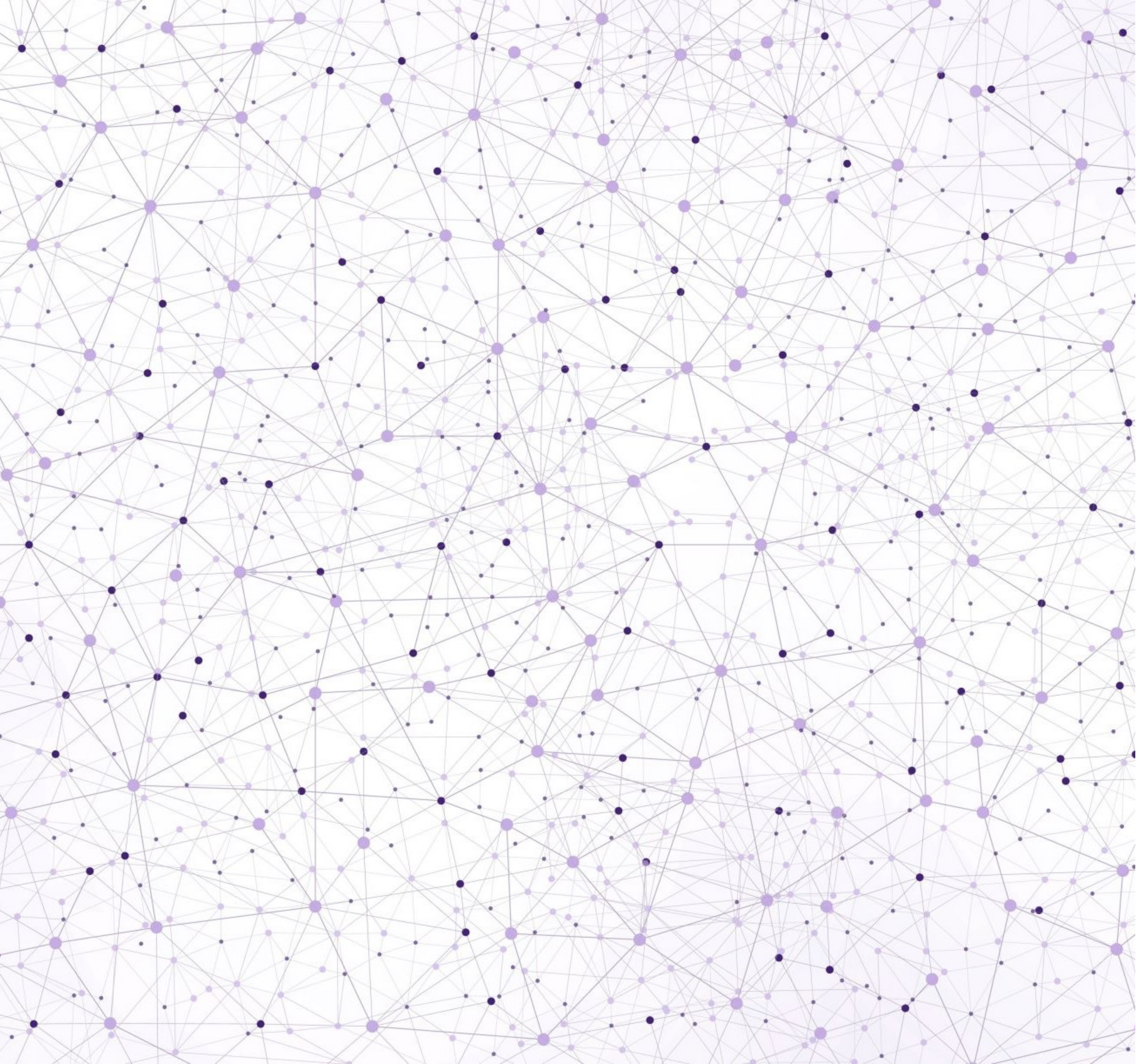
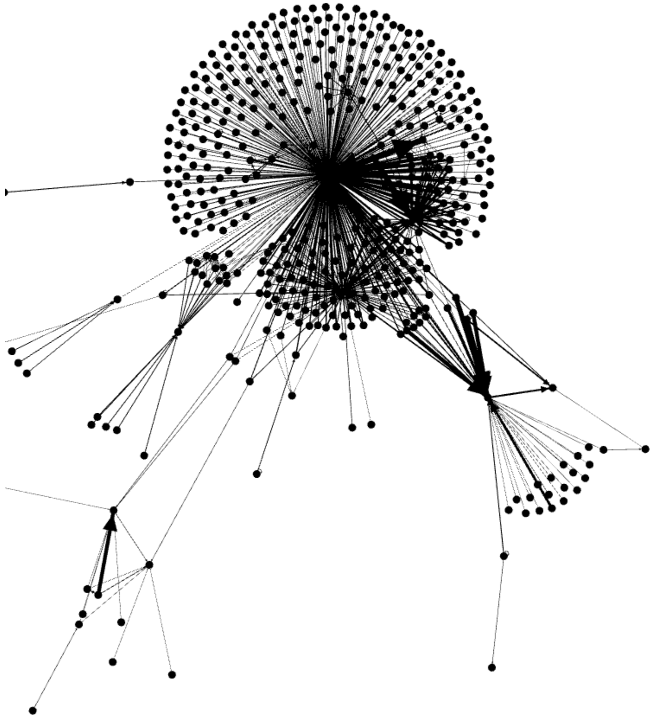


