Lista de exercícios sobre contagem de operações
Prof. João B. Oliveira

1.
```
metodo m ( Vetor V )
  int i, res = 0;
  para i de 1 a V.size
    res = res + V[i];
  return res;
```
Soma de elementos de um vetor, $O(\quad)$.

2.
```
metodo m ( Vetor V )
  int i, j;
  para i de 1 a V.size
    para j de i+1 a V.size
      se V[i] > V[j] -> troca(V[i], V[j]);
```
Ordena o vetor, é $O(\quad)$.

3. Dada uma lista encadeada A, escreva um algoritmo que escreve os elementos da lista de trás para a frente e analise seu desempenho em notação $O()$.

4.
```
metodo m ( LinkedList L )
  int i = L.length;
  enquanto i > 0 :
    imprime L.get(i);
    i--;
```
Imprime uma lista encadeada de trás para a frente, por isso é $O(\quad)$.

5.
```
metodo m ( LinkedList L )
  se L é NULL -> retorna;
  m( L -> prox );
  imprime L.get();
```
Imprime uma lista encadeada elemento a elemento, mas é $O(\quad)$ por causa da recursão.

6. Escreva um método que recebe dois arrays A e B e devolve como resultado um array C de booleanos, onde o booleano C_i indica se os elementos A_i e B_i são iguais.

7. Dado um vetor com números pares e ímpares, escreva um algoritmo para colocar todos os números pares à frente no vetor e os ímpares ao final. Escreva o algoritmo em duas versões:

   (a) Usando um vetor auxiliar

   (b) Sem usar outro vetor

8. Escreva um algoritmo que recebe dois arrays A e B representando conjuntos de m e n elementos respectivamente e em seguida preenche um vetor C que deve conter a intersecção dos conjuntos, ou seja, os elementos que estão em A e em B. Lembre que conjuntos não tem ordem nem repetições de elementos.

9. Repita o exercício anterior se C agora deve conter a união dos dois conjuntos.

10. Repita o exercício anterior se C agora deve conter os elementos que estão em A mas não estão em B.

11. Escreva um algoritmo que recebe dois vetores A e B contendo inteiros e escreve os elementos que estão em A e em B. Analise seu algoritmo em notação $O()$ para duas situações:

    (a) Você sabe que os vetores estão ordenados

    (b) Os vetores podem estar fora de ordem

12. Escreva um algoritmo que recebe dois vetores A e B contendo inteiros e acha o maior elemento que está em A e em B ao mesmo tempo. Analise seu algoritmo em notação $O()$ para duas situações:

    (a) Você sabe que os vetores estão ordenados

    (b) O vetor A pode estar fora de ordem.

    (c) Os dois vetores podem estar fora de ordem

13.
```
metodo m ( Vetor V )
  int i, j;
  para i de 1 a V.size
    para j de i+1 a V.size
      se V[i] é ímpar e V[j] é par -> troca(V[i], V[j]);
```
Altera o vetor colocando pares na frente e ímpares atrás, é $O($ $)$.

14.
```
metodo m ( Vetor V )
  int i, j;
  i = 1;
  j = V.size;
  enquanto i <= j :
    troca(V[i], V[j]);
    enquanto V[i] é par -> i++;
    enquanto V[j] é impar -> j--;
```
Ordena o vetor colocando pares na frente e ímpares atrás, mas é $O($ $)$.

15. Escreva um método que recebe um vetor A e dois inteiros a e n e insere o valor a na n-ésima posição de A, empurrando os outros elementos para a frente. Por exemplo, se $a = 4$ e $n = 6$ você deve inserir o valor 4 na sexta posição de A.

16.
```
metodo m ( Mat A, Mat B, int n )
  int i, j, k;
  Mat C = 0;
  para i de 1 a n
    para j de 1 a n
      para k de 1 a n
        C[i,j] += A[i,k] * B[k,j];
```
Multiplicação de matrizes, é $O($ $)$.

