**Bacharelado em Ciência da Computação**Centro de Ciências Exatas e Tecnológicas - CCET
**Disciplina**: Estrutura de Dados
**Data**: 03/08/2022
**Descrição**: Atividade Prática - AP02

unioeste
Universidade Estadual do Oeste do Paraná

# IMPLEMENTAÇÃO DE UM GRAFO, UMA FILA EM ORIENTAÇÃO A OBJETOS USANDO HERANÇA E UM MÉTODO DE ORDENAÇÃO.

**Aluna: Ellen Carine Bonafin Marques e Eduarda Elger**
**E-mail(s):** ellencarine41@gmail.com , eduarda_elger@hotmail.com
**GIT:** https://github.com/EllenBonafin/Tb_Final_ED

## 1. Introdução.

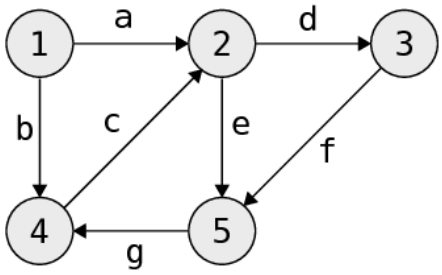
Este trabalho foi desenvolvido para implementar os principais conceitos de Grafos, Programação Orientada a Objetos e Métodos de Ordenação, neste caso o método implementado foi o Shellsort, que será apresentado posteriormente. Além disso, esta tese foi elaborada para a disciplina Estrutura de dados, ministrada pelo professor Leonardo Medeiros.

Toda a implementação apresentada nesta documentação foi codificada, compilada e feita com a ferramenta online Replit, onde seu terminal utiliza a seguinte versão Linux: "Ubuntu 20.04.2 LTS". Além disso, todas as três atividades terão três arquivos durante o desenvolvimento, a saber, main.c/cpp (programa principal), module.c/cpp (funções usadas pelo programa principal) e module.h/hpp (funções usadas pelo programa principal), exceto o makefile que incluirá a opção run para compilar todo o código fonte e executá-lo conforme solicitado pelo professor atual.

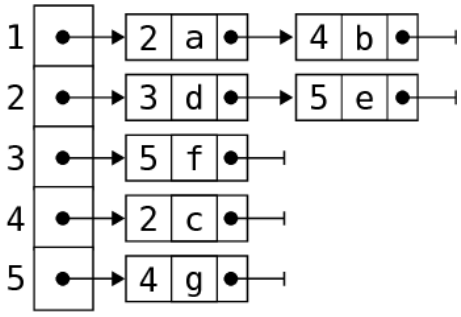
## 2. Métodos implementados.

a. TAD Grafo através de Listas de Adjacências usando ponteiros.

Uma lista de adjacências é uma estrutura de dados que representa um gráfico. Em uma representação de lista de adjacências, podemos manter, para cada vértice do grafo, uma lista de todos os outros vértices que possuem arestas (a "lista de adjacências" para aquele vértice). Conforme exemplo abaixo:




Grafo direcionado




Lista de adjacência

Partindo deste principio, implementamos então da seguinte maneira:

- Entradas: Quantidade de vértices e graus que o grafo possuirá. Conforme demonstrado abaixo.

```
Digite a quantidade de vértices o grafo possuirá:
5
Digite a quantidade de graus o grafo possuirá:
6
```

- Saídas: A saída de dados é determinada conforme a escolha que o usuário realiza através do seguinte menu que é apresentado na tela:

```
Escolha uma opção abaixo:
1. Imprimir gravo na tela.
2. Adicionar uma nova aresta.
3. Remover uma aresta.
4. Buscar no grafo.
5. Desalocar gravo.
0. Sair.
Digite a opção escolhida:
```

Sendo assim, abaixo será demonstrado conforme a opção escolhida como será a saída no terminal:

- Opção 1:

```
Digite a opção escolhida:
1
0:  0  0
1:  2  1  4
2:  2  2
3:  4  0  3  3
4:
```

- Opção 2:

```
Digite a opção escolhida:
2
Determine sua origemVertice: 1
Determine seu destinoVertice: 2
```

- Opção 3:

```
Digite a opção escolhida:
3
Determine sua origemVertice: 1
Determine seu destinoVertice: 2
```

- Opção 4:

```
Digite a opção escolhida:
4
Busca dentro do grafo:

1 - Procurar o menor caminho
2 - Busca em profundidade
3 - Busca em largura
Digite a opção escolhida:
1
Procura o menor caminho:

Determine o vertice inicial: 2
0: -1 -> {-nan}
1: -1 -> {-nan}
2: -1 -> {0.00}
3:  0 -> {-nan}
4: -1 -> {-1.00}
```

```
Digite a opção escolhida:
4
Busca dentro do grafo:

1 - Procurar o menor caminho
2 - Busca em profundidade
3 - Busca em largura
Digite a opção escolhida:
2
Busca em profundidade:

Determine o vertice inicial: 2
Busca em profundidade:

0 ->{0}
1 ->{0}
2 ->{1}
3 ->{0}
4 ->{0}
```

```
Digite a opção escolhida:
4
Busca dentro do grafo:

1 - Procurar o menor caminho
2 - Busca em profundidade
3 - Busca em largura
Digite a opção escolhida:
3
Busca em profundidade:
Determine o vertice inicial:
2
Busca em largura:

0 ->{0}
1 ->{0}
2 ->{1}
3 ->{0}
4 ->{0}
```

- Opção 5:

```
Digite a opção escolhida:
5
Desalocando o grafo.
```

- Funções:
  - typedef struct grafo Grafo: Função usada pra criar o grafo.
  - criaG: Função que inicializa o grafo.
  - desalocaG: Função que desaloca o grafo.
  - insereA: Função que realiza a inserção de valores randomizados.
  - removeA: Função de remoção.
  - imprimeG: Função de impressão na tela.
  - Funções de busca do grafo:
    - profundidadeG: Realiza a busca por profundidade.
    - larguraG: Realiza a busca por largura.
    - mCaminho: Realiza a busca pelo menor caminho.

