Grafos – Busca em Profundidade e em Largura

Prof. Luciano Antonio Digiampietri

Representação utilizada nesta aula

- Lista de adjacências
  - Por simplificação utilizaremos um arranjo de adjacências
- Cada nó possuirá:
  - Um arranjo de vizinhos
  - Um atributo com o número de vizinhos (igual a vizinhos.length)
  - Um atributo binário para indicar se já foi visitado
  - Alguns atributos adicionais que serão utilizados em outras implementações

```
class No{
 int id;
 int numVizinhos;
 No[] vizinhos;
 boolean visitado;
 int cor;
 int distancia;
 No(){}
 No(int pid){
    id = pid;
 }
}
```

Busca em Profundidade

A busca será realizada a partir de um nó inicial (atual) e procurando por um nó específico (fim).

Ela retornará, se houver, o tamanho do caminho percorrido do nó inicial até o nó buscado (não necessariamente será o menor caminho entre estes nós).

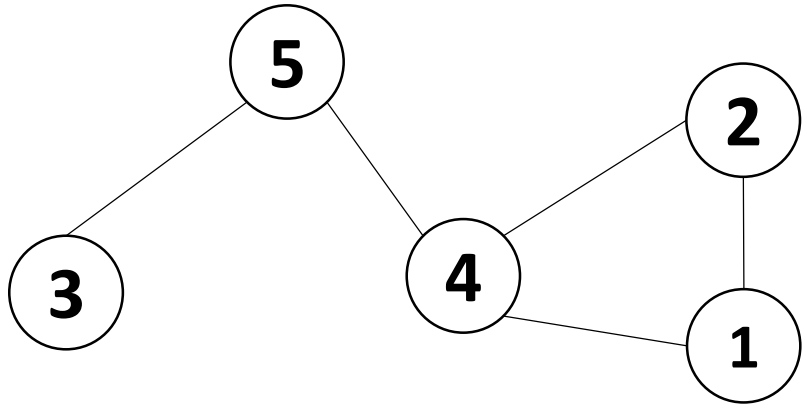
Realizaremos uma implementação recursiva.

Algoritmo

```
private static int buscaEmProfundidade(No atual, No fim){
   atual.visitado = true;
   if (atual == fim) return 0;
   for (No temp: atual.vizinhos){
      if (!temp.visitado){
         int res = buscaEmProfundidade(temp, fim) + 1;
         if (res > 0) return res;
      }
   }
   return -1;
}
```

Algoritmo

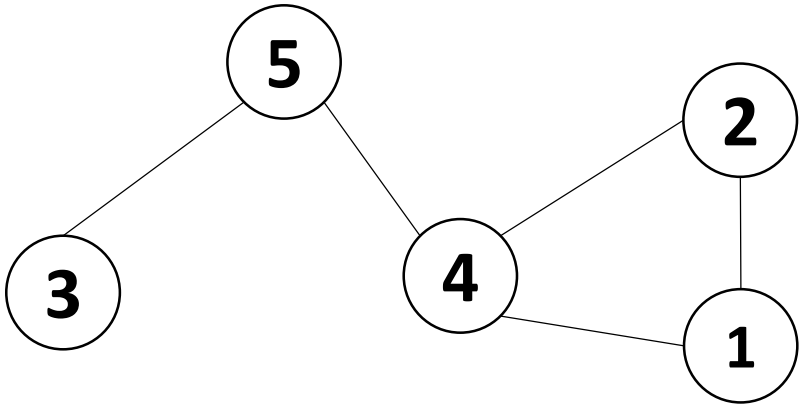
```
private static int buscaEmProfundidade(No atual, No fim){
   atual.visitado = true;
   if (atual == fim) return 0;
   for (No temp: atual.vizinhos){
      if (!temp.visitado){
         int res = buscaEmProfundidade(temp, fim) + 1;
         if (res > 0) return res;
      }
   }
   return -1;
}
```



