Cento Universitário UNA

Análise e Projeto de Algoritmos

Práticas de Laboratório

Wesley Dias Maciel

2017/01

Centro Universitário UNA
Sistemas de Informação
Análise e Projeto de Algoritmos
Prática de Laboratório
Wesley Dias Maciel
2017/01

Prática 01

Aula: somatório, vetor, matriz, algoritmos O(1), O(n), O(n^2) e O(n^3).

Exercícios:

Os algoritmos propostos abaixo devem ser escritos em linguagem de programação Java. Empregue interface de classe, atributos, métodos, encapsulamento e modularização de código. Apresenta a complexidade de tempo de cada algoritmo.

1) Escreva um algoritmo que leia um limite natural n a partir do teclado e que calcule os somatórios abaixo. O algoritmo deve imprimir cada termo e os resultados dos somatórios.

   a. $\sum_{i=0}^{n} i,\ i \in N,\ n \in N \text{ e } i \leq n.$

   b. $\sum_{i=0}^{n} (i + 5),\ i \in N,\ n \in N \text{ e } i \leq n.$

   c. $\sum_{i=2}^{n} (i - 3)^2/(i - 1),\ i \in N,\ n \in N \text{ e } i \leq n.$

2) Escreva um algoritmo que leia um limite natural n a partir do teclado e que calcule os produtórios abaixo. O algoritmo deve imprimir cada termo e os resultados dos produtórios.

   a. $\prod_{i=1}^{n} i,\ i \in N,\ n \in N \text{ e } i \leq n.$

   b. $\prod_{i=0}^{n} (i + 12),\ i \in N,\ n \in N \text{ e } i \leq n.$

   c. $\prod_{i=3}^{n} (i / 3)^4/(i - 2),\ i \in N,\ n \in N \text{ e } i \leq n.$

3) Escreva um algoritmo que preencha um vetor de 10 posições com números inteiros gerados randomicamente. O algoritmo deve informar se o primeiro elemento do vetor é par ou impar.

4) Escreva um algoritmo que preencha um vetor de 10 posições com números inteiros gerados randomicamente. O algoritmo deve trocar o elemento da primeira posição do vetor com o elemento da última posição do vetor. O algoritmo deve imprimir os números da primeira e da última posição do vetor antes e após a troca.

5) Escreva um algoritmo que preencha um vetor de 10 posições com números inteiros gerados randomicamente. O algoritmo deve apresentar a quantidade de números pares e a quantidade de números ímpares armazenados no vetor.

6) Escreva um algoritmo que preencha uma matriz 10 x 10 com números reais gerados randomicamente. O algoritmo deve apresentar os elementos da diagonal principal e da diagonal secundária da matriz. Exemplo:

M =

-1,9	-4	7,34	2	22	-1	9	234	17	2
3	5,5	6	9	0	6,78	4	1	12	3
5	-3,8	2	3	5	78	5	890	3	7
8	2	8	4	67	45	67	467	-9	9
0	1	4	8	87	13	1234	432	0	0
6	-4	9	9,8	54	-3	56	78	21	2,8
-3	3	0	7	35	90	8	65	67	0
0	-0,7	10	2	89	7	0	24	999	3,7
15	5	0	-39	34	4	1	-5	15	-15
-5	7	0	7,6	-98	2,5	-1	0	9	1

Diagonal principal = {-1,9; 5,5; 2; 4; 87; -3; 8; 24; 15; 1}
Diagonal secundária = {2; 12; 890; 67; 13; 54; 7; 10; 5; -5}

7) Escreva um algoritmo que preencha uma matriz 5 x 5 com números inteiros gerados randomicamente. O algoritmo deve apresentar a transposta da matriz gerada. Exemplo:

M =

1	4	7	2	2
3	5	6	9	3
5	3	2	3	7
8	2	8	4	9
5	7	0	7	1

$M^t =$

1	3	5	8	5
4	5	3	2	7
7	6	2	8	0
2	9	3	4	7
2	3	7	9	1

8) Escreva um algoritmo que preencha uma matriz 10 x 10 com números reais gerados randomicamente. O algoritmo deve somar os elementos abaixo da diagonal principal (região amarela na figura abaixo).