**b.** TAD Fila usando herança de uma Lista Encadeada Simples em C++.

A capacidade de uma classe de derivar propriedades e características de outra classe é chamada de **Herança** . Herança é um dos recursos mais importantes da Programação Orientada a Objetos.

A entrada e saída desta atividade é única e exclusivamente realizada conforme a opção que o usuário seleciona em um menu que é fornecido assim que o código é executado, demonstrado abaixo:

```
Escolha uma opção para ser realizada:
1. Inserir um elemento no inicio.
2. Inserir um elemento no final.
3. Remover um elemento do inicio.
4. Remover um elemento do final.
5. Imprimir a fila.
6. Buscar um elemento na fila.
7. Buscar um elemento em uma posição na fila.
8. Mostrar o primeiro elemento da fila.
9. Mostrar o último elemento da fila.
10. Desalocar a fila.
0. Sair.
```

Um exemplo de entrada de dados é na opção 1, conforme demonstrado abaixo:

```
Digite a opção escolhida:
1
Informe o dado que será inserido:
5
```

Um exemplo de saída de dados é na opção 8:

```
Digite a opção escolhida:
8
O primeiro elemento da fila é: 5
```

As funções utilizadas neste código possuem comentários dos quais deixam explicados quais são seus papéis, sendo descritos abaixo:

- lista () : Inicializando a lista.
- ~lista (): Destrói a lista.
- tamanho(): Função que retorna o tamanho da lista.
- vazio(): Função que verifica se a lista está vazia.
- imprime(): Função que imprime a lista na tela.
- Funções de Inserção:
  - insereinicio(Dado Dado): Função que insere no inicio.
  - inserefinal(Dado Dado): Função que insere no final.
- Funções de Remoção:
  - removeinicio():Função que remove um elemnto do inicio da lista.
  - removefinal():Função que remove um elemento do final da lista.
- Funções de Busca:
  - buscaelemento(Dado Dado): Função que busca um elemento na lista.
  - buscaposicao(int pos): Função que retorna um elemento conforme a posição passada por parâmetro.
- primeiroNo():Função que retorna o primeiro elemento da fila.
- ultimoNo():Função que retorna o ultimo elemento da fila.

**c.** Ordenação por Inserção através de incrementos decrescentes (Shellsort).

Criado por *Donald Shell* em 1959, o método Shell Sort é uma extensão do algoritmo de ordenação por inserção. Ele permite a troca de registros distantes um do outro, diferente do algoritmo de ordenação por inserção que possui a troca de itens adjacentes para determinar o ponto de inserção. A complexidade do algoritmo é desconhecida, ninguém ainda foi capaz de encontrar uma *fórmula fechada* para sua função de complexidade e o método não é estável.

Os itens separados de h posições (itens distantes) são ordenados: o elemento na posição x é comparado e trocado (caso satisfaça a condição de ordenação) com o elemento na posição x-h. Este processo repete até h=1, quando esta condição é satisfeita o algoritmo é equivalente ao método de inserção.

- Entrada: É solicitado ao usuário 10 valores que são armazenados em um vetor, conforme demonstrado abaixo:

```
A seguir informe os valores da sequencia:
Informe um valor: 9
Informe um valor: 3
Informe um valor: 0
Informe um valor: 4
Informe um valor: 10
Informe um valor: 8
Informe um valor: 30
Informe um valor: 1
Informe um valor: 7
Informe um valor: 2
```

- Saída: É apresentado na tela o vetor ordenado.

```
Valores ordenados: 0 1 2 3 4 7 8 9 10 30
```

- Função:
    - shell_sort(int *a, int size): Função que realiza a ordenação após a inserção dos dados pelo usuário.

## 3. Referências.


LISTA de adjacência. [*S. l.*], 20 mar. 2018. Disponível em: https://pt.wikipedia.org/wiki/Lista_de_adjac%C3%AAncia#:~:text=Em%20ci%C3%AAncia%20da%20computa%C3%A7%C3%A3o%2C%20uma,adjac%C3%AAncia%22%2C%20deste%20v%C3%A9rtice). Acesso em: 7 ago. 2022.

REPRESENTAÇÃO de redes por meio de lista de adjacências composta por n listas encadeadas. [*S. l.*], dez. 2013. Disponível em: https://www.researchgate.net/figure/Figura-A2-Representacao-de-redes-por-meio-de-lista-de-adjacencia-composta-por-n-listas_fig12_279962622. Acesso em: 7 ago. 2022.

HERANÇA EM C++. [*S. l.*]. Disponível em: https://acervolima.com/heranca-em-c/. Acesso em: 7 ago. 2022.

CONHEÇA os principais algoritmos de ordenação. [*S. l.*]: Daniel Viana. Disponível em: https://www.treinaweb.com.br/blog/conheca-os-principais-algoritmos-de-ordenacao#:~:text=Shell%20Sort,-Criado%20por%20Donald&text=Os%20itens%20separados%20de%20h,equivalente%20ao%20m%C3%A9todo%20de%20inser%C3%A7%C3%A3o. Acesso em: 7 ago. 2022.