Busca em grafos

TEG0001 - Teoria dos grafos

Prof. Dr. Ricardo José Pfitscher
ricardo.pfitscher@gmail.com

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Objetivos de aprendizagem

- Conhecer os algoritmos para busca em grafos
- Implementar o algoritmo de busca em profundidade
- Implementar o algoritmo de busca em largura

Aula passada: Exercícios (IME-USP)

1. Todos os caminhos. Faça uma lista de todos os caminhos simples com exatamente 4 vértices no grafo definido pelos arcos 7-3 1-4 7-8 0-5 5-2 3-8 2-9 0-6 4-9 2-6 6-4.
2. Todos os caminhos. Faça uma lista de todos os caminhos simples com exatamente 4 vértices no grafo não-dirigido definido pelas arestas 3-7 1-4 7-8 0-5 5-2 3-8 2-9 0-6 4-9 2-6 6-4.
3. Considere o grafo definido pelos arcos 0-1 1-2 2-0 2-3 3-1. A sequência 0-1-2-3-1-2-0 é um ciclo?
4. Escreva uma função booleana que verifique se uma sequência seq[0..k] de vértices de um grafo é um ciclo

Implementação - atividade - relembrando

- Desenvolva um programa para implementar a TDA grafo.
  - Utilizar lista de adjacências
  - O grafo deve ser ponderado
  - Oferecer um menu ao usuário com as seguintes opções:
    i. Adicionar vértices e arestas
    ii. Calcular o grau de um dado vértice
    iii. Mostrar os conjuntos de vértices (V) e arestas (E)
    iv. Responder se um vértice é alcançável **diretamente** a partir de outro
    v. Responder se um vértice é alcançável a partir de outro

Como poderíamos resolver isso?

Implementação - atividade

- Os dois últimos itens do exercício requerem que seja realizada uma **busca** no grafo
  a. Responder se um vértice é alcançável **diretamente** a partir de outro
  b. Responder se um vértice é alcançável a partir de outro

Busca em Grafos

- De IME-USP:

  “Um algoritmo de busca é um algoritmo que esquadrinha um grafo andando pelos arcos de um vértice a outro. Depois de visitar a ponta inicial de um arco, o algoritmo percorre o arco e visita sua ponta final. Cada arco é percorrido no máximo uma vez.”
- Utilidade:
  - Verificar se dois pontos da rede estão conectados
  - P. ex.: é possível chegar à um destino no mapa?
- Objetivos:
  - Encontrar tudo que pode ser encontrado (que possui um caminho) a partir de um vértice de início
  - Não passar por um vértice duas vezes em tempo linear -- O(n+m)

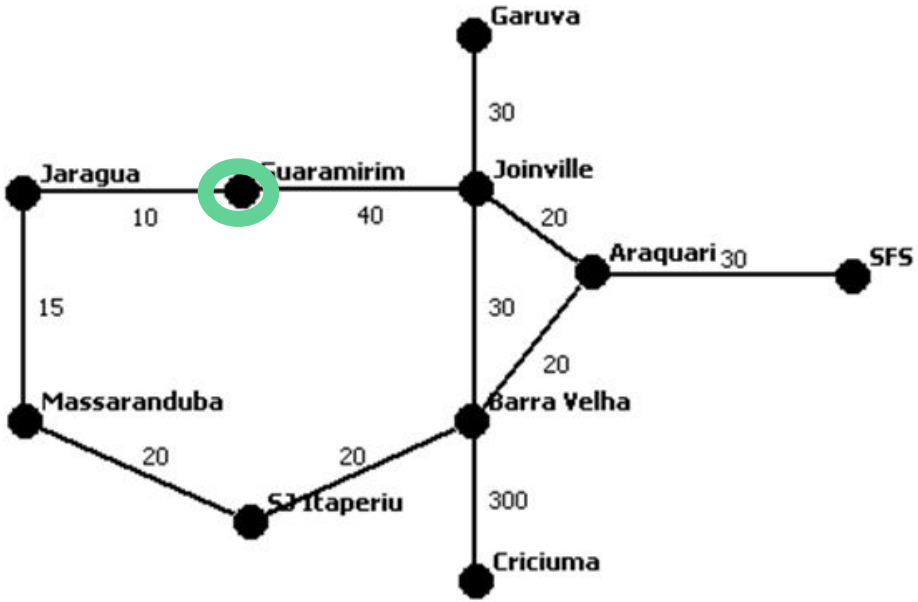
Busca em Grafos - Algoritmo Genérico

- Ideia geral: dividir o grafo entre território conquistado (explorado) e não conquistado (inexplorado) e avançar um nó de cada vez

```
genSearch(grafo G,vértice s){
- inicializar s como explorado e o resto dos vértices como não explorados
- enquanto possível
   - selecionar uma aresta (u,v) com u explorado e v inexplorado.
   - marcar v como explorado
}
```

