Plano de Aula

1 Código e nome da disciplina

ARA0174 ALGORITMOS E COMPLEXIDADE

2 Semana/Tema

Semana 1: Tema - 1. ANÁLISE DE ALGORITMO

3 Objetivos

Desenvolver algoritmos com funções e passagens de parâmetros, utilizando linguagem de programação estruturada (C/C++).

4 Tópicos

1.1 ALGORITMOS: FUNÇÕES E PASSAGEM DE PARÂMETROS

5 Procedimentos de ensino-aprendizagem

Nesta aula, serão apresentados conceitos relacionados a implementação de funções e passagens de parâmetros nas funções, utilizando uma linguagem de programação imperativa estruturada (C/C++ ou Java). Para isso, segue sugestão de roteiro:

- Situação-problema: Apresentar situações algorítmicas em que seja necessário a utilização de funções (como situações em que se deseja evitar repetição de código). A utilização de funções visa modularizar um programa, o que é muito comum em programação estruturada. Desta forma podemos dividir um programa em várias partes, no qual cada função realiza uma tarefa bem definida.

- Metodologia: Aula expositiva, iniciando com os conceitos de função e de como é realizado a passagem de parâmetro entre funções. Depois, apresentar as principais ferramentas (IDE) de implementação em linguagem estruturada e escolher aquela que melhor se adequa ao ambiente da turma.

- Atividade verificadora de Aprendizagem: O professor deve mediar as implementações realizadas e verificar se os alunos realizaram a resolução das situações algorítmicas, apresentadas na situação-problema, em que é necessário o uso de funções.

6 Recursos didáticos

Laboratório de Programação de Computadores, equipado com quadro branco, computadores com acesso à Internet, além de Datashow conectado ao computador do docente.

7 Leitura específica

[1] Jayme Luiz Szwarcfiter, Lilian markezon. Estruturas de Dados e Seus Algoritmos, 3rd Edition. [BV:MB]. 3a Edição. Rio de Janeiro: LTC. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788521629955/cfi/6/8!/4/2/4@0:0

8 Aprenda +

Aprendizagem Autônoma Aura - AAA

Questão 1: Implementar um algoritmo em linguagem estruturada (C/C++) que utilize função por passagem de parâmetro para calcular o quadrado de um número real.

Questão 2: Sobre **funções** e **passagens de parâmetros** por referências em linguagem estruturada C, sejam as seguintes afirmações:

I - Para utilizar uma função após o programa principal, deve-se primeiramente declarar a função no início do código, chamando esse processo de protótipo de função;
II - Os parâmetros da função na sua declaração são chamados parâmetros formais. Na chamada da função os parâmetros são chamados parâmetros atuais/reais;
III - Os nomes dos parâmetros que são passados para dentro de uma função devem ter o mesmo nome quando ?chegam? dentro da função.

Assinale a opção que contém as afirmativas verdadeiras:
A) I e II.
B) II e III.
C) I e III.
D) apenas o item II.
E) I, II e III.

Plano de Aula

1 Código e nome da disciplina

ARA0174 ALGORITMOS E COMPLEXIDADE

2 Semana/Tema

Semana 2: Tema - 1. ANÁLISE DE ALGORITMO

3 Objetivos

Implementar estruturas de dados homogêneas, heterogêneas e ponteiros, utilizando linguagem de programação imperativa e estruturada (C/C++).

4 Tópicos

1.2 ESTRUTURA DE DADOS: HOMOGÊNEAS, HETEROGÊNEAS E PONTEIRO

5 Procedimentos de ensino-aprendizagem

Nesta aula, serão apresentados conceitos relacionados a implementação de estruturas de dados homogêneas, heterogêneas e ponteiros, utilizando uma linguagem de programação imperativa estruturada (C/C++). Para isso, segue sugestão de roteiro:

- Situação-problema: Sugerir situações algorítmicas em que seja necessário a utilização de vetores, matrizes e registros. Elaborar algoritmos em que o uso de registros é necessário.

- Metodologia: Aula expositiva, iniciando com os conceitos de estrutura de dados homogêneas e heterogêneas e de como é realizado a implementação por ponteiros. Depois, apresentar as principais ferramentas (IDE) de implementação em linguagem estruturada e escolher aquela que melhor se adequa ao ambiente da turma.

- Atividade verificadora de Aprendizagem: O professor deve mediar as implementações realizadas e verificar se os alunos realizaram a resolução das situações algorítmicas, apresentadas na situação-problema, em que é necessário o uso de vetores unidimensionais e bidimensionais.

6 Recursos didáticos

Laboratório de Programação de Computadores, equipado com quadro branco, computadores com acesso à Internet, além de Datashow conectado ao computador do docente.

7 Leitura específica

[1] Jayme Luiz Szwarcfiter, Lilian markezon. Estruturas de Dados e Seus Algoritmos, 3rd Edition. [BV:MB]. 3a Edição. Rio de Janeiro: LTC. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788521629955/cfi/6/8!/4/2/4@0:0

8 Aprenda +

Atividade Autônoma Aura - AAA

Questão 1:
O que é impresso após a execução do seguinte programa em linguagem C:

```
#include <stdio.h>

struct T{

int x;

int y;

};

typedef struct T T;

void f1(T *a);

void f2(int *b);

int main(){

T a, b, *c, *d;

c = &a;

a.x = 2;

a.y = 4;

b.x = 2;

b.y = 2;

 d = c;

f1(d);

b = *d;

printf("x: %d \n",b.x);

printf("y: %d \n",b.y);

}

void f1(T *a){
```

```
f2(&(a->x));

f2(&(a->y));

}

void f2(int *b){

 *b = 2*(*b);

}
```

Questão 2:

Sobre vetores e as Structs em linguagem C, sejam as seguintes afirmações:

I - Cada registro tem um endereço na memória do computador. É muito comum usar um ponteiro para armazenar o endereço de um registro, por isso, cada registro deve ser apenas de um tipo primitivo, somente;
II - Um registro é um pacote de variáveis de tipos diferentes. Cada variável é um campo do registro.
III - Um vetor é capaz de armazenar diversos valores, com a restrição de que todos sejam de um mesmo tipo de dados.

