# Projeto e Análise de Algoritmos

Atividade A2 - Nota Processual - 18/11/2021 - Prof. Dr. Aparecido Freitas

**richard.santos@uscsonline.com.br** Alternar conta

Rascunho salvo.

Seu e-mail será registrado quando você enviar este formulário.

*Obrigatório

Com relação à uma árvore binária de busca, assinale a alternativa correta. 5 pontos *

- ( ) Por ser uma árvore binária, uma árvore binária de busca somente pode ter 0 (zero) ou 1 (um) filho.
- ( ) A complexidade de pior caso do processo de busca em uma árvore binária de busca é sempre maior do que a busca em uma árvore binária qualquer.
- ( ) Uma característica comum nas árvores binárias de busca é que todas são cheias, ou seja, todas as sub-árvores vazias pertencem aos nós do último nível.
- (•) Uma árvore binária de busca é caracterizada por seus elementos estarem organizados seguindo alguma ordem pré-definida, sendo também conhecidas como árvore binária ordenada.
- ( ) Como em uma árvore binária de busca os elementos estão fora de ordem, quando se deseja buscar um elemento, é necessário percorrer todos os elementos presentes na árvore até encontrar o elemento buscado.

Pesquisa Binária e Hash Code são duas técnicas de busca de dados em um arquivo ou tabela muito usados em informática, com grande vantagem sobre a Pesquisa Sequencial. Sobre essas técnicas, assinale a afirmação INCORRETA. * 5 pontos

- ○ Na Pesquisa Binária, os dados devem estar classificados pelo campo que é a chave de busca.
- ○ Na Pesquisa Binária, o número mínimo de tentativas para localizar um registro é 1, e o máximo é log2 n (arredondado para cima), no qual n é o tamanho do arquivo ou tabela.
- ○ Na técnica Hash Code, o número de tentativas para localizar um registro quando o arquivo é grande não aumenta significativamente, tal como acontece na Pesquisa Sequencial.
- ○ Na técnica Hash Code, o número máximo de tentativas para localizar um registro depende do método empregado e do índice de ocupação do arquivo ou tabela em relação ao tamanho máximo estimado.
- ◉ Na técnica Hash Code, os dados devem estar classificados pelo campo que é a chave de busca.

Indique a alternativa que representa uma Função de complexidade de algoritmos, cujo tempo de execução ocorre tipicamente em algoritmos que resolvem um problema quebrando-o em problemas menores, resolvendo cada um deles independentemente e, depois, ajuntando as soluções: * 5 pontos

- ○ O(2 n) (dois elevado a n)
- ○ O(n)
- ○ O(n 2) ( n ao quadrado)
- ○ O(log n)
- ◉ O( n log n)

- ◯ O( n 3) ( n ao cubo)

Desempenho é a grande vantagem da tabela na utilização hash. O tempo de busca na tabela hash tem complexidade O(1), se desconsiderarmos as colisões; entretanto, se as colisões são tratadas usando uma lista encadeada, qual é o tempo de busca máximo para uma tabela hash com n colisões? * 5 pontos

O(n)

Um algoritmo que apresenta a menor complexidade dentre todos os possíveis algoritmos para resolver o mesmo problema é considerado um algoritmo * 5 pontos

- ◯ matriz
- ◉ ótimo
- ◯ operacional
- ◯ simplificado
- ◯ natural

Um algoritmo é executado em 3 segundos para uma entrada de tamanho 100. Se o algoritmo é quadrático, quanto tempo em segundos ele gastará, aproximadamente, no mesmo computador, se a entrada tiver tamanho 200? * 5 pontos

C X 100^2 = 3
10000C = 3
C=3/10000

T(200) = 3/10000 X 200^2 = 120000/10000 = 12

Considere a Sequência de Fibonacci (0, 1, 1, 2, 3, 5, 8, 13, ...), onde os dois primeiros termos valem 0 e 1 respectivamente, e cada termo seguinte é a soma de seus dois predecessores.O pseudocódigo a seguir apresenta um algoritmo simples para o cálculo do N-ésimo termo dessa sequência. Qual a ordem de complexidade desse algoritmo ? * 5 pontos

```
function fibo (N)
if n = 1 then
    return 0
elif n = 2 then
    return 1
else
    penultimo := 0
    ultimo := 1
    for i := 3 until N do
        atual := penultimo + ultimo
        penultimo := ultimo
        ultimo := atual
    end for
    return atual
end if
```

