Estruturas de Dados II

Prof. Patrícia Noll de Mattos

Objetivos

Geral

O objetivo da disciplina é estudar modelos abstratos da representação de dados e identificar, através de algoritmos genéricos, possíveis implementações computacionais para os modelos identificados.

Específicos

Estimular o desenvolvimento e aprimoramento das seguintes habilidades:

- aplicar recursividade aos algoritmos vistos na disciplina;
- conhecer os conceitos de árvores e conseguir implementá-los em alguma linguagem de programação;
- implementar algoritmos de classificação de dados em memória e externos, conhecendo sua complexidade;
- implementar algoritmos de pesquisa de dados em memória, conhecendo sua complexidade;
- conhecer os algoritmos básicos de compressão de dados textuais e imagens.

Eixos Temáticos

- RECURSIVIDADE
- ÁRVORES
  - Árvores binárias
  - Árvores Balanceadas
  - Árvores Genéricas
  - Árvores B
- CLASSIFICAÇÃO DE DADOS
- PESQUISA DE DADOS
- COMPRESSÃO DE DADOS

Avaliação

Tanto para G1 como para G2

- Exercícios para entregar: 2.0 pontos
- Avaliação Presencial/Individual: 8.0 pontos

Revisão

Estruturas sequenciais x Estruturas encadeadas

Características?

Alocação estática | Alocação dinâmica

Posições contíguas | Posições aleatórias

Toda estrutura Alocada | Cada elemento

Tamanho fixo | Tamanho variável

Tempo de compilação | Tempo de exeução

Estruturas encadeadas: sequência lógica

Sequência fornecida por campo de ligação (link)

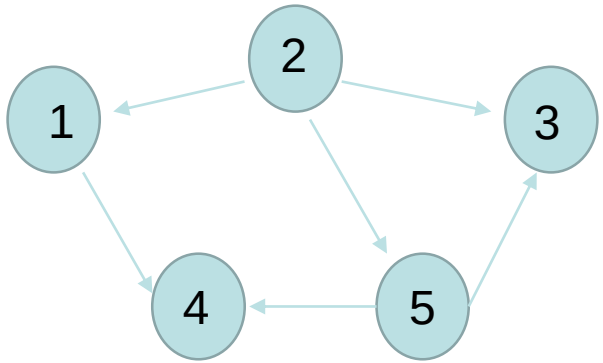
- simplesmente: um campo de ligação

5	7	8
end prox	end prox	end prox

- duplamente: 2 campos de ligação

5	7	8
end prox	end prox	end prox
end ant	end ant	end ant

- não regular: vários campos de ligação



Lista de Lista

Criando uma lista simplesmente encadeada

```
struct nodo {
        int dados;
        struct nodo *prox;
};
struct nodo *aux, *inicio=NULL, *fim=NULL;

aux = (struct nodo *) malloc (sizeof(struct nodo));
if(aux != NULL){
        aux->dados = valor;
        aux->prox = NULL;
        if(inicio==NULL) inicio=aux;
        else fim->prox = aux;
        fim = aux; }
```

Criando uma lista duplamente encadeada

```
struct nodo {
        int dados;
        struct nodo *prox;
        struct nodo *ant;
};
struct nodo *aux, *inicio=NULL, *fim=NULL;

aux = (struct nodo *) malloc (sizeof(struct nodo));
if(aux != NULL){
        aux->dados = valor;
        aux->prox = NULL;
        aux->ant = fim;
        if(inicio==NULL) inicio=aux;
        else fim->prox = aux;
        fim = aux; }
```

Com header

```
struct header {
      int qtde;
      struct nodo *inicio;
      struct nodo *fim;
};

struct header *lista;

lista = (struct header *) malloc (sizeof(struct header));
if(lista != NULL)
{
  lista->qtde=0;
  lista->inicio=NULL;
  lista->fim=NULL;
}
```

Com header

```
struct nodo *aux, *inicio=NULL, *fim=NULL;

aux = (struct nodo *) malloc (sizeof(struct nodo));
if(aux != NULL){
        aux->dados = valor;
        aux->prox = NULL;
        aux->ant = lista->fim;
        if(lista->inicio==NULL) lista->inicio=aux;
        else lista->fim->prox = aux;
        lista->fim = aux;
        lista->qtde++;
}
```

Exercício:

1. Escreva um programa que leia valores para uma lista duplamente encadeada com header, pare quando for digitado zero.
2. Mostre o conteúdo da lista na tela.
3. Calcule a média de valores da lista e mostre na tela.
4. Insira um novo nodo na lista contendo essa média como conteúdo.