17.
```
metodo m ( int n )
  int inicio, fim, aux, s;
  para inicio de 1 a n
    para fim de inicio a n
    s = 0;
    para aux de inicio a fim
      s += aux;
    se s == n imprime inicio, fim;
```
Problema das fatorizações para um único valor de n, por isso é $O($ $)$. Colocando mais um laço para cobrir todos os valores de n, temos $O($ $)$.

18. Escreva um método que recebe um array A e troca de posição seu maior e seu menor elementos.

19. Escreva o algoritmo que recebe um vetor V contendo números reais e determina a média dos elementos do vetor. Qual o desempenho do seu algoritmo?

20. Aumente seu algoritmo anterior para determinar também o elemento do vetor que tem o valor mais próximo da média. Qual o desempenho do seu algoritmo?

21. Escreva um algoritmo para inverter a ordem dos elementos de um vetor V. Você **não pode** usar outro vetor como área auxiliar. Qual o desempenho do seu algoritmo?

22. Suponha que você tem um vetor de inteiros $A = (a_1, \ldots, a_n)$, contendo números positivos e negativos. Escreva o algoritmo que analisa este vetor e acha o trecho do vetor que produz o maior valor quando os elementos daquele trecho são somados. Em seguida, determine quantas somas são necessárias para obter o resultado usando seu algoritmo. Um algoritmo bastante simples pode ser feito usando tempo $O(n^2)$ e memória $O(n^2)$ e um mais sofisticado pode ser feito em tempo $O(n)$ e memória $O(1)$. Avalie seu algoritmo e verifique se ele é melhor ou pior.

23. Você recebe uma lista de números $L = \langle l_1, \ldots, l_n \rangle$ e deve calcular seu desvio padrão L_σ, que é dado por

$$L_\sigma = \sqrt{\sum_{i=1}^{n} (l_i - \overline{L})^2},$$

onde $\overline{L}$ é a média dos valores de L. Escreva o algoritmo e determine quantas operações de mutiplicação são necessárias.

24. Você recebe uma lista de números $L = \langle l_1, \ldots, l_n \rangle$ e deve determinar o menor e o maior valor na lista. Esboce o algoritmo e determine o número de comparações que ele usa. Tente reduzir este número ao mínimo possível.

25. Você recebe uma lista de números $L = \langle l_1, \ldots, l_n \rangle$ e deve quebrar a lista em duas, no ponto em que a média dos valores de um lado e a média do outro lado tem a menor diferença possível. Escreva um algoritmo para fazer isso e determine o número de comparações que ele usa. É fácil escrever um algoritmo que seja $O(n^2)$ e ele pode facilmente ser melhorado para $O(n)$.

26. Um colega mediu o tempo de execução de um algoritmo e para uma entrada de tamanho 1000 o algoritmo resolveu o problema em 30 segundos. Para o problema de tamanho 2000 seu colega disse que o algoritmo vai levar 60 segundos, pois basta fazer uma regra de três. Quando ele está certo? Quando ele está errado? (Na maior parte das vezes ele vai estar errado. Por que?)

27. Um algoritmo de ordenação precisa de um segundo para ordenar mil itens em seu computador. Estime quanto tempo levará para ordenar dez mil itens, se

    - você acredita que o tempo é proporcional a n^2.
    - você acredita que o tempo é proporcional a $n \log(n)$.

28. Dois algoritmos precisam de n^2 dias e n^3 segundos respectivamente para resolver um problema de tamanho n. A partir daí, desponda:

    (a) Usando notação $O()$, qual o algoritmo mais eficiente?

    (b) Em que tamanho n de problema os dois algoritmos tem o mesmo desempenho?

    (c) Quanto tempo vai levar para resolver esse problema?

    (d) Na prática, qual algoritmo deve ser usado?