O(n)

Quando dois elementos estão fora de ordem, há uma inversão, e esses dois elementos são trocados de posição, ficando em ordem correta. Assim, o primeiro elemento é comparado com o segundo. Se uma inversão for encontrada, a troca é feita. Em seguida, independentemente de se houve ou não troca após a primeira comparação, o segundo elemento é comparado com o terceiro, e, caso uma inversão seja encontrada, a troca é feita. O processo continua até que o penúltimo elemento seja comparado com o último. Com esse processo, garante-se que o elemento de maior valor do vetor seja levado para a última posição. A ordenação continua com o posicionamento do segundo maior elemento, do terceiro etc., até que todo o vetor esteja ordenado. CELES, W.; CERQUEIRA, R.; RANGEL, J. L. Introdução a Estruturas de Dados. Rio de Janeiro: Elsevier, 2004, com adaptações.Em relação ao algoritmo descrito, é correto afirmar que a respectiva ordem de complexidade, no pior caso, é * 5 pontos

- [ ] O(log n)
- [x] O(n)
- [ ] O(n2)
- [ ] O(n log n)
- [ ] O(n3)

*

5 pontos

Deseja-se efetuar uma busca para localizar uma certa chave fixa *x*, em uma tabela contendo *n* elementos. A busca considerada pode ser a linear ou binária. No primeiro caso pode-se considerar que a tabela esteja ordenada ou não. No segundo caso a tabela está, de forma óbvia, ordenada.

Assinale a alternativa ***CORRETA:***

- A busca binária sempre localiza x, efetuando menos comparações que a busca linear.
- A busca linear ordenada sempre localiza x, efetuando menos comparações que a não ordenada.
- A busca linear não ordenada sempre localiza x, com menos comparações que a ordenada.
- A busca binária requer O(log n) comparações, no máximo, para localizar x.
- A busca linear ordenada nunca requer mais do que n/2 comparações para localizar x.

Considere o seguinte trecho de código em Java para ordenação de um conjunto de números. Qual a ordem de complexidade do algoritmo ? *

5 pontos

```
int[] numbers = {40, 7, 59, 4, 1};
for (int j =0 ; j < numbers.length; j++) {
      for(int i = 0; i < numbers.length-1; i++) {
            if (numbers[i] > numbers[i+1]) {
                  int temp = numbers[i];
                  numbers[i] = numbers[i+1];
                  numbers[i+1] = temp;
            }
      }
}
```

O(n^2)

*

5 pontos

Dado o algoritmo:

```
int func (int n)  {
    int a = 2;
    for (int i = 2; i < n-2; i++)
        a = a + 2;                    //1
    }
```

Quantas vezes a instrução //1 é executada?

2

*

5 pontos

O algoritmo clássico a seguir, implementado em um método em Java, é chamado de busca binária. O algoritmo recebe como parâmetro um vetor de inteiros *v* e um inteiro *num* e retorna verdadeiro lógico caso aquele inteiro *num* apareça no vetor *v* ou falso lógico em caso contrário. Considerando que a variável n é o tamanho do vetor v, qual das opções a seguir representa a ordem de grandeza de operações que precisam ser feitas para se buscar um inteiro *num* com a busca binária em um vetor *v* de tamanho n?

```
public boolean buscaBinaria( int v[], int num ) {
        int esq = 0;
        int dir = v.length-1;
        while(esq<=dir) {
                int meio = (esq+dir)/2;
                if(v[meio]==num) return true;
                if(v[meio]<num) esq = meio+1;
                else dir = meio-1;
        }
        return false;
}
```

- (●) log(n) operações, considere a base 2.
- ( ) n operações.
- ( ) 2n (o dobro de n) operações.
- ( ) n x n (n ao quadrado) operações.
- ( ) n/2 (metade de n) operações.