Está correto:
A) apenas o item I.
B) apenas os itens I e II.
C) apenas os itens II e III.
D) apenas os itens I e III.
E) os itens I, II e III.

Plano de Aula

1 Código e nome da disciplina

ARA0174 ALGORITMOS E COMPLEXIDADE

2 Semana/Tema

Semana 3: Tema - 1. ANÁLISE DE ALGORITMO

3 Objetivos

Descrever os principais conceitos da Notação Big O para classificar algoritmos em relação as mudanças de desempenho quanto ao tamanho da entrada.

4 Tópicos

1.3 ANÁLISE DE ALGORITMOS: CONCEITOS, NOTAÇÃO O E FUNÇÃO O

5 Procedimentos de ensino-aprendizagem

Nesta aula, serão apresentados conceitos e as principais definições relacionados a complexidade computacional e a Notação Big O. A notação Big O é um tópico importante e sua importância universal origina-se do fato de descrever a eficiência do código escrito em qualquer linguagem de programação. Para isso, segue sugestão de roteiro:

- Situação-problema: Apresentar problemas computacionais cujo a aplicação da notação O seja necessária. A notação Big O pode ser utilizada para descrever a complexidade de uma seção de código em termos de tempo de execução e espaço. Assim, a complexidade do tempo do Big O descreve o tempo de execução no pior cenário. Já a complexidade do espaço Big O, descreve quanta memória é necessária para executar uma seção de código no pior cenário. Segundo Cormen [3], quando observamos tamanhos de entrada grandes o suficiente para tornar relevante apenas a ordem de crescimento do tempo de execução, estamos estudando a eficiência assintótica dos algoritmos. Descrever situações em que a avaliação da complexidade de tempo e de espaço possam ser avaliadas, com um loop for que copia uma matriz, que precisará de muito mais memória para ser executado do que um loop que simplesmente modifica uma matriz existente.

- Metodologia: Aula expositiva, iniciando com os conceitos e as principais definições relacionados a complexidade computacional e a Notação Big O. Leitura dos conceitos e exemplos apresentados em [2] e [3].

- Atividade verificadora de Aprendizagem: O professor deve mediar os exemplos de problemas computacionais apresentados na situação-problema e verificar se os alunos identificaram a classes de complexidade de cada um deles, de acordo com os conceitos apresentados.

6 Recursos didáticos

Laboratório de Programação de Computadores, equipado com quadro branco, computadores com acesso à Internet, além de Datashow conectado ao computador do docente.

7 Leitura específica

[2] Nivio Ziviane. Projeto de algoritmos: com implementações em Java e C++. [BV:MB]. São Paulo: Cengage Learning. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788522108213/cfi/2!/4/4@0.00:49.1

[3] Thomas H. Cormen; Charles E. Leiserson; Ronald L. Rivest; Stein, Clifford. Algoritmos: Teoria e Prática [BV:PE]. Rio de Janeiro: Elsevier Editora Ltda, 2012.
Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595158092/cfi/6/4!/4/2/2@0:0

8 Aprenda +

Atividade Autônoma Aura - AAA

Questão 1:
Do ponto de vista de projeção de algoritmos, quais são as questões mais importantes à serem consideradas na escolha de um algoritmo?

Está correto:
A) Corretude, eficiência, robustez e reusabilidade
B) Corretude, eficiência, robustez e recursividade
C) Corretude, eficiência, robustez e versatilidade
D) Corretude, independência, robustez e autenticidade
E) Corretude, independência, robustez e recursividade

Questão 2:
Considere os seguintes algoritmos e suas complexidades na notação Big O:
- Algoritmo A: O(log n)
- Algoritmo B: O(n^2)
- Algoritmo C: O(n log n)

Considerando-se o pior caso de execução destes algoritmos, é correto afirmar que o algoritmo:
A) A é o menos eficiente.
B) C é o menos eficiente.
C) A não é o mais eficiente nem o menos eficiente.
D) B é o menos eficiente.
E) C é o mais eficiente.

Plano de Aula

1 Código e nome da disciplina

ARA0174 ALGORITMOS E COMPLEXIDADE

2 Semana/Tema

Semana 4: Tema - 1. ANÁLISE DE ALGORITMO

3 Objetivos

Desenvolver algoritmos em linguagem de programação imperativa e estruturada para analisar o comportamento de cada código e classificar cada algoritmo de acordo com o seu desempenho relativo ao tamanho da entrada

4 Tópicos

1.4 PRÁTICA DE ANÁLISE DE ALGORITMOS

5 Procedimentos de ensino-aprendizagem

Nesta aula, serão colocados em prática os problemas computacionais apresentados na aula anterior, bem como vetores, matrizes, algoritmos recursivos etc. Para isso, segue sugestão de roteiro:

- Situação-problema: Apresentar problemas computacionais nas mais diversas classes de complexidade computacionais, como algoritmos na classe constante O(1), linear O(n), logarítmica O(log n) etc.

- Metodologia: Aula expositiva e prática, com implementação dos problemas computacionais apresentados na situação problema.

- Atividade verificadora de Aprendizagem: O professor deve mediar os exemplos de problemas computacionais apresentados na situação-problema.

6 Recursos didáticos

Laboratório de Programação de Computadores, equipado com quadro branco, computadores com acesso à Internet, além de Datashow conectado ao computador do docente.

7 Leitura específica

[1] Jayme Luiz Szwarcfiter, Lilian markezon. Estruturas de Dados e Seus Algoritmos, 3rd Edition. [BV:MB]. 3a Edição. Rio de Janeiro: LTC. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788521629955/cfi/6/8!/4/2/4@0:0

[2] Nivio Ziviane. Projeto de algoritmos: com implementações em Java e C++. [BV:MB]. São Paulo: Cengage Learning. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788522108213/cfi/2!/4/4@0.00:49.1

[3] Thomas H. Cormen; Charles E. Leiserson; Ronald L. Rivest; Stein, Clifford. Algoritmos: Teoria e Prática [BV:PE]. Rio de Janeiro: Elsevier Editora Ltda, 2012. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595158092/cfi/6/4!/4/2/2@0:0

8 Aprenda +

Atividade Autônoma Aura - AAA

Questão 1:
Assinale a alternativa correta acerca da notação O:

A) A definição da notação O segue-se que: Seja g(n), para O(g(n)) = {t(n): existem as constantes positivas c e n_0 tais que 0 = t(n) = c.g(n), para todo n = n_0}.
B) Se temos que T(n)=n^2+n, então a complexidade de pior caso de T(n) é igual a n.
C) A definição da notação O segue-se que: Seja g(n), para O((g(n))= {t(n): existem as constantes positivas c_1,c_2 e n_0 tais que 0=c_1.g(n)=t(n)=c_2.g(n), para todo n=n_0}.
D) Um algoritmo de ordenação qualquer que tem a complexidade de cada uma das etapas do algoritmo n+n^3+n, significa que a complexidade assintótica final desse algoritmo é O(n^2).
E) A notação O sempre irá considerar situações dos algoritmos de melhor caso, haja vista que são algoritmos melhores de serem tratados e analisados.