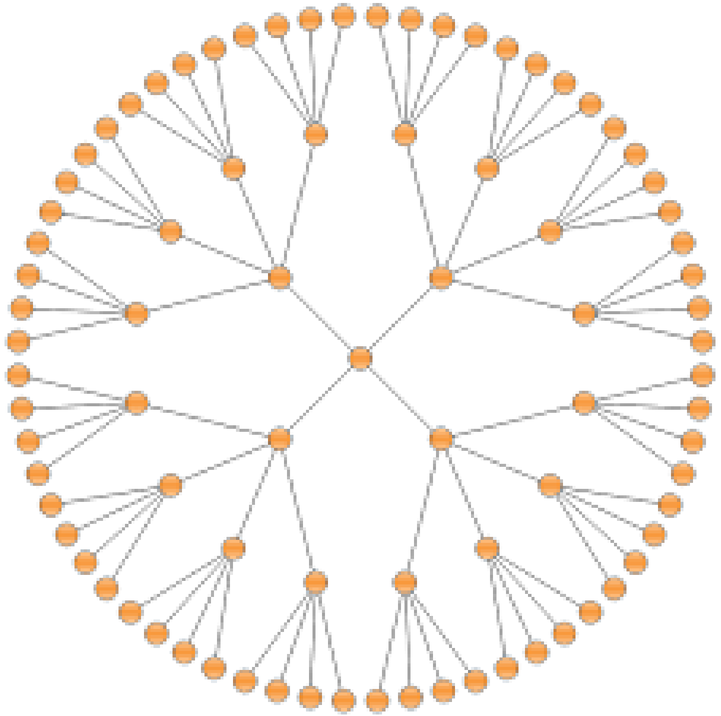
PERCURSO EM GRAFOS BUSCA EM LARGURA

PERCURSO EM GRAFO



- Percorrer um grafo consistem em seguir as arestas de maneira sistemática de forma a visitar (ou descobrir) seus vértices
- As travessias em grafos ajudam a descobrir bastante a respeito da sua estrutura, além de ser base para outros algoritmos
- Dois tipos de Travessia serão examinados:
  - BFS (Breadth First Search – Busca em Largura)
  - DFS (Depth First Search – Busca em Profundidade)