Um método de busca bastante utilizado, conhecido como hash, baseia-se na utilização que mapeia chaves em endereços de memória, de modo que os dados associados a cada chave possam ser rapidamente localizados e lidos. Quando há conflitos de localização, algum algoritmo de tratamento de colisão é empregado.Considere uma tabela hash armazenada em um arquivo no disco rígido. Supondo-se que a mesma possua uma função de hash no qual não haja conflitos, o número médio de acessos ao disco necessários para se recuperar um dado associado a uma determinada chave em um universo de N chaves, corresponde a * 5 pontos

- ○ n log n
- ○ log n
- ○ n / 2
- ○ n / log n
- ◉ 2

*

5 pontos

A *sequência de Fibonacci* é uma sequência de inteiros, cujo primeiro termo é 0, o segundo termo é 1, e a partir do terceiro, cada termo é igual à soma dos dois anteriores. O seguinte algoritmo recursivo retorna o *n*-ésimo termo da sequência

```
Procedimento F(n)
        se n < 3 então retornar n-1
        senão retornar F(n-1) + F(n-2)
```

A chamada externa é F(n), sendo n > 0.

Assinale a alternativa ***CORRETA:***

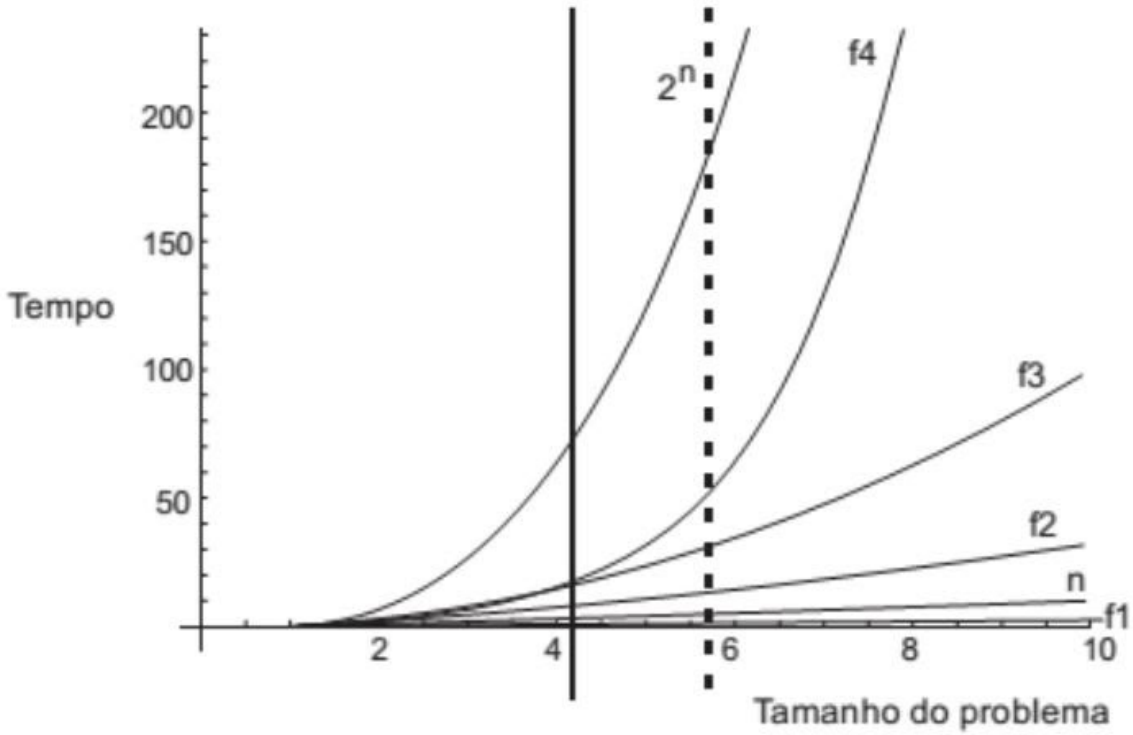
- ○ O algoritmo não está correto, pois não retorna o n-ésimo termo da sequência.
- ○ O algoritmo é ótimo, no que diz respeito ao número de passos.