29. Para cada um dos trechos de código descritos na tabela abaixo, apresente uma estimativa para o tempo de execução, em notação assintótica. Tente fazer com que sua estimativa seja a mais correta possível.

```
function mistério(int n)
    r := 0;
    for i := 1 to n − 1 do
        for j := i + 1 to n do
            for k := 1 to j do
                r := r + 1;
    return r;
```

```
function bizarro(int n)
    r := 0;
    for i := 1 to n do
        for j := 1 to i do
            for k := j to i + j do
                r := r + 1;
    return r;
```

```
function f1(int n)
    r := 0;
    for i := 1 to n do
        for j := 1 to i do
            for k := j to n do
                r := r + 1;
    return r;
```

```
function f2(int n)
    r := 0;
    for i := 1 to n − 1 do
        for j := 1 to n + i do
            r := r + n;
    return r;
```

```
function f3(int n)
    r := 0;
    for i := 1 to 2 ∗ n do
        for j := 1 to n do
            r := r + (i − j);
    return r;
```

```
function f4(int n)
    r := 0;
    for i := 1 to n^2 do
        for j := 1 to i do
            r := r + j;
    return r;
```

30. Dois algoritmos precisam de $n\sqrt{n}$ minutos e $20n^2$ segundos respectivamente para resolver um problema de tamanho n. Qual o melhor algoritmo para problemas pequenos (digamos, $n < 10$)? Qual o menor problema em que o primeiro algoritmo vence o segundo? Quanto tempo este problema leva para ser resolvido?

31. Dois algoritmos precisam de n^2 dias e 2^n segundos respectivamente para resolver um problema de tamanho n. Qual o menor problema em que o primeiro algoritmo vence o segundo? Quanto tempo este problema leva para ser resolvido?

32. Comparação de tempos: para cada função $f(n)$ e cada tempo t dados abaixo, determine o maior tamanho n de cada problema que pode ser resolvido no tempo t, assumindo que o algoritmo precisa de $f(n)$ microsegundos (um microsegundo = 10^{-6} segundo).

	1 segundo	1 minuto	1 hora	1 dia	1 mês	1 ano	1 século
$\log_2(n)$							
$\sqrt{n}$							
n							
$n \log_2(n)$							
n^2	10^3					5615692	
n^3							
2^n							
$n!$							

Por exemplo: se $f(n) = n^2$, então em um segundo poderemos resolver problemas de tamanho $n = 10^3$, pois neste caso $n^2 = (10^3)^2 = 10^6$, o número de microsegundos existente em um segundo. Já se tivermos um ano para o cálculo, temos de pensar no número de microsegundos existente em um ano (31536×10^9) e ver que número n satisfaz $f(n) = n^2 = 31536 \times 10^9$. Neste caso, concluímos que $n = 5615692.299$.

Interpretação: de um segundo para um ano temos 31536 mil vezes mais tempo, mas como o tempo para resolver o problema cresce quadraticamente com o tamanho do problema, em um ano só podemos resolver um problema 5.61 mil vezes maior do que em um segundo.

33. O que significa $O(1)$? Dê um exemplo de um algoritmo (ou operação) que segue esta ordem de complexidade.

34. Verifique se as seguintes afirmações são verdadeiras ou falsas:

    (a) $n^2 = O(n^3)$
    (b) $n^3 = O(n^2)$
    (c) $2n^2 + 1 = O(n^2)$
    (d) $n \log n = O(n\sqrt{n})$
    (e) $\sqrt{n} = O(\log n)$
    (f) $\log n = O(\sqrt{n})$
    (g) $3n^2 + \log n = O(n^2)$
    (h) $\sqrt{n} \log n = O(n)$

35. Dados os pares de funções $f(n)$ e $g(n)$ abaixo, é sempre verdade que $f(n) = O(g(n))$ ou $g(n) = O(f(n))$, mas as duas situações nunca são verdadeiras ao mesmo tempo. Então, decida qual das situações é válida para cada par:

    (a) $f(n) = (n^2 - n)/2$, $g(n) = 6n$
    (b) $f(n) = n + 2\sqrt{n}$, $g(n) = n^2$
    (c) $f(n) = n + \log n$, $g(n) = n\sqrt{n}$
    (d) $f(n) = n^2 + 3n + 4$, $g(n) = n^3$
    (e) $f(n) = n \log n$, $g(n) = n\sqrt{n/2}$
    (f) $f(n) = n + \log n$, $g(n) = \sqrt{n}$
    (g) $f(n) = 2(\log n)^2$, $g(n) = \log n + 1$
    (h) $f(n) = 4n \log n + n$, $g(n) = (n^2 - n)/2$