Questão 2:
Seja o seguinte trecho de um algoritmo em Linguagem C. Implementar e fazer a análise da quantidade de passos executados pelas suas estruturas de repetição:

```
int func_soma(int n)
{
    for(i =0; i < n; i++ ){
        for(j =0; j < n*n; j++ ){
            sum++;
        }

    }
    return sum;
}
```

Plano de Aula

1 Código e nome da disciplina

ARA0174 ALGORITMOS E COMPLEXIDADE

2 Semana/Tema

Semana 5: Tema - 2. RECURSIVIDADE

3 Objetivos

Descrever os conceitos de recursividade e implementação de algoritmos de natureza recursiva, utilizando linguagem de programação imperativa estruturada.

4 Tópicos

2.1 DEFINIÇÕES RECURSIVAS
2.2 COMO IMPLEMENTAR RECURSIVIDADE

5 Procedimentos de ensino-aprendizagem

Nesta aula, serão apresentadas as definições e conceitos relativos a recursividade. Para isso, segue sugestão de roteiro:

- Situação-problema: Apresentar problemas computacionais nos quais sejam necessárias a aplicação de recursividade. A ideia básica de um algoritmo recursivo consiste em diminuir sucessivamente o problema em um problema menor ou mais simples, até que o tamanho ou a simplicidade do problema reduzido permita resolvê-lo de forma direta, sem recorrer a si mesmo. Problemas que possuem essa natureza são: calcular o fatorial, a torre de Hanoi, transformação de decimal em binário, etc.

- Metodologia: Aula expositiva e prática, com implementação dos problemas computacionais apresentados na situação problema.

- Atividade verificadora de Aprendizagem: O professor deve mediar a implementação dos exemplos de problemas computacionais de natureza recursiva apresentados na situação-problema.

6 Recursos didáticos

Laboratório de Programação de Computadores, equipado com quadro branco, computadores com acesso à Internet, além de Datashow conectado ao computador do docente.

7 Leitura específica

[1] Jayme Luiz Szwarcfiter, Lilian markezon. Estruturas de Dados e Seus Algoritmos, 3rd Edition. [BV:MB]. 3a Edição. Rio de Janeiro: LTC. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788521629955/cfi/6/8!/4/2/4@0:0

[2] Nivio Ziviane. Projeto de algoritmos: com implementações em Java e C++. [BV:MB]. São Paulo: Cengage Learning. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788522108213/cfi/2!/4/4@0.00:49.1

[3] Thomas H. Cormen; Charles E. Leiserson; Ronald L. Rivest; Stein, Clifford. Algoritmos: Teoria e Prática [BV:PE]. Rio de Janeiro: Elsevier Editora Ltda, 2012. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595158092/cfi/6/4!/4/2/2@0:0

8 Aprenda +

Atividade Autônoma Aura - AAA

Questão 1:
Analise as sentenças abaixo sobre recursividade.
1. Um programa recursivo é um programa que chama a si mesmo.
2. Uma função recursiva é definida em termos dela mesma.
3. A recursividade pode ser classificada como direta ou indireta.
4. Para cada chamada de uma função recursiva os parâmetros e as variáveis locais são empilhados na pilha de execução.

Assinale as alternativas que contém a sequência verdadeira. Considere **V - Verdadeira** e **F - Falsa**.
A) 1-F, 2-V, 3-V, 4-V.
B) 1-V, 2-V, 3-V, 4-V.
C) 1-V, 2-F, 3-V, 4-F.
D) 1-F, 2-F, 3-V, 4-V.
E) 1-F, 2-F, 3-F, 4-F.

Questão 2:
Assinale a opção correta. O que faz a função F definida a seguir?

```
int F (int n) {
if (n == 0)
return(0)
else
return(n + F(n-1))
}
```

Alternativas:
A) Soma os inteiros de n até 0, inclusive.
B) Soma os inteiros de 0 até n-1, inclusive
C) Não compila, pois não tem caso base
D) Calcula o fatorial de um inteiro n passado como parâmetro
E) Soma os inteiros de 1 até n-1, inclusive.

Plano de Aula

1 Código e nome da disciplina

ARA0174 ALGORITMOS E COMPLEXIDADE

2 Semana/Tema

Semana 6: Tema - 2. RECURSIVIDADE

3 Objetivos

Implementação de algoritmos recursivos e não recursivos, utilizando linguagem de programação imperativa estruturada (C/C++) a fim de realizar uma análise comparativa dentre esses dois tipos de implementação.

4 Tópicos

2.3 DESENVOLVENDO ALGORITMOS COM RECURSIVIDADE
2.4 QUANDO NÃO USAR RECURSIVIDADE

5 Procedimentos de ensino-aprendizagem

Nesta aula, serão implementados algoritmos recursivos e algoritmos não recursivos, apresentando as vantagens e desvantagens de cada um. Para isso, segue sugestão de roteiro:

- Situação-problema: Apresentar problemas computacionais nos quais sejam necessárias a aplicação de recursividade e também de não recursividade. Para todo algoritmo recursivo existe um outro correspondente iterativo (não recursivo), que executa a mesma tarefa. Pode ser apresentado um exemplo de um problema em que o uso de memória (espaço) tem custo elevado, portanto a forma interativa do algoritmo é mais interessante sob o aspecto de economia de pilha de memória.

- Metodologia: Aula expositiva e prática, com implementação dos problemas computacionais apresentados na situação problema.