M =

-1,9	-4	7,34	2	22	-1	9	234	17	2
3	5,5	6	9	0	6,78	4	1	12	3
5	-3,8	2	3	5	78	5	890	3	7
8	2	8	4	67	45	67	467	-9	9
0	1	4	8	87	13	1234	432	0	0
6	-4	9	9,8	54	-3	56	78	21	2,8
-3	3	0	7	35	90	8	65	67	0
0	-0,7	10	2	89	7	0	24	999	3,7
15	5	0	-39	34	4	1	-5	15	-15
-5	7	0	7,6	-98	2,5	-1	0	9	1

9) Escreva um algoritmo que preencha uma matriz 4 x 4 com números reais gerados randomicamente. O algoritmo deve apresentar o resultado da multiplicação da primeira matriz pela segunda. Exemplo:

A =

1	4	7	2
3	5	6	9
5	3	2	3
8	2	8	4

B =

0	2	0	3
8	1	7	0
4	0	6	1
1	0	4	9

A x B =

0+32+28+2	2+4+0+0	0+28+42+8	3+0+7+18
0+40+24+9	6+5+0+0	0+35+36+36	9+0+6+81
0+24+8+3	10+3+0+0	0+21+12+12	15+0+2+27
0+16+32+4	16+2+0+0	0+14+48+16	24+0+8+36

A x B =

62	6	78	28
73	11	107	96
35	13	45	44
52	18	78	68

O algoritmo deve:

a) Possuir um método "**public void preencher (double mat[4][4])**" para preencher a matriz com números gerados randomicamente.
b) Possuir um método "**public void imprimir (double mat[4][4])**" para imprimir matriz.
c) Possuir um método "**public void multiplicar (double mat1[4][4], double mat2[4][4], double result[4][4])**" responsável pela multiplicação das matrizes.

10) Escreva um algoritmo para venda de ingressos em uma sala de cinema. A sala de cinema deve ser representada como uma matriz 10 x 10 de inteiros. O algoritmo deve:

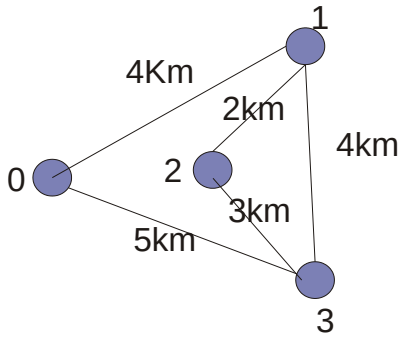
a) Possuir um método para iniciar a matriz com o valor 0 que indica lugares vazios.
b) Possuir um método para venda de lugares na sala, marcando o assento vendido com o valor 1.
c) Possuir um método que imprima a matriz.
d) Possuir um método que receba o valor do ingresso como parâmetro e retorne o total apurado numa seção.

11) Escreva um algoritmo que armazene em uma matriz a distância das estradas que ligam 4 cidades vizinhas. O algoritmo deve:

a) Possuir um método para inserir a distância entre duas cidades na matriz.
b) Possuir um método para contar quantas estradas ligam as cidades.
c) Possuir um método para imprimir a matriz.
d) Possuir um método que receba a identificação de duas cidades adjacentes como parâmetro e:
    i. Retornar a distância entre as cidades, se houver uma estrada entre elas.
    ii. Retornar zero, se não houver uma estrada ligando as duas cidades.

Exemplo:

Mapa:



Matriz:

	0	1	2	3
0	0	4	0	5
1	4	0	2	4
2	0	2	0	3
3	5	4	3	0

Centro Universitário UNA
Sistemas de Informação
Análise e Projeto de Algoritmos
Prática de Laboratório
Wesley Dias Maciel
2017/01

Prática 02

Aula: análise de complexidade e algoritmos de ordenação (método bolha).