36. Dados os pares de funções $f(n)$ e $g(n)$ abaixo, verifique se $f(n) = O(g(n))$, se $f(n) = \Omega(g(n))$, se $f(n) = \Theta(g(n))$ ou se nada pode ser dito:

    (a) $f(n) = n^2 + 3n + 4$, $g(n) = 6n + 7$
    (b) $f(n) = \sqrt{n}$, $g(n) = \log(n + 3)$
    (c) $f(n) = n\sqrt{n}$, $g(n) = n^2 - n$
    (d) $f(n) = n + n\sqrt{n}$, $g(n) = 4n \log(n^2 + 1)$
    (e) $f(n) = (n^2 - 2)/(1 + 2^{-n})$, $g(n) = n + 3$
    (f) $f(n) = 2^n - n^2$, $g(n) = n^4 + n^2$

37. Escreva um algoritmo que examina um array A contendo números inteiros e calcula a média dos elementos do vetor. Depois o algoritmo encontra o valor de A que está mais próximo desta média e apresenta seu valor.

38. Escreva o algoritmo que recebe um array A de tamanho n contendo inteiros e encontra o par de elementos a e b do vetor que fazem com que a diferença $a - b$ seja a maior possível. Analise seu algoritmo em notação $O()$. Se seu algoritmo for $O(n^2)$ tente encontrar outro que seja $O(n)$.

39. Dada uma matriz M com m linhas e n colunas, escreva um algoritmo que acha a linha de M que tem a maior soma dos seus elementos. Analise seu algoritmo em notação $O()$.

40. Dada uma matriz M com m linhas e n colunas, escreva um algoritmo que acha a sub-matriz de M que tem a maior soma dos seus elementos. (Uma sub-matriz de uma matriz é qualquer bloco retangular de elementos). Analise seu algoritmo em notação $O()$.

41. O algoritmo abaixo serve para converter um número n para binário. Analise seu desempenho em notação $O()$.

```
método binario(inteiro n)
 string s = "";
 enquanto n > 0
   s = (n % 2) + s;
   n = ⌊n / 2⌋;
 retorna s;
```

42. Analise os dois algoritmos abaixo em notação $O()$.

```
método m1(inteiro n)
  local i, j, cont;
  cont = 0;
  para i de 1 a n
    j = 0;
    enquanto j * j < n {
      cont++;
      j++;
    }
  retorna cont;
```

```
método m2(inteiro n)
  local i, j, k, cont;
  cont = 0;
  para i de 1 a n
     para j de 2*i a 2*n
       k = 0;
       enquanto k < n {
         cont++;
         k++;
       }
  retorna cont;
```

43. Escreva um algoritmo que recebe um número X e um inteiro n e calcula X^n. Depois disso analise seu algoritmo em notação $O()$. O desempenho do seu algoritmo depende de X?

44. O algoritmo abaixo serve para separar os elementos de um vetor, colocando os pares na frente do vetor e os ímpares na frente. Analise seu desempenho em notação $O()$.

```
método organiza(array V, tamanho n)
  inteiro i;
  booleano ok = falso;
  enquanto not ok
      ok = verdadeiro;
      para i de 1 a n-1
        se V[i] é ímpar e V[i+1] é par
          troca V[i] e V[i+1];
          ok = falso;
```

45. Quanto custa multiplicar dois números inteiros, se usarmos o método manual que você aprendeu na escola? Suponha que os números tem m e n casas respectivamente.

46. Escreva um algoritmo que recebe um vetor A contendo n inteiros e determina se ele tem elementos repetidos. Analise seu algoritmo em notação $O()$ para duas situações:

    (a) Você sabe que o vetor A está ordenado

    (b) O vetor A pode estar fora de ordem

    (c) Escreva um algoritmo que retira de A seus elementos repetidos, deixando apenas uma ocorrência de cada um. Depois disso avalie o número de comparações feitas em seu algoritmo em função de n em notação $O()$. Tente obter um algoritmo $O(n^2)$, mas depois explique como podemos obter facilmente um algoritmo $O(n \log n)$.