- Atividade verificadora de Aprendizagem: O professor deve mediar a implementação dos exemplos de problemas computacionais de natureza recursiva e não recursiva apresentados na situação-problema.

6 Recursos didáticos

Laboratório de Programação de Computadores, equipado com quadro branco, computadores com acesso à Internet, além de Datashow conectado ao computador do docente.

7 Leitura específica

[1] Jayme Luiz Szwarcfiter, Lilian markezon. Estruturas de Dados e Seus Algoritmos, 3rd Edition. [BV:MB]. 3a Edição. Rio de Janeiro: LTC. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788521629955/cfi/6/8!/4/2/4@0:0

[2] Nivio Ziviane. Projeto de algoritmos: com implementações em Java e C++. [BV:MB]. São Paulo: Cengage Learning. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788522108213/cfi/2!/4/4@0.00:49.1

[3] Thomas H. Cormen; Charles E. Leiserson; Ronald L. Rivest; Stein, Clifford. Algoritmos: Teoria e Prática [BV:PE]. Rio de Janeiro: Elsevier Editora Ltda, 2012. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595158092/cfi/6/4!/4/2/2@0:0

8 Aprenda +

Atividade Autônoma Aura - AAA

Questão 1: Implemente uma função recursiva que receba dois parâmetros inteiros (x e y) e calcule a potenciação pelo método dos produtos sucessivos (x^y). Ex.: 2^3 = 2 * 2 * 2.

Questão 2: Implemente uma função recursiva e não recursiva do algoritmo que calcula o fatorial de um número.

Plano de Aula

1 Código e nome da disciplina

ARA0174 ALGORITMOS E COMPLEXIDADE

2 Semana/Tema

Semana 7: Tema - 3. ALGORITMOS DE ORDENAÇÃO AVANÇADOS

3 Objetivos

Descrever e Implementar os principais algoritmos de ordenação elementares, bem como algoritmos de intercalação.

4 Tópicos

3.1 ANÁLISE DOS ALGORITMOS DE ORDENAÇÃO ELEMENTARES
3.2 ORDENAÇÃO RÁPIDA (QUICKSORT)

5 Procedimentos de ensino-aprendizagem

Nesta aula, serão apresentados e implementados os principais algoritmos de ordenação, tais como bubble sort, insertion sort e merge sort (intercalação). Para isso, segue sugestão de roteiro:

- Situação-problema: Suponha que uma empresa de tecnologia esteja interessada em contratar um programador que tenha conhecimento de como implementar e escolher qual (ou quais) o melhor algoritmo de ordenação existente na literatura. A leitura e entendimento dos algoritmos em [3] é recomendado.

- Metodologia: Apresentar (e rastrear) os principais algoritmos de ordenação, como bubble sort, insertion sort e merge sort e sugerir que façam uma análise empírica dos algoritmos de ordenação implementados.

- Atividade verificadora de Aprendizagem: O professor deve mediar a implementação dos algoritmos de ordenação vistos e que os alunos apresentem os resultados da análise empírica dos algoritmos.

6 Recursos didáticos

Laboratório de Programação de Computadores, equipado com quadro branco, computadores com acesso à Internet, além de Datashow conectado ao computador do docente.

7 Leitura específica

[3] Thomas H. Cormen; Charles E. Leiserson; Ronald L. Rivest; Stein, Clifford. Algoritmos: Teoria e Prática [BV:PE]. Rio de Janeiro: Elsevier Editora Ltda, 2012. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595158092/cfi/6/4!/4/2/2@0:0

8 Aprenda +

Atividade Autônoma Aura - AAA

Questão 1: Sejam as seguintes complexidades, O(n^2) e O (n log n). Marque a correlação correta sobre a complexidade de pior caso dos algoritmos de ordenação:

I - InsertionSort
II - BubbleSort
III - QuickSort
IV - MergeSort

Está(ão) correto(s):
A) I - O(n^2) / II - O (n log n) / III - O(n^2) / IV - O (n log n)
B) I - O (n log n) / II - O(n^2) / III - O (n log n) / IV - O (n log n)
C) I - O(n^2) / II - O(n^2) / III - O(n^2) / IV - O (n log n)
D) I - O(n^2) / II - O(n^2) / III - O(n^2) / IV - O(n^2)
E) I - O(n^2) / II - O(n^2) / III - O (n log n) / IV - O (n log n)

Questão 2: Implementar o algoritmo de ordenação *MergeSort* e realizar a sua análise de complexidade.

Plano de Aula

1 Código e nome da disciplina

ARA0174 ALGORITMOS E COMPLEXIDADE

2 Semana/Tema

Semana 8: Tema - 3. ALGORITMOS DE ORDENAÇÃO AVANÇADOS

3 Objetivos

Descrever e Implementar os principais algoritmos de ordenação rápida e de seleção.

4 Tópicos

3.1 ORDENAÇÃO POR INTERCALAÇÃO (MERGESORT)
3.4 ORDENAÇÃO SHELLSORT

5 Procedimentos de ensino-aprendizagem

Nesta aula, serão apresentados e implementados os principais algoritmos de ordenação rápida e de seleção, tais como Quicksort e Shellsort (seleção). Para isso, segue sugestão de roteiro:

- Situação-problema: Suponha que uma empresa de tecnologia esteja interessada em contratar um programador que tenha conhecimento de como implementar e escolher qual (ou quais) o melhor algoritmo de ordenação existente na literatura. A leitura e entendimento dos algoritmos de ordenação em [3] é recomendado.

- Metodologia: Apresentar (e rastrear) os principais algoritmos de ordenação, como Quicksort e Shellsort e sugerir que façam uma análise empírica dos algoritmos de ordenação implementados.

- Atividade verificadora de Aprendizagem: O professor deve mediar a implementação dos algoritmos de ordenação vistos e que os alunos apresentem os resultados da análise empírica dos algoritmos.

6 Recursos didáticos

Laboratório de Programação de Computadores, equipado com quadro branco, computadores com acesso à Internet, além de Datashow conectado ao computador do docente.

7 Leitura específica

[3] Thomas H. Cormen; Charles E. Leiserson; Ronald L. Rivest; Stein, Clifford. Algoritmos: Teoria e Prática [BV:PE]. Rio de Janeiro: Elsevier Editora Ltda, 2012. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595158092/cfi/6/4!/4/2/2@0:0

8 Aprenda +

Atividade Autônoma Aura - AAA

Questão 1: Implementar o algoritmo de ordenação *QuickSort* e realizar a sua análise de complexidade.