- ○ O número de passos efetuados pelo algoritmo é linear em n.
- ○ O número de passos efetuados pelo algoritmo é polinomial em n.
- ◉ O número de passos efetuados pelo algoritmo é exponencial em n.

Uma árvore binária completa de busca, isto é, uma árvore em que todos os níveis têm o máximo número de elementos, tem um total de N nós.O número máximo de comparações necessárias para encontrar um elemento nessa árvore é * 5 pontos

- ○ N
- ○ N2 ( N ao quadrado)
- ○ log (N)
- ◉ (N+1)
- ○ N3 (N ao cubo)

O gráfico abaixo mostra a relação de dominação assintótica entre funções de complexidade de algoritmos. Os valores de tempo e tamanho do problema são apenas referenciais. Considere apenas os seus valores crescentes. * 5 pontos




- A notação O permite comparar funções de complexidade assintótica. No caso da figura, um programa O(f4) é sempre melhor que um O(f3).
- o comportamento assintótico de uma função f (n) corresponde ao limite do comportamento do custo quando n aproxima-se de 2n.
- f1, no gráfico, corresponde à função n log2n.
- f2, no gráfico, corresponde à função log2n.
- f3 e f4, embora sejam exponenciais, apresentam desempenho superior a 2 elevado a n.

Assinale a alternativa que apresenta o tempo de execução do pior caso e do melhor caso para o algoritmo Quicksort. * 5 pontos

- ○ Pior caso: O(n2) ; melhor caso: O(n)
- ○ Pior caso: O(n log n) ; melhor caso: O(n)
- ○ Pior caso: O(n) ; melhor caso: O(n + m)
- ○ Pior caso: O(n log n) ; melhor caso: O(n + m)
- ◉ Pior caso: O(n2) ; melhor caso: O(n log n)

* 5 pontos

**A pesquisa de dados envolve a determinação da chave pesquisada estar ou não entre os dados pesquisados e, caso esteja, que seja encontrada sua localização. Em computação, a pesquisa tem um papel importante, pois de posse do campo chave a ser pesquisado fica mais fácil encontrar determinado arquivo, ou mesmo qualquer item que se queira buscar. Já a classificação envolve a organização dos dados em uma determinada ordem, por exemplo: crescente, decrescente, ordem alfabética, numérica, entre outros. Acerca dos algoritmos de pesquisa e classificação, analise as afirmativas a seguir.**

I. Diz-se que o algoritmo *O(log n)* tem um tempo de execução linear.
II. A pesquisa binária executa em O(*log n*) vezes, pois cada passo remove metade dos elementos restantes.
III. O algoritmo de classificação por inserção executa no tempo $O(n^2)$, no pior caso e no caso médio.
IV.No pior caso, a primeira chamada à classificação por intercalação tem de fazer O(*n*) comparações para preencher os n *slots* no *array* final.

**Estão corretas apenas as afirmativas**

- ◉ II e IV
- ○ I e III
- ○ I e IV
- ○ II e III
- ○ I e II

No pior caso, a complexidade do algoritmo conhecido por Busca Linear é: 5 pontos *

- O(n2) (n ao quadrado)
- O(1)
- O(n)
- O(log n)
- O( n logn)

Considerando que o programa abaixo não reutilize resultados previamente computados, quantas chamadas são feitas à função fib para computar fib (3) ? * 5 pontos

Os números de Fibonacci constituem uma seqüência de números na qual os dois primeiros elementos são 0 e 1 e os demais, a soma dos dois elementos imediatamente anteriores na seqüência. Como exemplo, a seqüência formada pelos 10 primeiros números de Fibonacci é: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34. Mais precisamente, é possível definir os números de Fibonacci pela seguinte relação de recorrência:

fib (n) = 0, se n = 0
fib (n) = 1, se n = 1
fib (n) = fib (n – 1) + fib (n – 2), se n > 1

Abaixo, apresenta-se uma implementação em linguagem funcional para essa relação de recorrência:

```
fib :: Integer -> Integer
fib 0 = 0
fib 1 = 1
fib n = fib (n - 1) + fib (n - 2)
```

5

Enviar Limpar formulário

Nunca envie senhas pelo Formulários Google.