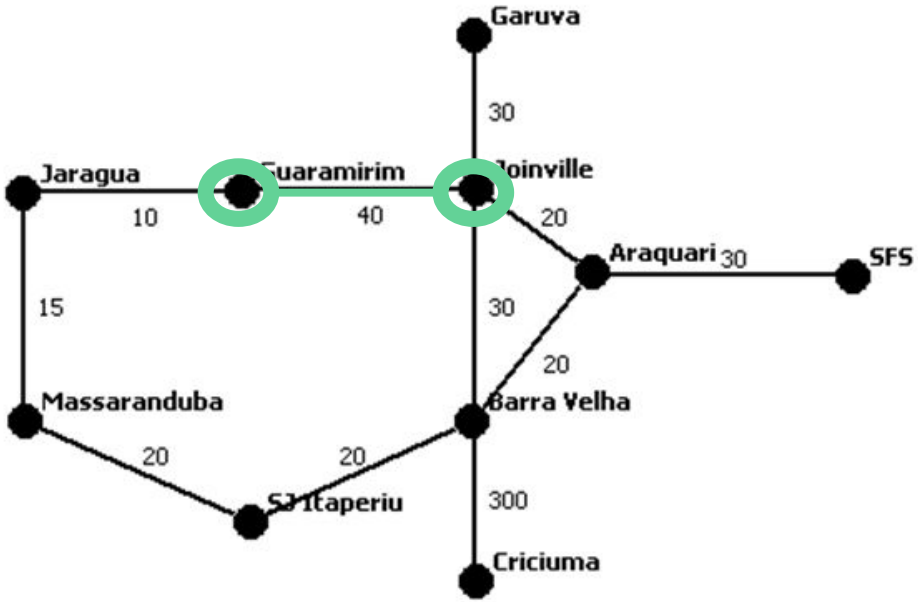
Como fazer isso?

Busca em Grafos - Algoritmo Genérico



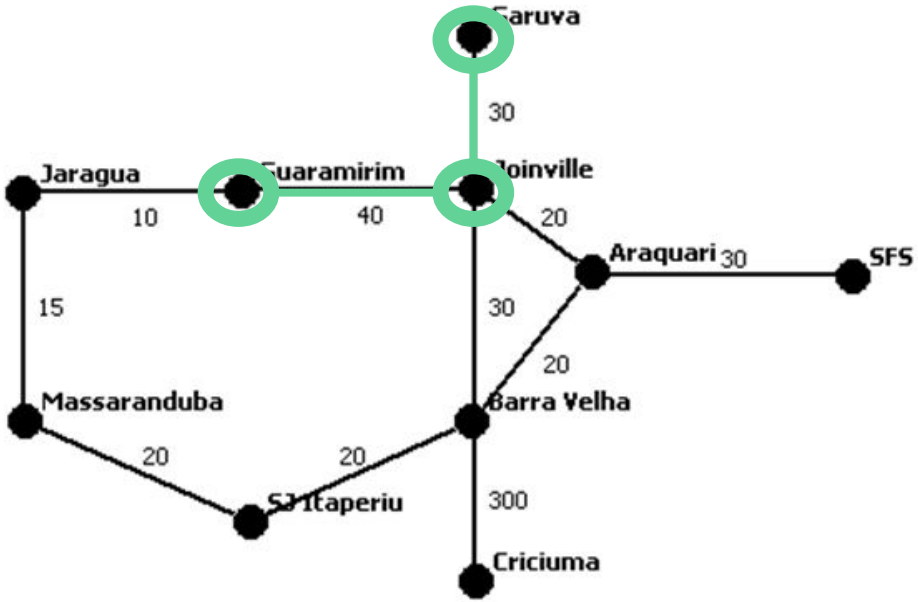

```
genSearch(grafo G,vértice s){
    - s ← explorado
    - enquanto possível
        - selecionar uma
        aresta (u,v) com u
        explorado e v
        inexplorado.
        - v ← explorado
}
```