Questão 2:Assinale com X as alternativas corretas acerca da notação O e algoritmos de ordenação:

É correto o que se afirma em:
( ) O algoritmo de ordenação *BubbleSort* é considerado um algoritmo estável, pois ao decorrer das trocas das chaves iguais, os mesmos não mudam de posição.
( ) O algoritmo de ordenação *MergeSort* tem complexidade de pior caso O(n log?n).
( ) A notação O sempre irá considerar situações de pior caso nos algoritmos de ordenação, somente.
( ) O algoritmo de ordenação *InsertionSort* tem complexidade de pior caso O(n^2).
( ) O algoritmo de ordenação *ShellSort* é considerado um algoritmo estável.

Plano de Aula

1 Código e nome da disciplina

ARA0174 ALGORITMOS E COMPLEXIDADE

2 Semana/Tema

Semana 9: Tema - 4. ALGORITMOS EM ÁRVORES BINÁRIA E ÁRVORE AVL

3 Objetivos

Implementar as principais operações em Árvore de Pesquisa Binária e AVL e realizar a análise de complexidade de cada um deles.

4 Tópicos

4.1 ARVORE BINÁRIA DE BUSCA: BUSCA, INSERÇÃO E REMOÇÃO COM ANÁLISE DE COMPLEXIDADE

5 Procedimentos de ensino-aprendizagem

Nesta aula, serão apresentadas as principais operações realizadas em Árvore de Pesquisa Binária (Binary Search Tree) e seus principais algoritmos e análise de complexidade de cada um. Para isso, segue sugestão de roteiro:

- Situação-problema: Suponha que uma empresa de tecnologia trabalhe com diversas aplicações que necessitam que se represente um conjunto de objetos e suas relações hierárquicas, tais como diagrama hierárquico de uma organização, hierarquização de arquivos, busca de dados armazenados no computador. Apresentar as principais operações de árvore binária e de como ela pode representar/modelar bem esses problemas citados.

- Metodologia: Apresentar (e rastrear) a implementação árvore de pesquisa binária, bem como suas principais operações (inserção, busca, remoção) e sua análise de complexidade.

- Atividade verificadora de Aprendizagem: O professor deve mediar a implementação dos algoritmos de operações de árvore de pesquisa binária.

6 Recursos didáticos

Laboratório de Programação de Computadores, equipado com quadro branco, computadores com acesso à Internet, além de Datashow conectado ao computador do docente.

7 Leitura específica

[1] Jayme Luiz Szwarcfiter, Lilian markezon. Estruturas de Dados e Seus Algoritmos, 3rd Edition. [BV:MB]. 3a Edição. Rio de Janeiro: LTC. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788521629955/cfi/6/8!/4/2/4@0:0

8 Aprenda +

Atividade Autônoma Aura - AAA

Questão 1:Seja o algoritmo de **inserção** em Árvore de Pesquisa Binária, vistos em sala de aula. Dê a complexidade de tempo na notação quando a árvore está com seus nós **balanceados** e **desbalanceados** e **justifique**.

Questão 2:Implemente em linguagem estruturada os algoritmos recursivos de busca e inserção em árvores binárias.

Plano de Aula

1 Código e nome da disciplina

ARA0174 ALGORITMOS E COMPLEXIDADE

2 Semana/Tema

Semana 10: Tema - 4. ALGORITMOS EM ÁRVORES BINÁRIA E ÁRVORE AVL

3 Objetivos

Descrever os conceitos de Árvore de Pesquisa Binária (Binary Search Tree) e seus principais algoritmos de visita (percurso) e realizar a análise de complexidade de cada um deles.

4 Tópicos

4.2 PERCURSO EM ÁRVORES BINÁRIA: ALGORITMOS DOS PERCURSOS EM ORDEM, PÓS-ORDEM E PRÉ-ORDEM COM RECURSIVIDADE E ANÁLISE DE COMPLEXIDADE

5 Procedimentos de ensino-aprendizagem

Nesta aula, serão apresentados os principais conceitos de Árvore Binária (Binary Tree) e seus principais algoritmos de percurso. Para isso, segue sugestão de roteiro:

- Situação-problema: Suponha que uma empresa de tecnologia trabalhe com diversas aplicações que necessitam que se represente um conjunto de objetos e suas relações hierárquicas, tais como diagrama hierárquico de uma organização, hierarquização de arquivos, busca de dados armazenados no computador. Apresentar o conceito de árvore de pesquisa binária e de como ela pode representar/modelar bem esses problemas citados.

- Metodologia: Apresentar (e rastrear) a implementação árvore de pesquisa binária, bem como seus principais algoritmos de percurso sistemáticos (recursivos) e sua análise de complexidade.

- Atividade verificadora de Aprendizagem: O professor deve mediar a implementação dos algoritmos de árvore binária.

6 Recursos didáticos

Laboratório de Programação de Computadores, equipado com quadro branco, computadores com acesso à Internet, além de Datashow conectado ao computador do docente.

7 Leitura específica

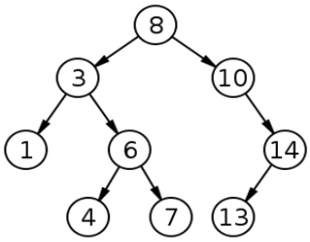
[1] Jayme Luiz Szwarcfiter, Lilian markezon. Estruturas de Dados e Seus Algoritmos, 3rd Edition. [BV:MB]. 3a Edição. Rio de Janeiro: LTC. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788521629955/cfi/6/8!/4/2/4@0:0

8 Aprenda +

Atividade Autônoma Aura - AAA

Questão 1:Implementar os algoritmos recursivos de Pré-ordem, Em ordem e Pós-ordem de uma Árvore Binária (*Binary Tree*).

Questão 2:Dê o resultado dos algoritmos de caminhamento em **Pré Ordem**, **Pós Ordem** e **Em ordem** da seguinte Árvore de Busca Binária:



Plano de Aula

1 Código e nome da disciplina

ARA0174 ALGORITMOS E COMPLEXIDADE

2 Semana/Tema

Semana 11: Tema - 4. ALGORITMOS EM ÁRVORES BINÁRIA E ÁRVORE AVL

3 Objetivos

Descrever os principais conceitos de balanceamento e realizar a análise de complexidade.