Algoritmo

```
private static int buscaEmProfundidade(No atual, No fim){
   atual.visitado = true;
   if (atual == fim) return 0;
   for (No temp: atual.vizinhos){
      if (!temp.visitado){
         int res = buscaEmProfundidade(temp, fim) + 1;
         if (res > 0) return res;
      }
   }
   return -1;
}
```

buscando o nó 2

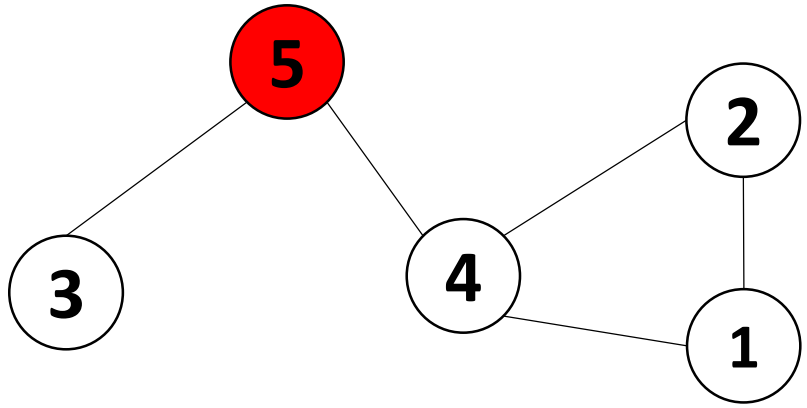


Algoritmo

```
private static int buscaEmProfundidade(No atual, No fim){
   atual.visitado = true;
   if (atual == fim) return 0;
   for (No temp: atual.vizinhos){
      if (!temp.visitado){
         int res = buscaEmProfundidade(temp, fim) + 1;
         if (res > 0) return res;
      }
   }
   return -1;
}
```

buscando o nó 2

1. atual=5



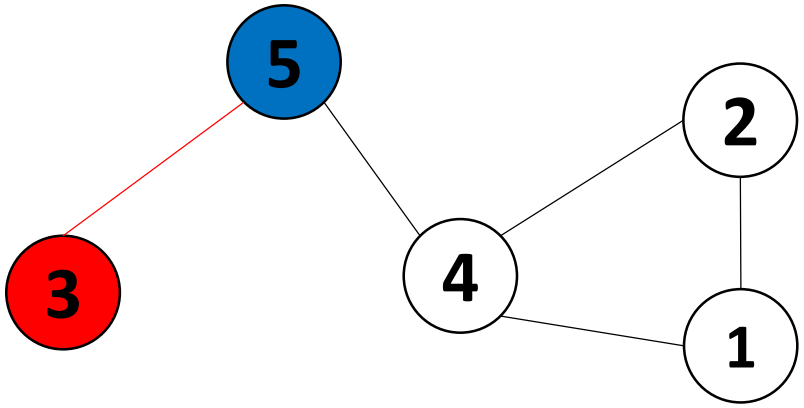
Algoritmo

```
private static int buscaEmProfundidade(No atual, No fim){
   atual.visitado = true;
   if (atual == fim) return 0;
   for (No temp: atual.vizinhos){
      if (!temp.visitado){
         int res = buscaEmProfundidade(temp, fim) + 1;
         if (res > 0) return res;
      }
   }
   return -1;
}
```

buscando o nó 2

1. atual=5

1.1. atual=3



Algoritmo

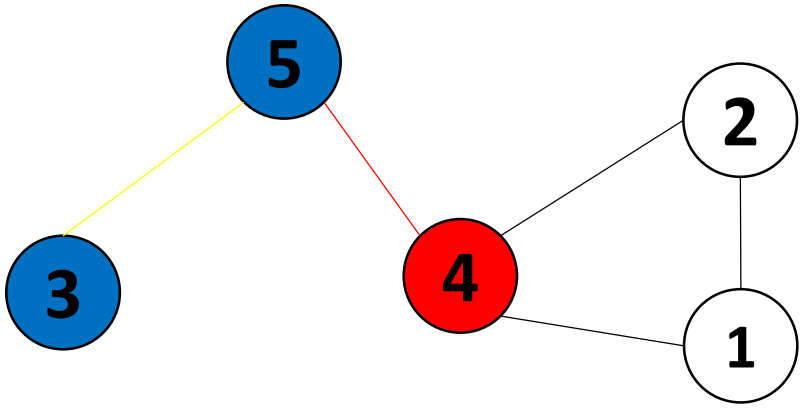
```
private static int buscaEmProfundidade(No atual, No fim){
   atual.visitado = true;
   if (atual == fim) return 0;
   for (No temp: atual.vizinhos){
      if (!temp.visitado){
         int res = buscaEmProfundidade(temp, fim) + 1;
         if (res > 0) return res;
      }
   }
   return -1;
}
```