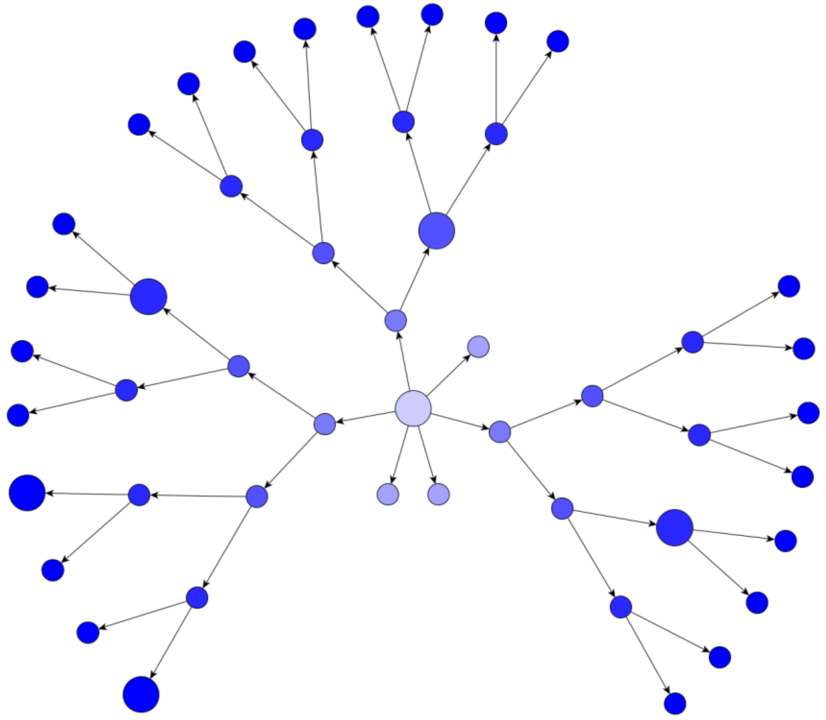
BUSCA EM LARGURA

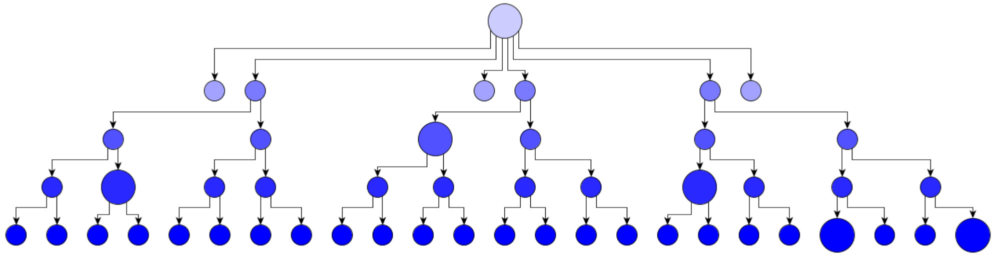


- Breadth-First Search
  - A partir de um vértice inicial ‘S’, descobrir todos os vértices alcançáveis a partir de ‘S’
  - “em camadas”
- Estratégia
  - Visitar todos os vértices que se encontram a uma distância ‘k’ de S **antes** de visitar os vértices que se encontram a uma distância ‘k+1’

BUSCA EM LARGURA

- Uma vez executado o algoritmo, o resultado é
  - Encontra-se o caminho mais curto (em número de vértices) a partir de ‘S’ para todos os demais vértices alcançáveis
  - Uma BFS-Tree é construída.





O ALGORITMO BFS

- Simples
- Bastante Intuitivo
- Ideia
  - A partir do vértice Inicial
  - Visita os vértices mais próximos
  - Em seguida visita os vértices seguintes

BFS

- Artifício*
  - Para manter controle dos vértices visitados, usaremos um artifício: 3 cores (branco, cinza, preto)
  - Branco – vértices não visitados. Inicialmente todos são brancos
  - Cinza – vértices que já foram alcançados, mas ainda não foram processados
  - Preto – vértices já processados

