sucessivos valores de *n*.

n	1	2	4	8	16	32	64	128	256	512	1024
$M(n)$	0	1	3	7	15	31	63	127	255	511	1023

Trata-se de um algoritmo com ordem de complexidade:

a) logarítmica.

b) linear.

c) quadrática.

d) Nenhuma está correta.

Exercício 5 – Escolha múltipla

Considere o *array* ordenado de 8 elementos. Usando o algoritmo de **procura binária**:

0	1	2	3	4	5	6	7
1	3	5	7	9	11	13	15

a) O elemento de valor 5 é encontrado ao fim de 3 tentativas.

b) O elemento de valor 11 é encontrado ao fim de 2 tentativas.

c) Ambas estão corretas.

d) Nenhuma está correta.

Tarefa 1 – Procura Sequencial

- Considere as duas versões do algoritmo
- Usando as expressões obtidas na aula, determine o número de comparações efetuadas no caso médio para diferentes valores da probabilidade de o valor procurado pertencer ao array
- Por exemplo, considere p = 10%, 20%, 40%, 60% e 80%
- Construa uma tabela e analise os valores obtidos

Tarefa 2 – Procura Binária

- Simule o funcionamento do algoritmo para arrays ordenados com um maior número de elementos
- Por exemplo, para n = 13 e n = 15
- Habitualmente expressa-se o esforço computacional deste algoritmo através do número de consultas a elementos do array
- Melhor caso : Quando ocorre ? Quantos elementos são consultados ?
- Pior caso : Quando ocorre ? Quantos elementos são consultados ?

Sugestão de leitura

Sugestão de leitura

- J. J. McConnell, Analysis of Algorithms, 1st Edition, 2001
  - Capítulo 2: secções 2.1, 2.2