buscando o nó 2

1. atual=5

~~1.1. atual=3~~

1.2. atual=4



Algoritmo

```
private static int buscaEmProfundidade(No atual, No fim){
   atual.visitado = true;
   if (atual == fim) return 0;
   for (No temp: atual.vizinhos){
      if (!temp.visitado){
         int res = buscaEmProfundidade(temp, fim) + 1;
         if (res > 0) return res;
      }
   }
   return -1;
}
```

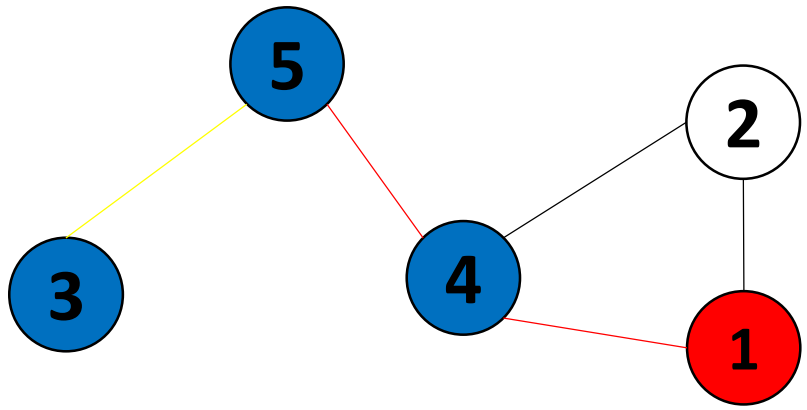
buscando o nó 2

1. atual=5

~~1.1. atual=3~~

1.2. atual=4

1.2.1 atual=1



Algoritmo

```
private static int buscaEmProfundidade(No atual, No fim){
   atual.visitado = true;
   if (atual == fim) return 0;
   for (No temp: atual.vizinhos){
      if (!temp.visitado){
         int res = buscaEmProfundidade(temp, fim) + 1;
         if (res > 0) return res;
      }
   }
   return -1;
}
```

buscando o nó 2

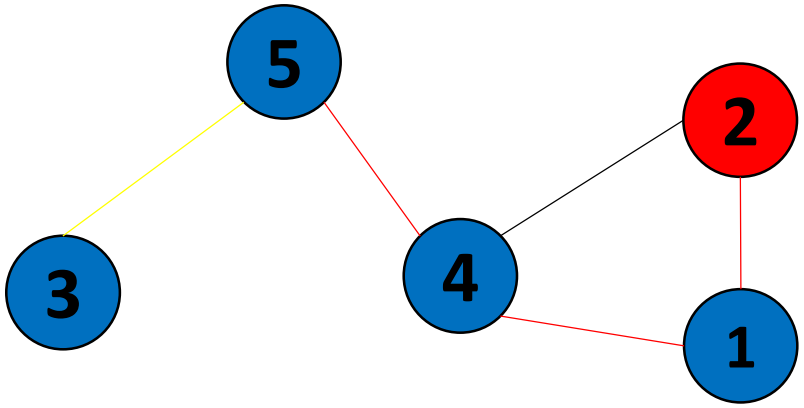
1. atual=5

~~1.1. atual=3~~

1.2. atual=4

1.2.1 atual=1

1.2.1.1 atual=2



Algoritmo

```
private static int buscaEmProfundidade(No atual, No fim){
   atual.visitado = true;
   if (atual == fim) return 0;
   for (No temp: atual.vizinhos){
      if (!temp.visitado){
         int res = buscaEmProfundidade(temp, fim) + 1;
         if (res > 0) return res;
      }
   }
   return -1;
}
```