Exercícios:

1) Escreva um algoritmo em Java que instancie um vetor de 10 posições. O algoritmo deve preencher as 5 primeiras posições do vetor com números randômicos. Crie um contador para contabilizar o número de valores inseridos no vetor. Ao final da execução, o algoritmo deve imprimir o tamanho do vetor seguido do valor do contador. Em seguida, repita o experimento com vetores de tamanho 100, 1.000, 10.000, 100.000 e 1.000.000. Construa um gráfico "Tamanho do Vetor (N)" X "Número de Operações (OP)" no Excel.

2) Escreva um algoritmo em Java que instancie um vetor de 10 posições. O algoritmo deve preencher todas as posições do vetor com números randômicos. Crie um contador para contabilizar o número de valores inseridos no vetor. Ao final da execução, o algoritmo deve imprimir o tamanho do vetor seguido do valor do contador. Em seguida, repita o experimento com vetores de tamanho 100, 1.000, 10.000, 100.000 e 1.000.000. Construa um gráfico "Tamanho do Vetor (N)" X "Número de Operações (OP)" no Excel.

3) Escreva um algoritmo em Java que instancie uma matriz N x N. O algoritmo deve preencher todas as posições da Matriz com números randômicos. Crie um contador para contabilizar o número de valores inseridos na Matriz. Ao final da execução, o algoritmo deve imprimir o valor de N seguido do valor do contador. Em seguida, repita o experimento com matrizes de tamanho 100, 1.000, 10.000, 100.000 e 1.000.000. Construa um gráfico "Tamanho do Vetor (N)" X "Número de Operações (OP)" no Excel.

4) Escreva um pacote em linguagem Java para ordenação de vetores. O pacote deve conter o algoritmo de ordenação pelo método bolha. Teste o algoritmo de ordenação implementado no pacote, usando vetores de tamanho (N):
    a. 10.
    b. 100.
    c. 1.000.
    d. 10.000.
    e. 100.000.

    Os elementos dos vetores devem ser gerados aleatoriamente. No Excel, gere 2 gráficos:

    a. Tamanho do vetor x Número de Comparações
    b. Tamanho do vetor x Número de Trocas

Centro Universitário UNA
Sistemas de Informação
Análise e Projeto de Algoritmos
Prática de Laboratório
Wesley Dias Maciel
2017/01

Cada gráfico deve apresentar o desempenho do algoritmo de ordenação para os vetores de tamanho N especificados (10, 100, 1.000, 10.000 e 100.000). Apresente a complexidade do algoritmo de ordenação no melhor caso, caso médio e pior caso. Indique se o algoritmo de ordenação é estável ou não estável. Indique também se o algoritmo de ordenação é adaptativo ou não adaptativo.

Exemplo:

```
package item;
public class Item {
  public int chave;

  public Item (int chave) {this.chave = chave;}

  public int getChave () {return chave;}

  public void setChave (int chave) {
    this.chave = chave;
  }
}
```

```
package ordena;
import item.Item;
import java.util.Random;
public abstract class Ordena {
  Item[] v;

  public Item[] getVetor () {return v;}

  public void setVetor (int n) {
    int i;
    Random rand = new Random ();
    this.v = new Item[n];
    for (i = 0; i < this.v.length; i++)
      this.v[i] = new Item (rand.nextInt (100));
  }

  public void imprime () {
    int i, n = this.v.length;
    for (i = 0; i < n; i++)
      System.out.print (this.v[i].chave + " ");
    System.out.println ();
  }
}
```

```
package ordena;
import item.Item;
public class Bolha extends Ordena {
  public Bolha () {}

  public void bolha () {
    int i, j, n = this.v.length;
    Item aux;

    for (i = 0; i < n - 1; i++)
      for (j = 1; j < n - i; j++)
        if (this.v[j].chave < this.v[j - 1].chave) {
          aux = this.v[j];
          this.v[j] = this.v[j - 1];
          this.v[j - 1] = aux;
        }
  }
}
```

```
package principal;
import ordena.Bolha;
import item.Item;
import java.util.Random;

public class Principal {
  public static void main (String[] args) {
    Principal o = new Principal ();
    o.ordenaBolha (5);
  }
  public void ordenaBolha (int n) {
    Bolha b = new Bolha ();
    b.setVetor (n);
    System.out.println ("Método da Bolha");
    System.out.println ("Vetor desordenado: ");
    b.imprime ();
    b.bolha ();
    System.out.println ("Vetor ordenado: ");
    b.imprime ();
  }
}
```