4 Tópicos

4.3 BALANCEAMENTO DE ÁRVORE: CONCEITOS E ALGORITMO DSW E ANÁLISE DE COMPLEXIDADE

5 Procedimentos de ensino-aprendizagem

Nesta aula, serão apresentadas os principais conceitos de balanceamento em Árvore de Pesquisa Binária (Binary Tree) e seus principais algoritmos e análise de complexidade de cada um. Para isso, segue sugestão de roteiro:

- Situação-problema: Supondo que uma empresa de tecnologia tenha interesse em manter eficientemente balanceamento dos seus dados de arquivo. Como manter esse balanceamento através de modelagem do problema da árvore binária? Além das árvores serem muito apropriadas para representar a estrutura hierárquica de um certo domínio, o processo de busca por um elemento em uma árvore tende a ser muito mais rápido do que em uma lista encadeada. O algoritmo Day-Stout-Warren (DSW) é um método para equilibrar eficientemente as árvores de pesquisa binária - ou seja, diminuindo sua altura para O (log n) nós, em que n é o número total de nós.

- Metodologia: Apresentar (e rastrear) a implementação do algoritmo Day-Stout-Warren (DSW) e sua análise de complexidade. O algoritmo baseia-se em percorrer uma árvore binária de busca tornando-a uma arvore degenerada (similar a uma lista encadeada) e posteriormente percorrê-la novamente tornando-a uma árvore perfeitamente balanceada.

- Atividade verificadora de Aprendizagem: O professor deve mediar a implementação dos algoritmo DSW e se houve um perfeito balanceamento das árvores implementadas.

6 Recursos didáticos

Laboratório de Programação de Computadores, equipado com quadro branco, computadores com acesso à Internet, além de Datashow conectado ao computador do docente.

7 Leitura específica

[1] Jayme Luiz Szwarcfiter, Lilian markezon. Estruturas de Dados e Seus Algoritmos, 3rd Edition. [BV:MB]. 3a Edição. Rio de Janeiro: LTC. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788521629955/cfi/6/8!/4/2/4@0:0

[2] Nivio Ziviane. Projeto de algoritmos: com implementações em Java e C++. [BV:MB]. São Paulo: Cengage Learning. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788522108213/cfi/2!/4/4@0.00:49.1

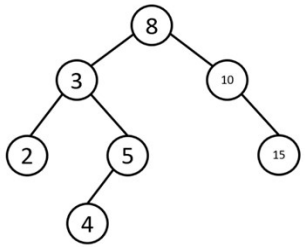
[3] Thomas H. Cormen; Charles E. Leiserson; Ronald L. Rivest; Stein, Clifford. Algoritmos: Teoria e Prática [BV:PE]. Rio de Janeiro: Elsevier Editora Ltda, 2012. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595158092/cfi/6/4!/4/2/2@0:0

8 Aprenda +

- Leitura recomendada do artigo sobre o algoritmo de balanceamento em árvores ***Tree Rebalancing in Optimal Time and Space***. Disponível em http://web.eecs.umich.edu/~qstout/pap/CACM86.pdf
- Adam Drozdek (1996).*Data Structures and Algorithms in C++*.

Atividade Autônoma Aura - AAA

Questão 1:Seja a seguinte árvore binária abaixo. Dê os valores de fator de balanceamento de cada nó da árvore.



Questão 2: Assinale com X as alternativas corretas acerca do balanceamento de árvores binárias e do algoritmo DSW (Day–Stout–Warren)

É correto o que se afirma em:
( ) O algoritmo Day-Stout-Warren (DSW) é um algoritmo que mantém um balanceamento local de cada nó de uma árvore binária. Sua complexidade de tempo é O(n^2).
( ) O Desbalanceamento progressivo de uma árvore binária ocorre somente no processo de inserção de nós.
( ) A forma mais natural de definirmos uma estrutura de árvore é usando recursividade.
( ) O objetivo do algoritmo Day-Stout-Warren (DSW) é “destruir” a árvore binária original que está desbalanceada e construí-la através de sucessivos processos de rotação, mantendo-a balanceada.

Plano de Aula

1 Código e nome da disciplina

ARA0174 ALGORITMOS E COMPLEXIDADE

2 Semana/Tema

Semana 12: Tema - 4. ALGORITMOS EM ÁRVORES BINÁRIA E ÁRVORE AVL

3 Objetivos

Reconhecer as principais propriedades da Árvore de Pesquisa Binária e AVL e realizar a análise de complexidade de cada um deles.

4 Tópicos

4.4 ÁRVORE AVL: CONCEITOS, PROPRIEDADES E ANÁLISE DE COMPLEXIDADE

5 Procedimentos de ensino-aprendizagem

Nesta aula, será apresentado o algoritmo de balanceamento local em árvores binárias, transformando-a em Árvores AVL e análise de complexidade de cada um. Para isso, segue sugestão de roteiro:

- Situação-problema: Supondo que uma empresa de tecnologia deseja que um programador efetue o balanceamento de suas estruturas (árvores) hierárquicas, para que a busca e outras operações possam ser realizadas de forma eficiente. Além de que, esse balanceamento deve ser feito de forma local. Vimos na aula anterior algoritmos que balanceiam árvores globalmente. Contudo, o rebalanceamento pode ocorrer localmente se alguma porção da árvore for desbalanceada por uma operação de inserção ou remoção de um elemento da árvore. Um método clássico para tal foi proposto por Adelson-Velskii e Landis e o nome dado a uma árvore modificada com este método é árvore AVL. O balanceamento de um nó da árvore binária é definido como a altura de sua subárvore esquerda menos a altura de sua subárvore direita. Cada nó numa árvore binária balanceada (AVL) tem balanceamento de 1, -1 ou 0. Se o valor do balanceamento do nó for diferente de 1, -1 e 0, então essa árvore não é balanceada (AVL).

- Metodologia: Apresentar de forma expositiva (e rastrear) a implementação da árvore AVL e sua análise de complexidade.