47. O algoritmo abaixo foi modificado para converter um número n para uma base b qualquer. Analise seu desempenho em notação $O()$.

function f_1(**int** n)
  $r := 0$;
  for $i := 1$ **to** n **do**
  for $j := 1$ **to** i **do**
  for $k := j$ **to** n **do**
  $r := r + 1$;
  return r;
	function f_2(**int** n)
	  $r := 0$;
	  for $i := 1$ **to** $n - 1$ **do**
	  for $j := 1$ **to** $n + i$ **do**
	  $r := r + n$;
	  return r;
		function f_3(**int** n)
		  $r := 0$;
		  for $i := 1$ **to** $2 * n$ **do**
		  for $j := 1$ **to** n **do**
		  $r := r + (i - j)$;
		  return r;

Tabela 1: Avalie o desempenho dos algoritmos acima em notação $O()$

```
método binario(inteiro n, inteiro b)
  string s = "";
  enquanto n > 0
    s = (n % b) + s;
    n = ⌊n / b⌋;
  retorna s;
```

48. O algoritmo abaixo recebe um conjunto S de inteiros e escreve todos os subconjuntos de S. Analise seu desempenho em notação $O()$.

```
método sub(conjunto S, conjunto s)
  se S é vazio {
     escreve "{" s "}";
     retorna;
   }
  elemento a = retiraalgum(S);
  sub(S, s );
  sub(S, s + {a} );
```

49. O algoritmo abaixo também serve para separar os elementos de um vetor, colocando os pares na frente do vetor e os ímpares na frente. Analise seu desempenho em notação $O()$.

```
método organiza(array V, tamanho n)
  inteiro i, j;
  i = 1;
  j = n;
  faça
    enquanto V[i] é par faça i++;
    enquanto V[j] é ímpar faça j--;
    troca V[i] e V[j];
    i++;
    j--;
  enquanto (i<j);
```

50. Avalie o desempenho dos algoritmos da Tabela 1 em notação $O()$.

51.
```
metodo m ( int n )
  int res = 0;
  enquanto n > 0 :
    se n é ímpar -> res++;
    n = n / 2;
  return res;
```
Conta o número de bits de n, é $O($ $)$.

52.
```
metodo m ( int a, int b )
  int res = 0;
  enquanto b > 0 :
    se b é ímpar -> res += a;
    b = b / 2;
    a = a * 2;
 return res;
```
Multiplicação à moda russa, é $O($ $)$.

53. Para cada uma das estruturas à esquerda, determine o custo das operações à direita, apresentando-o em notação $O()$:

Estrutura	Operação
Árvore binária de pesquisa com balanceamento	Inserção de elemento
Árvore binária de pesquisa sem balanceamento	Busca de elemento qualquer
Max-heap baseado em vetor	Deleção de elemento qualquer
Lista encadeada com elementos em ordem	Busca do maior elemento
Lista encadeada sem ordem dos elementos	Busca do menor elemento
Vetor ordenado	Retirada do maior elemento
Vetor não-ordenado	Retirada do menor elemento

54. Você foi contratado por uma concessionária de automóveis que vende k modelos de automóvel. Como eles também são consertados na concessionária, você também tem acesso a uma lista de n peças para cada modelo de carro. Neste caso, responda as seguintes perguntas sobre o desempenho do seu sistema de gerência de peças:

    (a) Como é a notação $O()$ para a parte de leitura de dados?

    (b) O que seria melhor? Um algoritmo que fosse $O(k^2n)$ ou um que fosse $O(kn^2)$? Justifique sua resposta.

    (c) Você guardou todas as peças em uma árvore binária sem balanceamento. Quantos níveis ela pode ter, em notação $O()$?

    (d) Para a árvore da pergunta anterior, quantas operações custa a inserção de um nodo?

    (e) Usando uma árvore balanceada, como ficam as respostas das duas perguntas anteriores?

    (f) Se você fizer um vetor de árvores, com uma árvore para cada modelo de carro, como fica o desempenho da inserção de um nodo?