buscando o nó 2

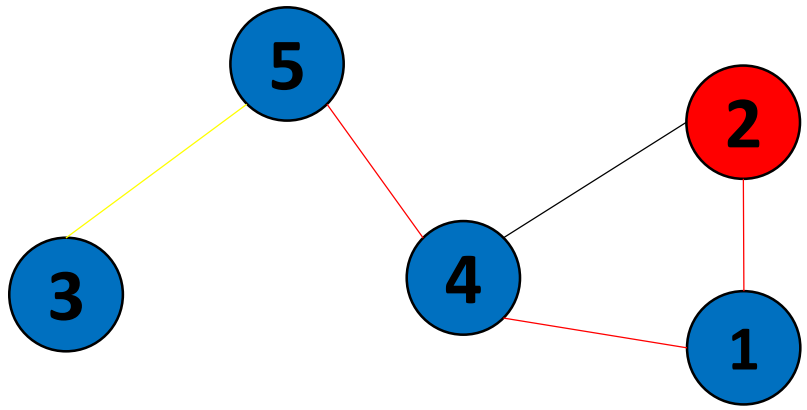
1. atual=5

~~1.1. atual=3~~

1.2. atual=4

1.2.1 atual=1

1.2.1.1 atual=2



Distância = 3

Busca em Largura

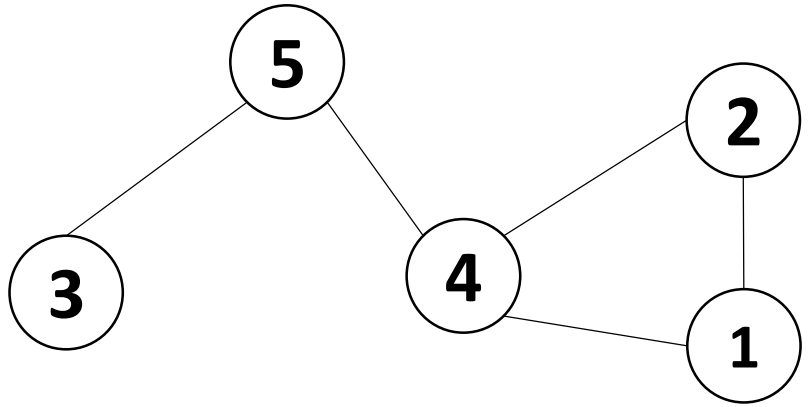
A busca será realizada a partir de um nó inicial (atual) e procurando por um nó específico (fim).

Ela retornará, se houver, o tamanho do menor caminho (em termos de número de arestas) do nó inicial até o nó buscado.

Realizaremos uma implementação iterativa (utilizando uma fila [utilizaremos a classe LinkedList])

```java
private static int buscaEmLargura(No inicio, No fim){
    if (inicio == fim) return 0;
    LinkedList<No> ll = new LinkedList<No>();
    inicio.distancia = 0; inicio.visitado = true;
    ll.add(inicio);
    No atual;
    while (!ll.isEmpty()){
       atual = ll.remove();
       for (No temp: atual.vizinhos){
          if (!temp.visitado){
              if (temp == fim) return atual.distancia + 1;
              temp.visitado = true;
              temp.distancia = atual.distancia + 1;
              ll.add(temp);
          }
       }
    }
    return -1;
 }
```

```java
private static int buscaEmLargura(No inicio, No fim){
    if (inicio == fim) return 0;
    LinkedList<No> ll = new LinkedList<No>();
    inicio.distancia = 0; inicio.visitado = true;
    ll.add(inicio);
    No atual;
    while (!ll.isEmpty()){
       atual = ll.remove();
       for (No temp: atual.vizinhos){
          if (!temp.visitado){
              if (temp == fim) return atual.distancia + 1;
              temp.visitado = true;
              temp.distancia = atual.distancia + 1;
              ll.add(temp);
          }
       }
    }
    return -1;
 }
```