- Atividade verificadora de Aprendizagem: O professor deve mediar o entendimento dos conceitos do algoritmos de operações de árvore AVL.

6 Recursos didáticos

Laboratório de Programação de Computadores, equipado com quadro branco, computadores com acesso à Internet, além de Datashow conectado ao computador do docente.

7 Leitura específica

[1] Jayme Luiz Szwarcfiter, Lilian markezon. Estruturas de Dados e Seus Algoritmos, 3rd Edition. [BV:MB]. 3a Edição. Rio de Janeiro: LTC. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788521629955/cfi/6/8!/4/2/4@0:0

[2] Nivio Ziviane. Projeto de algoritmos: com implementações em Java e C++. [BV:MB]. São Paulo: Cengage Learning. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788522108213/cfi/2!/4/4@0.00:49.1

[3] Thomas H. Cormen; Charles E. Leiserson; Ronald L. Rivest; Stein, Clifford. Algoritmos: Teoria e Prática [BV:PE]. Rio de Janeiro: Elsevier Editora Ltda, 2012. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595158092/cfi/6/4!/4/2/2@0:0

[4] Adelson-Velskii, G. M.; Landis, E. M. (1962), "An algorithm for organization of information", Doklady Akademii Nauk SSSR, 146: 263?266.

8 Aprenda +

Atividade Autônoma Aura - AAA

Questão 1:Uma árvore binária é denominada **AVL** quando, para qualquer nó da árvore, a **diferença** das alturas de suas duas subárvores, esquerda e direita, é no **máximo** 1. O nome AVL da árvore vem dos pesquisadores em ciência da computação e matemáticos soviéticos **Adelson Velsky** e **Landis** [4]. Para manter o fator de balanceamento tanto na inserção quanto na exclusão de nós, é necessário realizar quatro (4) processos de rotação de nós.
Implemente e dê os algoritmos em linguagem estruturada de (2) processos de **rotação simples** de nós de uma árvore AVL.

Questão 2: Marque a alternativa correta quanto as árvores AVL:

A) Em uma rotação dupla a direita, significa que as rotações sucessivas no nó são feitas à direita duas vezes no nó que está desregulado.
B) O tempo de pesquisar em uma árvore AVL é de O(log n).
C) Toda árvore AVL é balanceada, isto é, sua altura é O(n log n).
D) A inserção dos seguintes nós, exatamente nessa ordem: 50, 40, 30, 45, 47, 55, 56, 52, 1, 2, 3, 49 gera uma árvore que é AVL.
E) O fator de balanceamento serve para averiguar se um nó está desregulado e assim, o algoritmo de DSW aplicado ao nó desregulado.

Plano de Aula

1 Código e nome da disciplina

ARA0174 ALGORITMOS E COMPLEXIDADE

2 Semana/Tema

Semana 13: Tema - 5. ALGORITMOS EM GRAFOS (ATIVIDADE PRÁTICA SUPERVISIONADA)

3 Objetivos

Descrever a contextualização e os principais conceitos de teoria dos grafos, suas principais classes e propriedades estruturais.

4 Tópicos

5.1 CONCEITOS DE GRAFOS

5 Procedimentos de ensino-aprendizagem

Nesta aula, será apresentado os principais conceitos em teoria dos grafos, suas principais classes e propriedades combinatórias estruturais. Para isso, segue sugestão de roteiro:

- Situação-problema: A teoria dos grafos é um ramo da matemática que estuda as relações entre os objetos de um determinado conjunto. Para tal são utilizadas estruturas chamadas de grafos, onde V é um conjunto não vazio de objetos denominados vértices e E é um subconjunto de pares não ordenados de V. É amplamente utilizada para modelar muitos problemas clássicos em computação. A preocupação central da teoria dos grafos é a busca por algoritmos eficientes que resolvam problemas sobre grafos, portanto, é de suma importância apresentar nessa etapa diversos problemas em computação que são modelados através de grafos, como por exemplo:
o Grafo da web: A World Wide Web pode ser modelada como um grafo orientado, no qual cada página é representada por um vértice e uma aresta começa na página a da web e termina na página b, se existir um link em a que direcione para b.
o Grafos de mapas rodoviários: Usando grafos para modelar mapas rodoviários, os vértices representam as intersecções e as arestas representam estradas. Arestas não orientadas representam estradas de mão dupla, e as arestas orientadas representam estradas de mão única. Arestas não orientadas múltiplas representam estradas de mão dupla múltiplas que conectam as mesmas duas intersecções.
o Grafos de relação interpessoal em redes sociais: Amplamente utilizado para modelar as relações entre pessoas nas redes sociais, um grafo pode representar uma pessoa (vértice) que tenha amizade com outra pessoa (outro vértice) e a amizade entre elas é representada através de arestas.

- Metodologia: Apresentar de forma expositiva os conceitos de grafos, suas propriedades estruturais e suas principais classes de grafos.

- Atividade verificadora de Aprendizagem: O professor deve mediar o entendimento dos conceitos de grafos e suas principais definições estruturais e suas classes.

6 Recursos didáticos

Laboratório de Programação de Computadores, equipado com quadro branco, computadores com acesso à Internet, além de Datashow conectado ao computador do docente.

7 Leitura específica

[1] Jayme Luiz Szwarcfiter, Lilian markezon. Estruturas de Dados e Seus Algoritmos, 3rd Edition. [BV:MB]. 3a Edição. Rio de Janeiro: LTC. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788521629955/cfi/6/8!/4/2/4@0:0

[2] Nivio Ziviane. Projeto de algoritmos: com implementações em Java e C++. [BV:MB]. São Paulo: Cengage Learning. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788522108213/cfi/2!/4/4@0.00:49.1

[3] Thomas H. Cormen; Charles E. Leiserson; Ronald L. Rivest; Stein, Clifford. Algoritmos: Teoria e Prática [BV:PE]. Rio de Janeiro: Elsevier Editora Ltda, 2012. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595158092/cfi/6/4!/4/2/2@0:0

8 Aprenda +

Plano de Aula

1 Código e nome da disciplina

ARA0174 ALGORITMOS E COMPLEXIDADE

2 Semana/Tema

Semana 14: Tema - 5. ALGORITMOS EM GRAFOS (ATIVIDADE PRÁTICA SUPERVISIONADA)

3 Objetivos

Descrever a representação computacional de grafos e implementar utilizando linguagem de programação imperativa e estruturada.