Busca em Grafos - Algoritmo Genérico



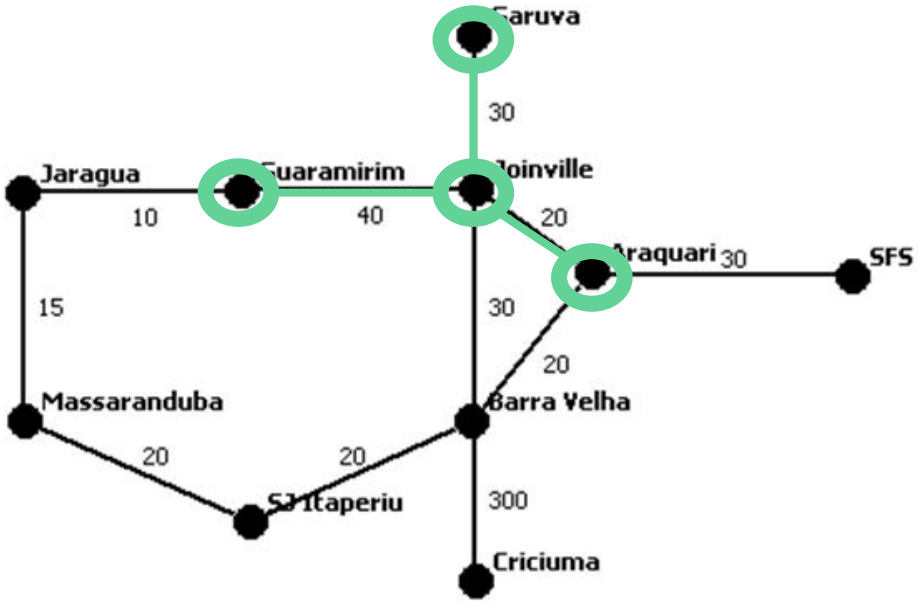

```
genSearch(grafo G,vértice s){
    - s ← explorado
    - enquanto possível
        - selecionar uma
        aresta (u,v) com u
        explorado e v
        inexplorado.
        - v ← explorado
}
```

Busca em Grafos - Algoritmo Genérico



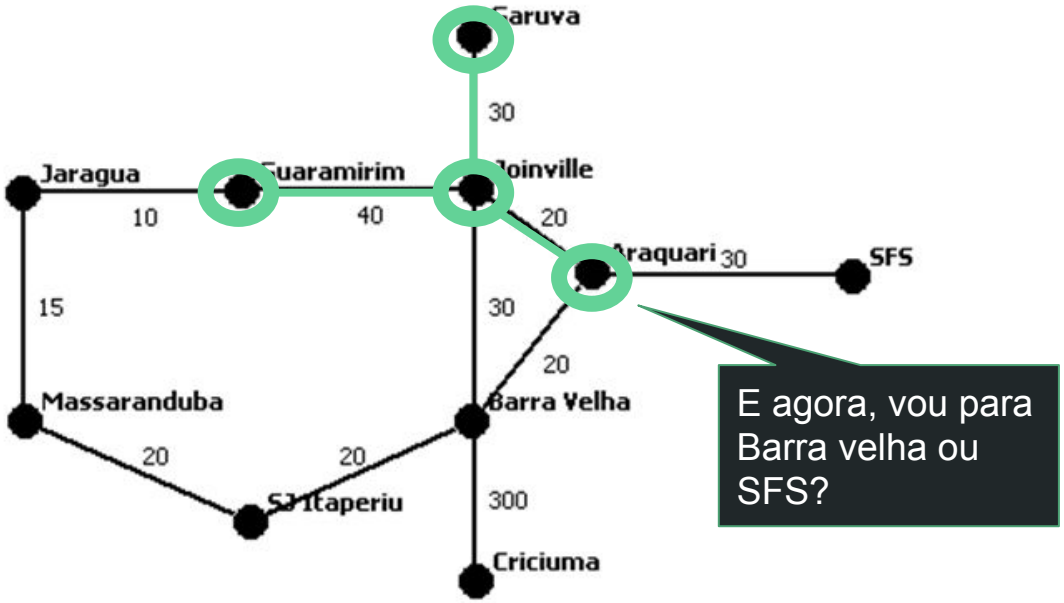

```
genSearch(grafo G,vértice s){
    - s ← explorado
    - enquanto possível
        - selecionar uma
        aresta (u,v) com u
        explorado e v
        inexplorado.
        - v ← explorado
}
```