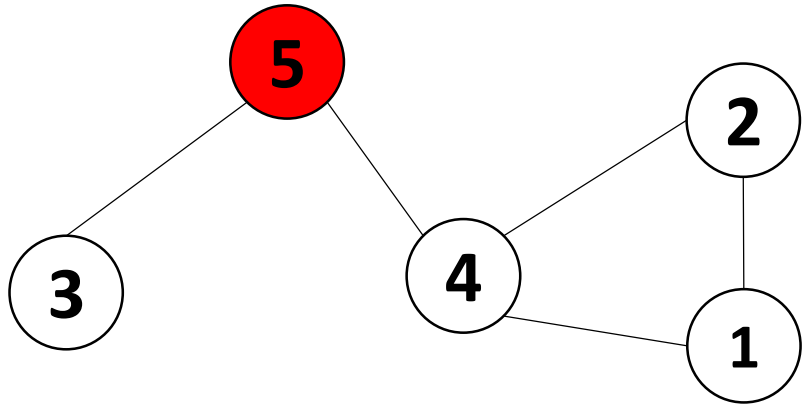


```java
private static int buscaEmLargura(No inicio, No fim){
    if (inicio == fim) return 0;
    LinkedList<No> ll = new LinkedList<No>();
    inicio.distancia = 0; inicio.visitado = true;
    ll.add(inicio);
    No atual;
    while (!ll.isEmpty()){
       atual = ll.remove();
       for (No temp: atual.vizinhos){
          if (!temp.visitado){
              if (temp == fim) return atual.distancia + 1;
              temp.visitado = true;
              temp.distancia = atual.distancia + 1;
              ll.add(temp);
          }
       }
    }
    return -1;
 }
```

buscando o nó 2

atual=5 ; dist=0

Fila: <vazia>

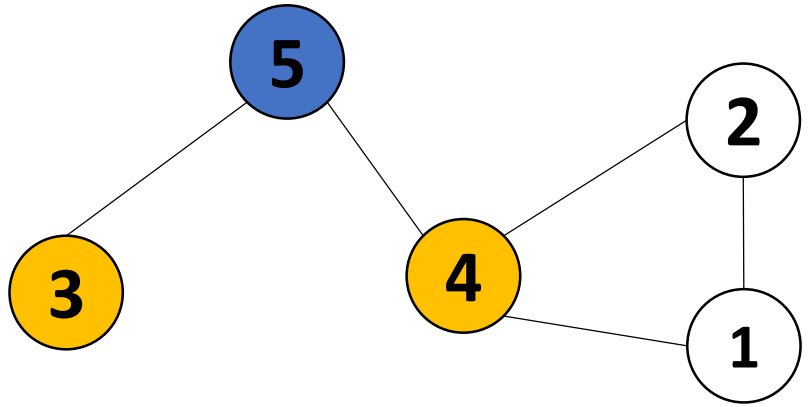


```java
private static int buscaEmLargura(No inicio, No fim){
    if (inicio == fim) return 0;
    LinkedList<No> ll = new LinkedList<No>();
    inicio.distancia = 0; inicio.visitado = true;
    ll.add(inicio);
    No atual;
    while (!ll.isEmpty()){
       atual = ll.remove();
       for (No temp: atual.vizinhos){
          if (!temp.visitado){
              if (temp == fim) return atual.distancia + 1;
              temp.visitado = true;
              temp.distancia = atual.distancia + 1;
              ll.add(temp);
          }
       }
    }
    return -1;
 }
```