4 Tópicos

5.2 REPRESENTAÇÃO DE GRAFO

5 Procedimentos de ensino-aprendizagem

Nesta aula, será apresentado as principais formas de representação de grafos, através de matriz de adjacência e lista de adjacência. Para isso, segue sugestão de roteiro:

- Situação-problema: Propor em sala diversas formas de representação de grafos e determinar qual tipo de grafos a ser representado computacionalmente e a sua melhor forma de implementação. Em geral, grafos esparsos são melhores representados por matriz de adjacência e grafos densos por lista de adjacência. Os grafos são estruturas de dados muito usadas na ciência da computação e seus algoritmos são fundamentais nesta área. Muitos problemas são definidos em termos de grafos e resolvidos com recursos computacionais. Por isso, a importância da escolha da melhor de representar grafos.

- Metodologia: Apresentar de forma expositiva a implementação das principais formas de representar os grafos, seja por matriz de adjacência, seja por lista de adjacência e a análise de complexidade de cada uma delas.

- Atividade verificadora de Aprendizagem: De acordo com as formas de representação de grafos proposto na situação-problema, mediar qual (ou quais) as melhores formas de representar computacionalmente o grafo escolhido.

6 Recursos didáticos

Laboratório de Programação de Computadores, equipado com quadro branco, computadores com

acesso à Internet, além de Datashow conectado ao computador do docente.

7 Leitura específica

[1] Jayme Luiz Szwarcfiter, Lilian markezon. Estruturas de Dados e Seus Algoritmos, 3rd Edition. [BV:MB]. 3a Edição. Rio de Janeiro: LTC. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788521629955/cfi/6/8!/4/2/4@0:0

[2] Nivio Ziviane. Projeto de algoritmos: com implementações em Java e C++. [BV:MB]. São Paulo: Cengage Learning. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788522108213/cfi/2!/4/4@0.00:49.1

[3] Thomas H. Cormen; Charles E. Leiserson; Ronald L. Rivest; Stein, Clifford. Algoritmos: Teoria e Prática [BV:PE]. Rio de Janeiro: Elsevier Editora Ltda, 2012. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595158092/cfi/6/4!/4/2/2@0:0

8 Aprenda +

Leitura extra recomendada: NICOLETTI, M.C., E.R. HRUSCHKA JR. ***Fundamentos da Teoria dos Grafos para Computação***. Edição: 3. Editora: LTC. 2018

Plano de Aula

1 Código e nome da disciplina

ARA0174 ALGORITMOS E COMPLEXIDADE

2 Semana/Tema

Semana 15: Tema - 5. ALGORITMOS EM GRAFOS (ATIVIDADE PRÁTICA SUPERVISIONADA)

3 Objetivos

Descrever algoritmos de busca em grafos, Depth-First Search (DFS) e Breadth-First Search (BFS) e suas principais propriedades.

4 Tópicos

5.3 ALGORITMOS DE BUSCA

5 Procedimentos de ensino-aprendizagem

Nesta aula, será apresentado os principais algoritmos de busca em grafos. Para isso, segue sugestão de roteiro:
- Situação-problema: Supor situações clássicos que podem ser resolvidos através de busca/visita em grafos, tais como ordenação topológica de elementos, labirintos, menor caminhos em grafos a parti de um nó inicial, etc. Os algoritmos de busca são os procedimentos mais utilizados na teoria de grafos, com objetivo de mostrar uma sistemática de como caminhar pelos vértices e arestas de um grafo. Para tanto, existem dois métodos de busca: DFS - Depth First Search (Busca em Profundidade) e BFS - Breadth First Search (Busca em Largura).
o DFS (Depth First Search): A ideia básica da DFS é buscar mais fundo no grafo quando possível. Este procedimento atende a um critério de explorar um vértice marcado (ou visitado), dentre os vários marcados e incidentes a alguma aresta ainda não explorada, escolher aquele vértice mais recentemente alcançado na busca.
o BFS - Breadth First Search: Na busca em largura (BFS) os vértices do grafo são visitados nível a nível. Segundo Cormen [3], dado um grafo G(V, E) e um vértice de origem v, a busca em largura explora as arestas de G até todos os vértices alcançáveis a partir de v. Além disso, o algoritmo calcula a menor distância em número de arestas de v até todos os vértices acessíveis a ele. A busca em largura recebe esse nome porque descobre todos os vértices que estão a uma distância k de v, antes de descobrir os vértices que se encontram a uma distância k + 1.

- Metodologia: Apresentar de forma expositiva a implementação dos principais algoritmos de busca em grafos DFS e BFS.

- Atividade verificadora de Aprendizagem: De acordo com as situações clássicas apresentadas na situação problema.

6 Recursos didáticos

Laboratório de Programação de Computadores, equipado com quadro branco, computadores com acesso à Internet, além de Datashow conectado ao computador do docente.

7 Leitura específica

[1] Jayme Luiz Szwarcfiter, Lilian markezon. Estruturas de Dados e Seus Algoritmos, 3rd Edition. [BV:MB]. 3a Edição. Rio de Janeiro: LTC. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788521629955/cfi/6/8!/4/2/4@0:0

[2] Nivio Ziviane. Projeto de algoritmos: com implementações em Java e C++. [BV:MB]. São Paulo: Cengage Learning. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788522108213/cfi/2!/4/4@0.00:49.1

[3] Thomas H. Cormen; Charles E. Leiserson; Ronald L. Rivest; Stein, Clifford. Algoritmos: Teoria e Prática [BV:PE]. Rio de Janeiro: Elsevier Editora Ltda, 2012. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595158092/cfi/6/4!/4/2/2@0:0

8 Aprenda +

Plano de Aula

1 Código e nome da disciplina

ARA0174 ALGORITMOS E COMPLEXIDADE

2 Semana/Tema

Semana 16: Tema - 5. ALGORITMOS EM GRAFOS (ATIVIDADE PRÁTICA SUPERVISIONADA)

3 Objetivos

4 Tópicos

5.4 ALGORITMO DO CAMINHO MÍNIMO

5 Procedimentos de ensino-aprendizagem

6 Recursos didáticos

7 Leitura específica

8 Aprenda +