Busca em Grafos - Algoritmo Genérico



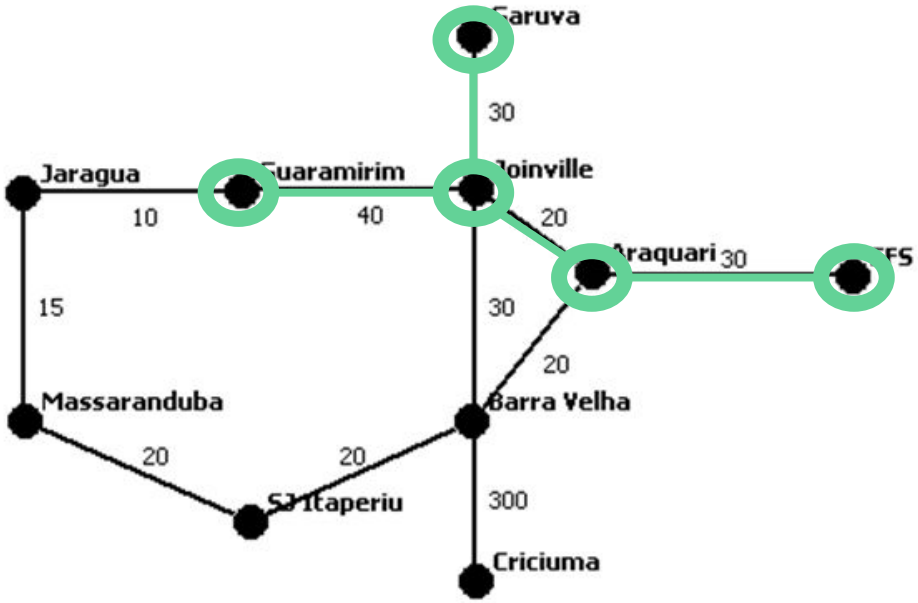

```
genSearch(grafo G,vértice s){
    - s ← explorado
    - enquanto possível
        - selecionar uma
        aresta (u,v) com u
        explorado e v
        inexplorado.
        - v ← explorado
}
```

Busca em Grafos - Algoritmo Genérico



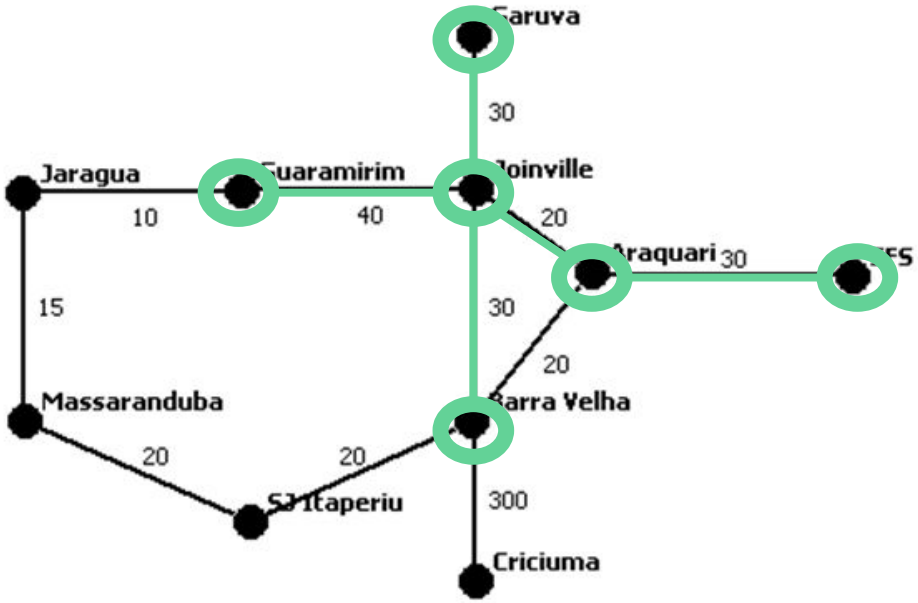

```
genSearch(grafo G,vértice s){
    - s ← explorado
    - enquanto possível
        - selecionar uma aresta (u,v) com u explorado e v inexplorado.
        - v ← explorado
}
```