ALGORITMO BFS

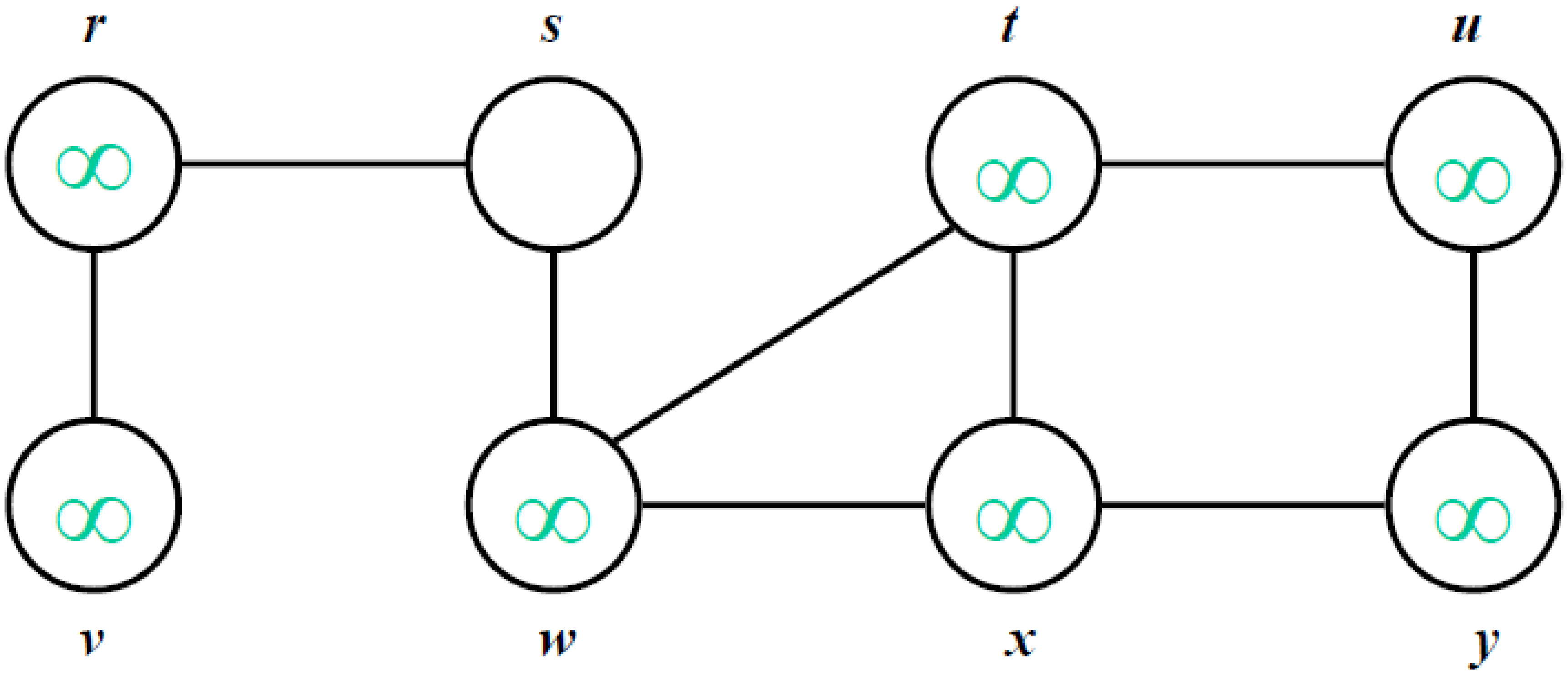
- Usa estruturas auxiliares
  - Cor[] – mantém a informação da cor de cada vértice
  - ∏[] – mantém a informação do vértice predecessor (pai) do vértice
  - D[] = mantém a informação da distância do vértice em relação ao vértice inicial
  - Uma FILA “Q” para gerenciar a lista de vértices cinza