Centro Universitário UNA
Sistemas de Informação
Análise e Projeto de Algoritmos
Prática de Laboratório
Wesley Dias Maciel
2017/01

Prática 03

Aula: análise de complexidade e algoritmos de ordenação (método de seleção e de inserção).

Exercícios:

1) Em seu pacote para ordenação de vetores escrito em linguagem Java, acrescente os algoritmos de ordenação pelo método de seleção e de inserção. Teste os algoritmos de ordenação implementados no pacote, usando vetores de tamanho (N):
    f. 10.
    g. 100.
    h. 1.000.
    i. 10.000.
    j. 100.000.

   Os elementos dos vetores devem ser gerados aleatoriamente. No Excel, gere 2 gráficos para cada algoritmo de ordenação:

    c. Tamanho do vetor x Número de Comparações
    d. Tamanho do vetor x Número de Trocas

Cada gráfico deve apresentar o desempenho dos algoritmos de ordenação para os vetores de tamanho N especificados (10, 100, 1.000, 10.000 e 100.000). Apresente a complexidade dos algoritmos de ordenação no melhor caso, caso médio e pior caso. Indique se os algoritmos de ordenação são estáveis ou não estáveis. Indique também se os algoritmos de ordenação são adaptativos ou não adaptativos.

Algoritmo de ordenação pelo método de Seleção:

```
public void selecao () {
   int i, j, menor, n = this.v.length;
   Item aux;

   for (i = 0; i < n - 1; i++) {
       menor = i;
       for (j = i + 1; j < n; j++) {
           if (this.v[j].chave < this.v[menor].chave) {
               menor = j;
           }
       }
       aux = this.v[i];
       this.v[i] = this.v[menor];
       this.v[menor] = aux;
   }
}
```

Centro Universitário UNA
Sistemas de Informação
Análise e Projeto de Algoritmos
Prática de Laboratório
Wesley Dias Maciel
2017/01

Algoritmo de ordenação pelo método de Inserção:

```
public void insercao () {
    int i, j, n = this.v.length;
    Item aux;
    for (i = 1; i < n; i++) {
        aux = this.v[i];
        for (j = i - 1; (j >= 0) && (aux.chave < this.v[j].chave); j--) {
            this.v[j + 1] = this.v[j];
        }
        this.v[j + 1] = aux;
    }
}
```

Centro Universitário UNA
Sistemas de Informação
Análise e Projeto de Algoritmos
Prática de Laboratório
Wesley Dias Maciel
2017/01

Prática 04

Aula: análise de complexidade e algoritmos de ordenação (método Quicksort).

Exercícios:

1) Em seu pacote para ordenação de vetores escrito em linguagem Java, acrescente o algoritmo de ordenação pelo método Quicksort. Teste os algoritmos de ordenação implementados no pacote, usando vetores de tamanho (N):
    k. 10.
    l. 100.
    m. 1.000.
    n. 10.000.
    o. 100.000.

   Os elementos dos vetores devem ser gerados aleatoriamente. No Excel, gere 2 gráficos para cada algoritmo de ordenação:

    e. Tamanho do vetor x Número de Comparações
    f. Tamanho do vetor x Número de Trocas

Cada gráfico deve apresentar o desempenho dos algoritmos de ordenação para os vetores de tamanho N especificados (10, 100, 1.000, 10.000 e 100.000). Apresente a complexidade dos algoritmos de ordenação no melhor caso, caso médio e pior caso. Indique se os algoritmos de ordenação são estáveis ou não estáveis. Indique também se os algoritmos de ordenação são adaptativos ou não adaptativos.