buscando o nó 2

atual=5 ; dist=0

Fila: 3, 4

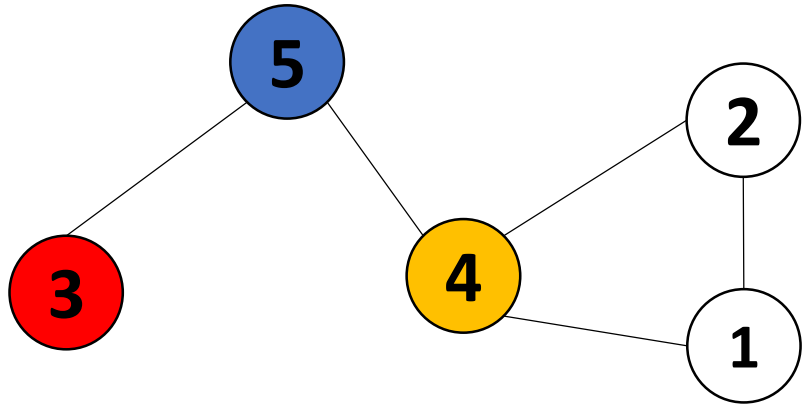


```java
private static int buscaEmLargura(No inicio, No fim){
    if (inicio == fim) return 0;
    LinkedList<No> ll = new LinkedList<No>();
    inicio.distancia = 0; inicio.visitado = true;
    ll.add(inicio);
    No atual;
    while (!ll.isEmpty()){
       atual = ll.remove();
       for (No temp: atual.vizinhos){
          if (!temp.visitado){
              if (temp == fim) return atual.distancia + 1;
              temp.visitado = true;
              temp.distancia = atual.distancia + 1;
              ll.add(temp);
          }
       }
    }
    return -1;
 }
```

buscando o nó 2

atual=3 ; dist=1

Fila: 4

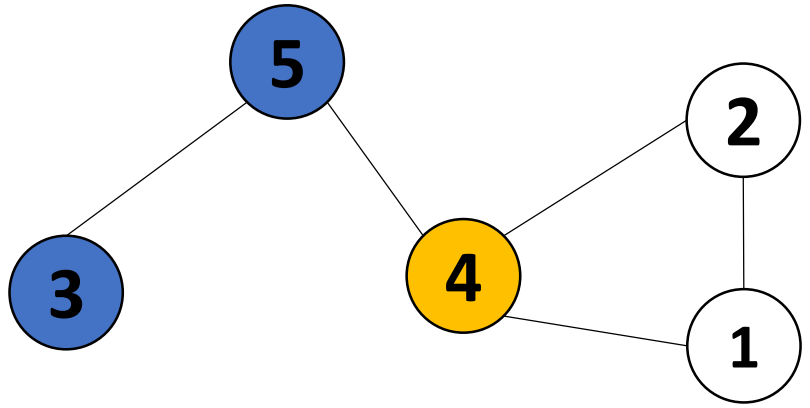


```java
private static int buscaEmLargura(No inicio, No fim){
    if (inicio == fim) return 0;
    LinkedList<No> ll = new LinkedList<No>();
    inicio.distancia = 0; inicio.visitado = true;
    ll.add(inicio);
    No atual;
    while (!ll.isEmpty()){
       atual = ll.remove();
       for (No temp: atual.vizinhos){
          if (!temp.visitado){
              if (temp == fim) return atual.distancia + 1;
              temp.visitado = true;
              temp.distancia = atual.distancia + 1;
              ll.add(temp);
          }
       }
    }
    return -1;
 }
```

buscando o nó 2

atual=3 ; dist=0

Fila: 4

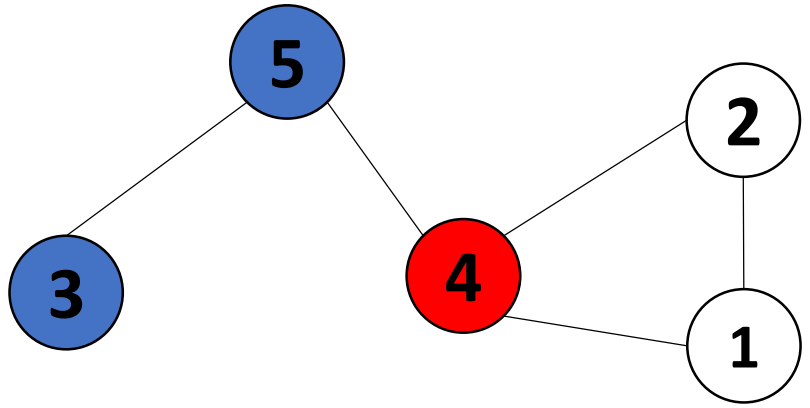


```java
private static int buscaEmLargura(No inicio, No fim){
    if (inicio == fim) return 0;
    LinkedList<No> ll = new LinkedList<No>();
    inicio.distancia = 0; inicio.visitado = true;
    ll.add(inicio);
    No atual;
    while (!ll.isEmpty()){
       atual = ll.remove();
       for (No temp: atual.vizinhos){
          if (!temp.visitado){
              if (temp == fim) return atual.distancia + 1;
              temp.visitado = true;
              temp.distancia = atual.distancia + 1;
              ll.add(temp);
          }
       }
    }
    return -1;
 }
```

buscando o nó 2

atual=4 ; dist=1

Fila:

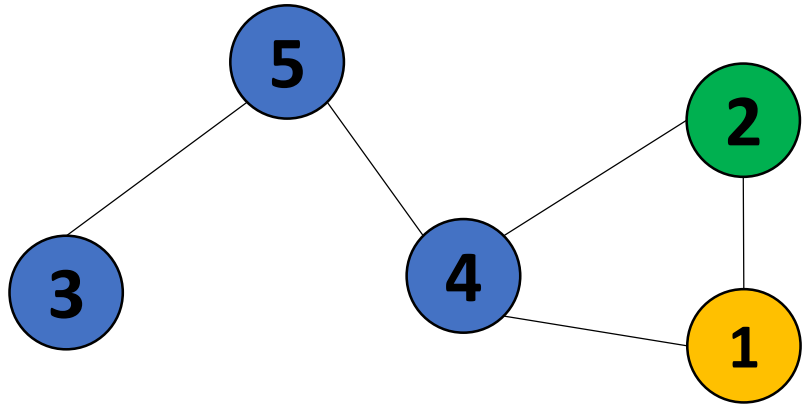


```java
private static int buscaEmLargura(No inicio, No fim){
    if (inicio == fim) return 0;
    LinkedList<No> ll = new LinkedList<No>();
    inicio.distancia = 0; inicio.visitado = true;
    ll.add(inicio);
    No atual;
    while (!ll.isEmpty()){
       atual = ll.remove();
       for (No temp: atual.vizinhos){
          if (!temp.visitado){
              if (temp == fim) return atual.distancia + 1;
              temp.visitado = true;
              temp.distancia = atual.distancia + 1;
              ll.add(temp);
          }
       }
    }
    return -1;
 }
```

buscando o nó 2

atual=4 ; dist=1

Fila: 1



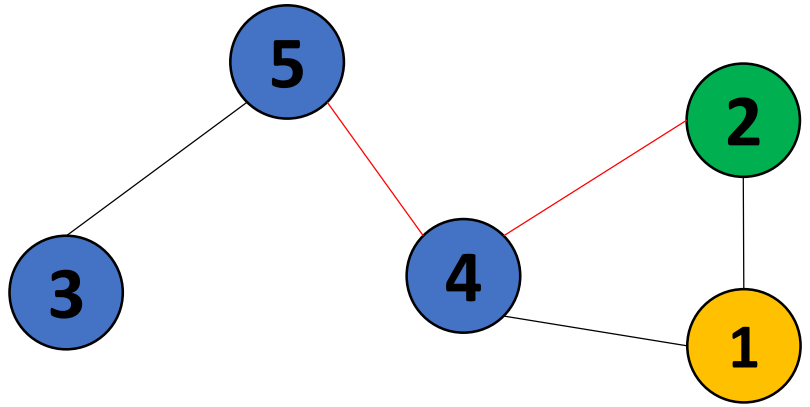
```java
private static int buscaEmLargura(No inicio, No fim){
    if (inicio == fim) return 0;
    LinkedList<No> ll = new LinkedList<No>();
    inicio.distancia = 0; inicio.visitado = true;
    ll.add(inicio);
    No atual;
    while (!ll.isEmpty()){
       atual = ll.remove();
       for (No temp: atual.vizinhos){
          if (!temp.visitado){
              if (temp == fim) return atual.distancia + 1;
              temp.visitado = true;
              temp.distancia = atual.distancia + 1;
              ll.add(temp);
          }
       }
    }
    return -1;
 }
```

buscando o nó 2

atual=4 ; dist=1

Fila: 1

Distância = 2



Grafos – Busca em Profundidade e em Largura

Prof. Luciano Antonio Digiampietri