ALGORITMO BFS

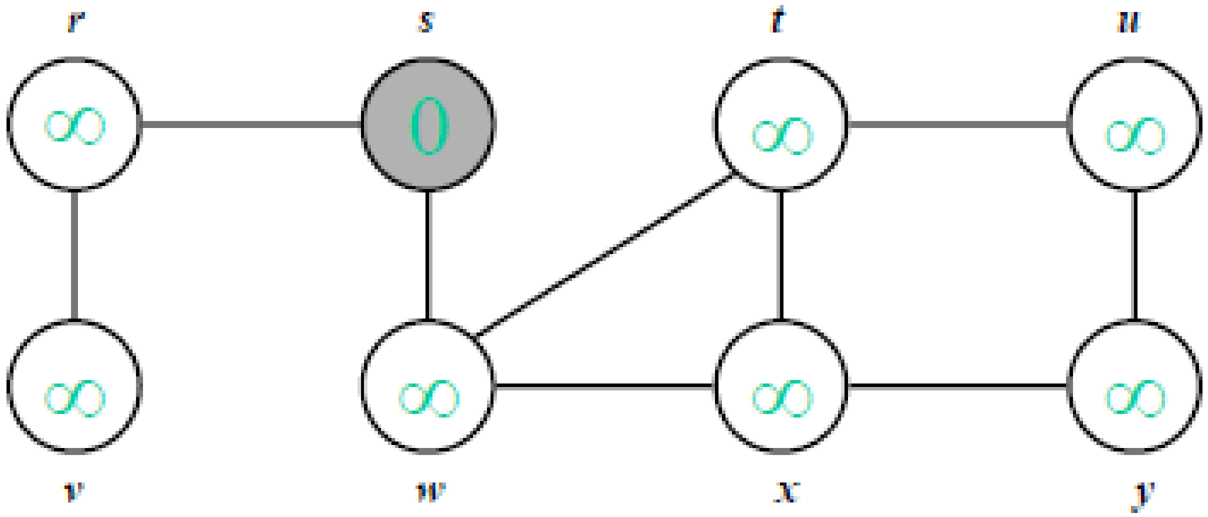
```
BFS(G, s)
for ∀u ∈ V[G] - {s} do
    cor[u] ← BRANCO
    D[u] ← ∞
    π[u] ← NIL
cor[s] ← CINZA
D[s] ← 0
π[s] ← NIL
Q ← {s} //Enfileira o S
```

```
While Q ≠ Ø do
    u ← Desenfileira[Q]                    O(1)
    for ∀v ∈ Adj[u] d            O(|Adj|)
        if cor[v] = BRANCO then
            cor[v] ← CINZA
            D[v] ← d[u] + 1
            π[v] ← u
            Enfileira(Q, v)
    cor[u] ← PRETO
```