Busca em Grafos - Algoritmo Genérico



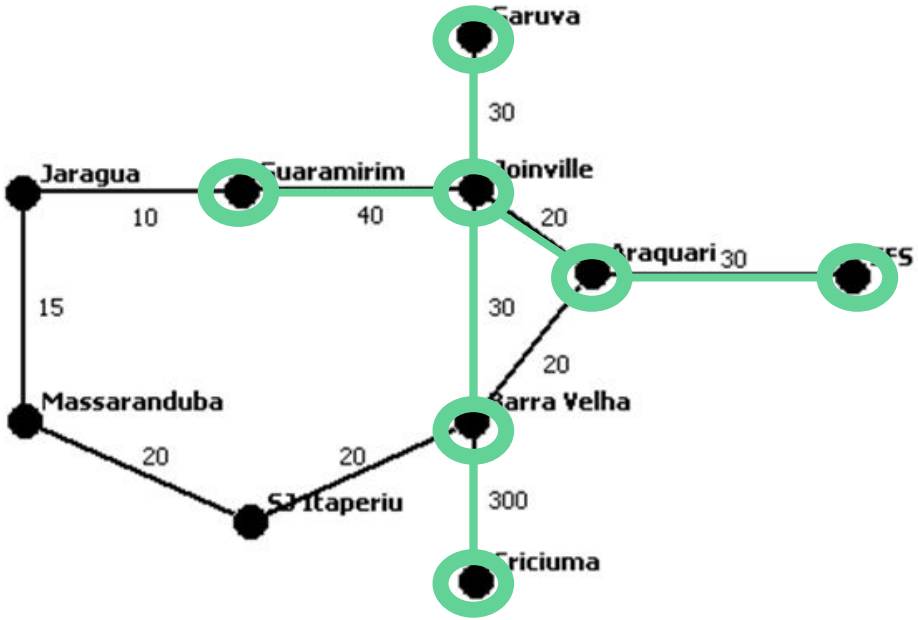

```
genSearch(grafo G,vértice s){
    - s ← explorado
    - enquanto possível
        - selecionar uma
        aresta (u,v) com u
        explorado e v
        inexplorado.
        - v ← explorado
}
```

Busca em Grafos - Algoritmo Genérico



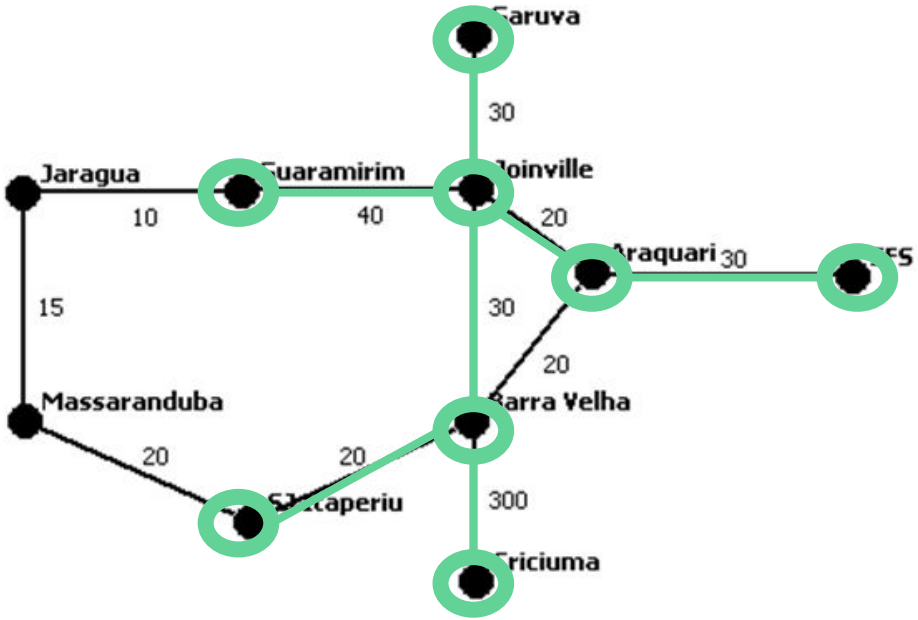

```
genSearch(grafo G,vértice s){
    - s ← explorado
    - enquanto possível
        - selecionar uma
        aresta (u,v) com u
        explorado e v
        inexplorado.
        - v ← explorado
}
```