Algoritmo de ordenação pelo método Quicksort:

Centro Universitário UNA
Sistemas de Informação
Análise e Projeto de Algoritmos
Prática de Laboratório
Wesley Dias Maciel
2017/01

```
package ordena;

import item.Item;

public class Quicksort  extends Ordena {
  public int i, j;

  public Quicksort () {
  }

  public Quicksort (Item[] vetor) {
    this.v = vetor;
  }

  public void quicksort () {
    this.ordena (0, this.v.length - 1);
  }

  public void ordena (int e, int d) {
    this.particao (e, d);
    if (e < this.j)
      this.ordena (e, this.j);
    if (this.i < d)
      this.ordena (this.i, d);
  }

  public void particao (int e, int d) {
    Item x, aux;
    this.i = e;
    this.j = d;
    x = this.v[(this.i + this.j) / 2];
    do {
      while (this.v[this.i].chave < x.chave)
        this.i++;
      while (this.v[this.j].chave > x.chave)
        this.j--;
      if (this.i <= this.j) {
        aux = this.v[this.i];
        this.v[this.i] = this.v[this.j];
        this.v[this.j] = aux;

        this.i++;
        this.j--;
      }
    } while (this.i <= this.j);
  }
}
```

Centro Universitário UNA
Sistemas de Informação
Análise e Projeto de Algoritmos
Prática de Laboratório
Wesley Dias Maciel
2017/01

Prática 05

Aula: análise de complexidade e algoritmos de busca (sequencial, sequencial com sentinela e binária).

Exercícios:

1) Em Java, crie um novo projeto para a prática 05. Esse projeto deve possuir 2 pacotes. Um pacote deve conter as classes que implementam os algoritmos de busca. O outro pacote deve conter a classe responsável pela execução do projeto, classe que contém o método "main".
2) Em seu pacote para os algoritmos de busca, crie uma classe para cada uma das estratégias de busca: sequencial, sequencial com sentinela e binária.
3) Em seu pacote para classe de execução do projeto, crie um algoritmo para testar cada um dos algoritmos de busca implementados, usando vetores de tamanho (N):
    a) 10.
    b) 100.
    c) 1.000.
    d) 10.000.
    e) 100.000.

Os elementos dos vetores devem ser gerados aleatoriamente. No Excel, gere 1 gráfico para cada algoritmo de busca:

a) Tamanho do vetor x Número de Comparações

Cada gráfico deve apresentar o desempenho dos algoritmos de busca para os vetores de tamanho N especificados (10, 100, 1.000, 10.000 e 100.000). Apresente a complexidade dos algoritmos de busca no melhor caso, caso médio e pior caso.

Algoritmo de busca sequencial:

```
int search (int a[], int v, int l, int r) {
   for (int i = l; i <= r; i++)
      if (v == a[i])
         return i;
   return -1;
}
```

OBS:
a: vetor.
v: valor procurado.
l: left (esquerda).
r: right (direira).

Centro Universitário UNA
Sistemas de Informação
Análise e Projeto de Algoritmos
Prática de Laboratório
Wesley Dias Maciel
2017/01

Algoritmo de busca sequencial com sentinela:

```
int search (int a[], int v, int l, int r) {
   int i, n = r + 1;
   a[n] = v;
   for (i = l; v != a[i]; i++);
   if (i < n)
      return (i);  /*Chave encontrada!*/
   else
      return (-1); /*Sentinela encontrada.*/
}
```

OBS:
a: vetor.
v: valor procurado.
l: left (esquerda).
r: right (direira).

Algoritmo de busca binária:

```
int search (int a[], int v, int l, int r) {
    while (l <= r) {
        int m = (l + r) / 2;
        if (v == a[m])
            return m;
        if (v < a[m])
            r = m - 1;
        else
            l = m + 1;
    }
    return -1;
}
```

OBS:
a: vetor.
v: valor procurado.
l: left (esquerda).
r: right (direira).