Busca em Grafos - Algoritmo Genérico



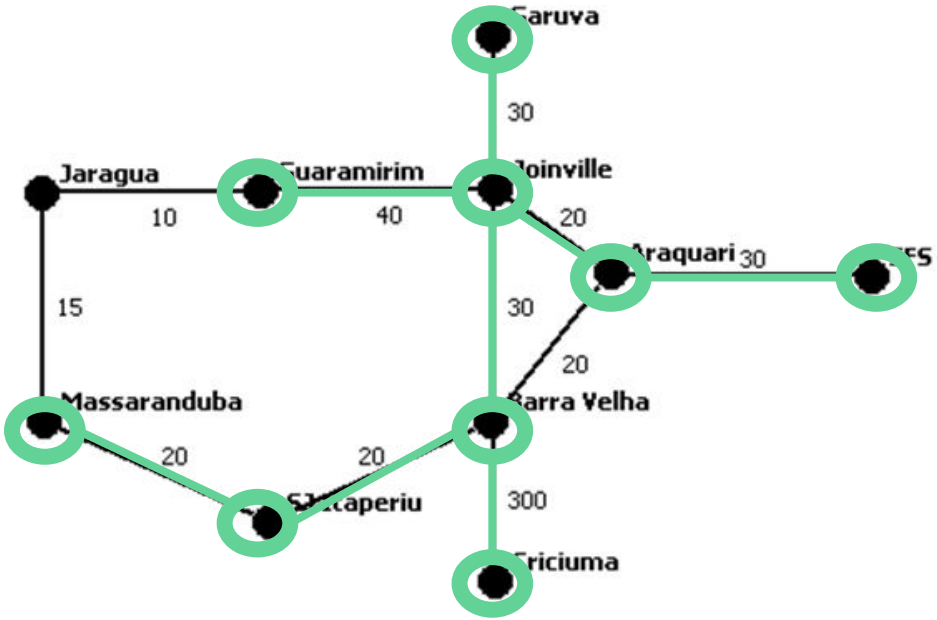

```
genSearch(grafo G,vértice s){
    - s ← explorado
    - enquanto possível
        - selecionar uma
        aresta (u,v) com u
        explorado e v
        inexplorado.
        - v ← explorado
}
```

Busca em Grafos - Algoritmo Genérico



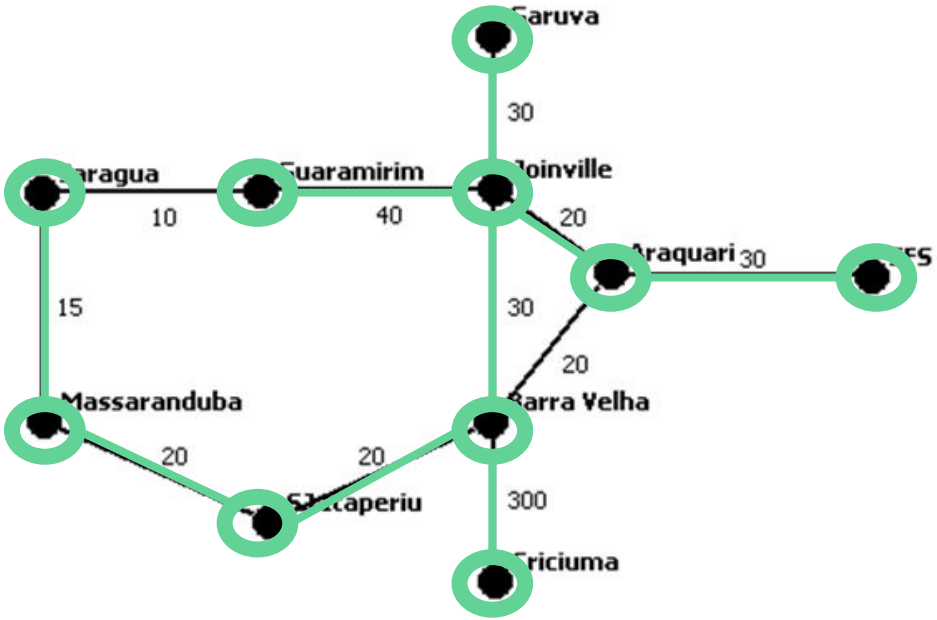

```
genSearch(grafo G,vértice s){
    - s ← explorado
    - enquanto possível
        - selecionar uma
        aresta (u,v) com u
        explorado e v
        inexplorado.
        - v ← explorado
}
```

Busca em Grafos - Algoritmo Genérico




```
genSearch(grafo G,vértice s){
    - s ← explorado
    - enquanto possível
        - selecionar uma
        aresta (u,v) com u
        explorado e v
        inexplorado.
        - v ← explorado
}
```

Busca em Grafos - Algoritmo Genérico




```
genSearch(grafo G,vértice s){
    - s ← explorado
    - enquanto possível
        - selecionar uma
        aresta (u,v) com u
        explorado e v
        inexplorado.
        - v ← explorado
}
```

Busca em Grafos

- Existem duas formas de varrer o grafo:
  - Em largura ou em profundidade
- A diferença está na escolha do próximo nó não visitado
  - Se o algoritmo aprofunda (vai para um vértice mais longe do anterior) é busca em profundidade (DFS - Deep First Search)
  - Se o algoritmo abrange (vai para um vértice vizinho ao anterior) é busca em largura (BFS - Breadth First Search)
- Em termos de implementação, a diferença está na estrutura de dados auxiliar que registra os nós vizinhos e retira os nós a serem visitados
  - BFS - Fila
  - DFS - Pilha

Busca em Grafos - Largura - BFS

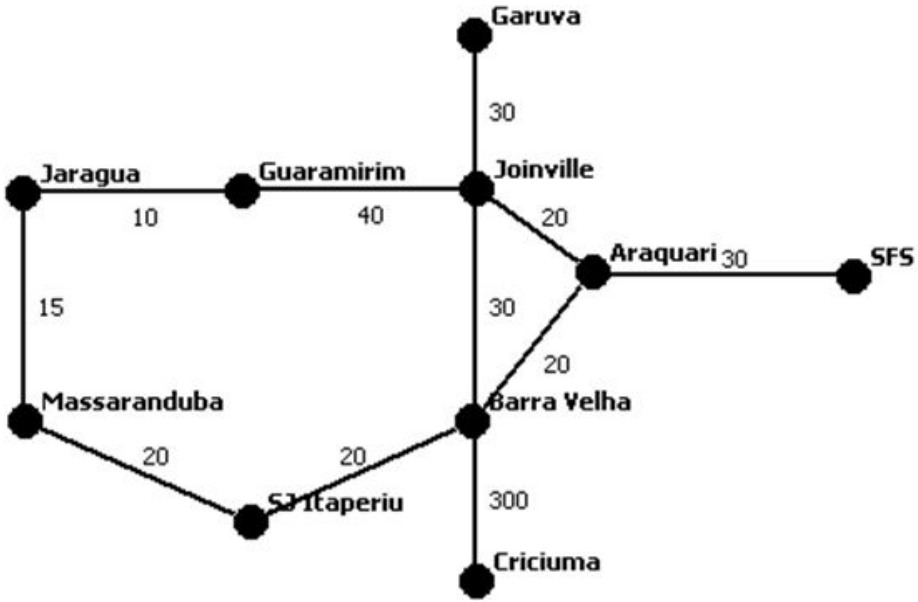
- Breadth First Search (BFS) – Busca em largura
  - Dados um grafo G = (V, E) e um vértice s, chamado de fonte, a busca em largura sistematicamente explora as arestas de G de maneira a visitar todos os vértices alcançáveis a partir de s.
  - Explora os nós em forma de layers (camadas)
  - Pode encontrar o caminho mais curto (shortest path)
    - Desde que sejam anotados os pesos percorridos
  - Pode calcular os componentes conexos em um grafo não direcionado
  - O(m + n) se implementado usando uma fila (FIFO)