Centro Universitário UNA
Sistemas de Informação
Análise e Projeto de Algoritmos
Prática de Laboratório
Wesley Dias Maciel
2017/01

Prática 06

Aula: análise de complexidade e algoritmos de ordenação (método Heapsort).

Exercícios:

1) Em seu pacote para ordenação de vetores escrito em linguagem Java, acrescente o algoritmo de ordenação pelo método Heapsort. Teste os algoritmos de ordenação implementados no pacote, usando vetores de tamanho (N):
    a. 10.
    b. 100.
    c. 1.000.
    d. 10.000.
    e. 100.000.

   Os elementos dos vetores devem ser gerados aleatoriamente. No Excel, gere 2 gráficos para cada algoritmo de ordenação:

    a. Tamanho do vetor x Número de Comparações
    b. Tamanho do vetor x Número de Trocas

Cada gráfico deve apresentar o desempenho dos algoritmos de ordenação para os vetores de tamanho N especificados (10, 100, 1.000, 10.000 e 100.000). Apresente a complexidade dos algoritmos de ordenação no melhor caso, caso médio e pior caso. Indique se os algoritmos de ordenação são estáveis ou não estáveis. Indique também se os algoritmos de ordenação são adaptativos ou não adaptativos.

Algoritmo de ordenação pelo método Heapsort em pseudocódigo:

```
heapsort (A) {
    build-max-heap (A)
    for (i = A.comprimento; i > 0; i--)
        troca (A[0], A[i])
        A.tamanho-do-heap = A.tamanho-do-heap - 1
        max-heapify (A, 0)
}
```

Centro Universitário UNA
Sistemas de Informação
Análise e Projeto de Algoritmos
Prática de Laboratório
Wesley Dias Maciel
2017/01

```
build-max-heap (A) {
    A.tamanho-do-heap = A.comprimento
    for (i = piso (A.tamanho-do-heap / 2); i >= 0; i--)
        max-heapify (A, i)
}
```

```
max-heapify (A, i) {
    e = esquerda (i)
    d = direita (i)
    if ((e < A.tamanho-do-heap) e (A[e] > A[i])) maior = e else maior = i
    if ((d < A.tamanho-do-heap) e (A[d] > A[maior])) maior = d
    if (maior != i) {
        troca (A[i], A[maior])
        max-heapify (A, maior)
    }
}
```

Centro Universitário UNA
Sistemas de Informação
Análise e Projeto de Algoritmos
Prática de Laboratório
Wesley Dias Maciel
2017/01

Prática 07

Aula: análise de complexidade e tabela hash.

Exercício:

1) Juntamente com esta prática, será disponibilizado um projeto Netbeans que demonstra a utilização de tabelas hash em Java. O projeto cria um cadastro de pessoas e suas respectivas profissões. As pessoas são cadastradas numa tabela hash. Cada entrada da tabela hash armazena uma profissão e possui uma referência para uma lista encadeada que armazena o nome das pessoas referentes à profissão. O nome e profissão de cada pessoa é lido de 2 arquivos locais do projeto. Um arquivo contém o nome das pessoas. O outro arquivo contém a profissão das respectivas pessoas. Você deve:
    a. Criar um método para apresentar o número de colisões ocorridas ao cadastrar as pessoas na tabela hash.
    b. Criar um método para apresentar o tamanho de cada lista encadeada da tabela hash.
    c. Criar um método para apresentar o tamanho médio das listas encadeadas da tabela hash.
    d. Criar um método que receba uma profissão como parâmetro e que apresente todas as pessoas relativas a essa profissão.
    e. Criar um método que receba o nome de uma pessoa como parâmetro e que apresente a profissão dessa pessoa. Se o nome não for encontrado na tabela hash, o método deve informar que a pessoa não foi cadastrada. O método deve apresentar o número de comparações realizadas para encontrar a pessoa na tabela hash.