Busca em Grafos - Largura - BFS

```
bfs(graph G, vertex s):
    setAllUnexplored(G); //Marca todos como não explorados
    s.explored = true;
    Queue Q; //Cria fila
    Q.put(s);
    while(Q.size != 0):
        v = Q.get();
        for each edge (v,w) where w is unexplored:
            w.explored = true;
            Q.put(w); //Insere no final da fila
```

O(N^2) se matriz de adjacência ou O(N+M) se lista de adjacências

Bus[illegible]s - BFS




```
bfs(graph G, vertex s):
    setAllUnexplored(G);
    s.explored = true;
    Queue Q; //Cria fila
    Q.put(s);
    while(Q.size != 0):
        v = Q.get();
        for each edge (v,w) where w is unexplored:
            w.explored = true;
            Q.put(w); //Insere no final da fila
```

Exercício

- Implemente o algoritmo de busca em largura para a sua classe grafo

Busca em Grafos - profundidade - DFS

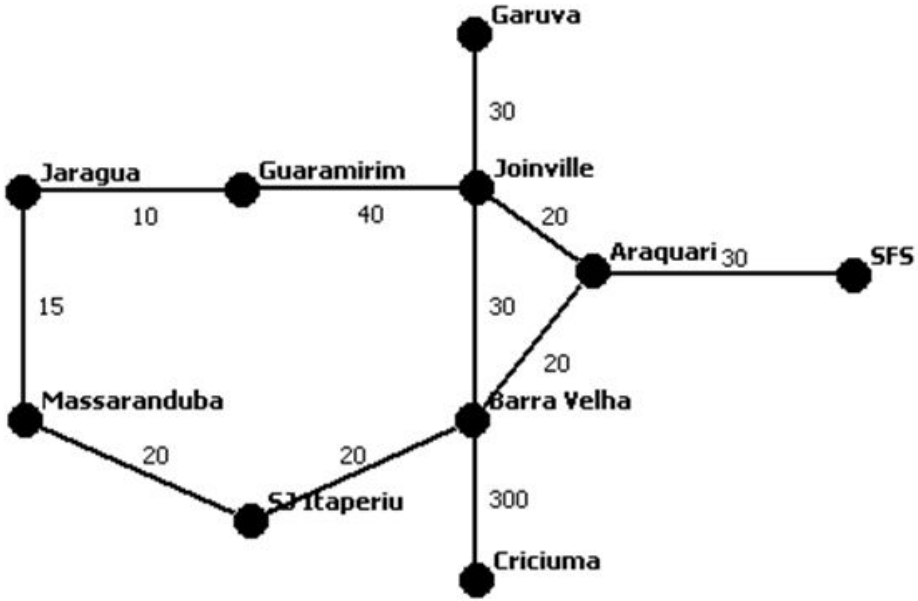
- Depth First Search (DFS) – Busca em profundidade
  - As arestas são exploradas a partir do vértice v **mais recentemente descoberto** que ainda possui arestas não exploradas saindo dele
  - É uma busca mais agressiva, faz *backtrack* apenas quando realmente necessário. (Ex: Você tentando achar a saída de um labirinto)
  - Calcula os componentes conexos em um grafo direcionado.
  - O(m + n) se implementado usando uma **pilha (LIFO)**

Busca em Grafos - profundidade - DFS

```
dfs(graph G, vertex s):
    setAllUnexplored(G);//Marca todos como não explorados
    Stack P; //Cria pilha
    P.push(s);
    while(P.size != 0):
        v = P.pop();
        v.explored = true;
        for each edge(v,u) where u is unexplored:
            P.push(u);
```

Busca em Grafos - DFS

```
dfs(graph G, vertex s):
    setAllUnexplored(G);
    Stack P; //Cria pilha
    P.push(s);
    while(P.size != 0):
        v = P.pop();
        v.explored = true;
        for each edge(v,u) where u is unexplored:
            P.push(u);
```



Exercício

- Implemente o algoritmo de busca em profundidade para a sua classe grafo
- Pense em como modificar a implementação das buscas para permitir o cálculo dos menores caminhos até todos os destinos