ALGORITMO

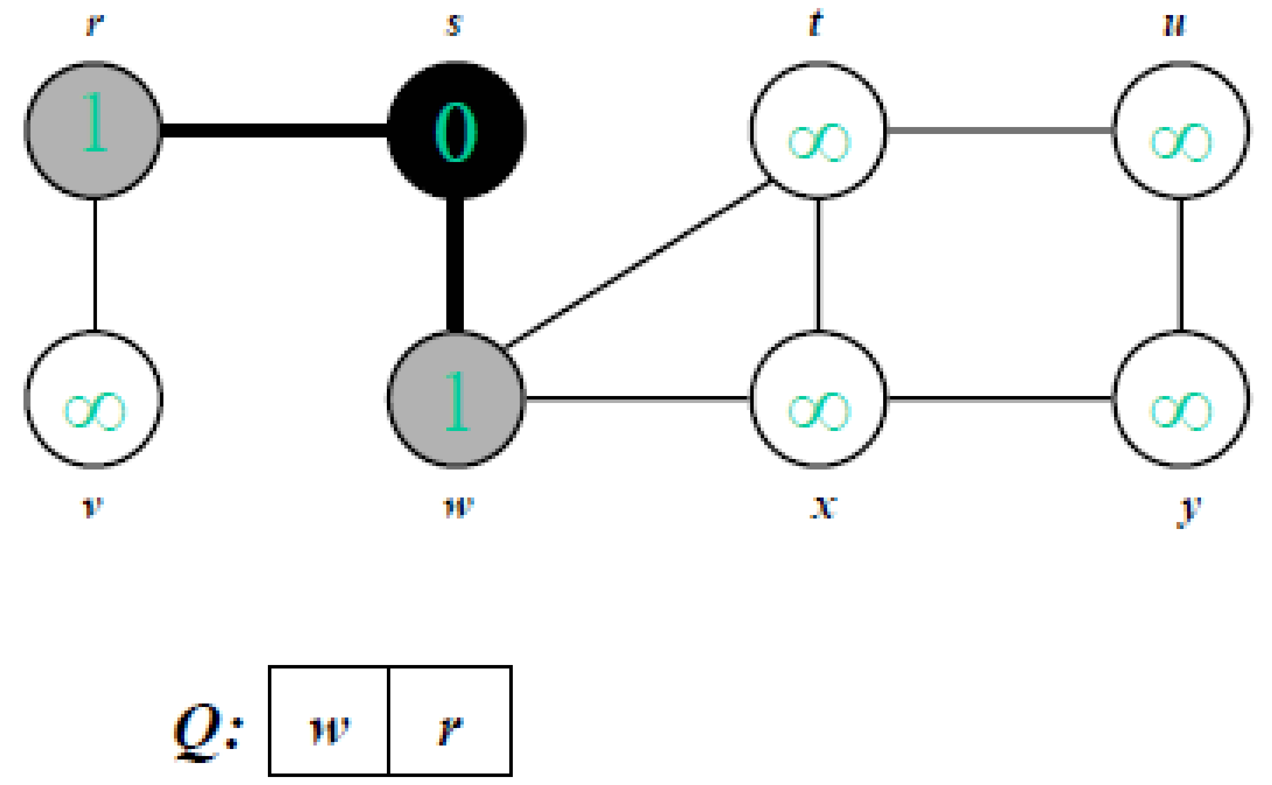


ALGORITMO

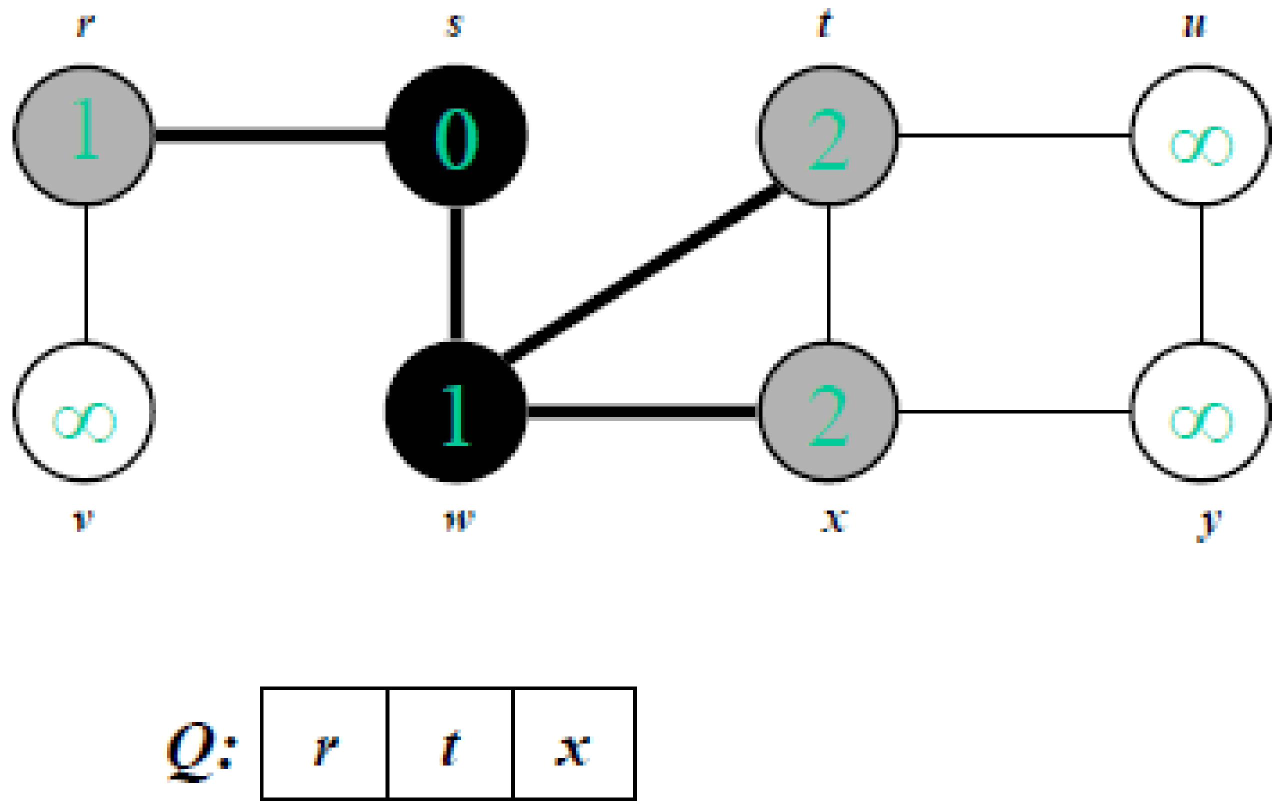


Q: s

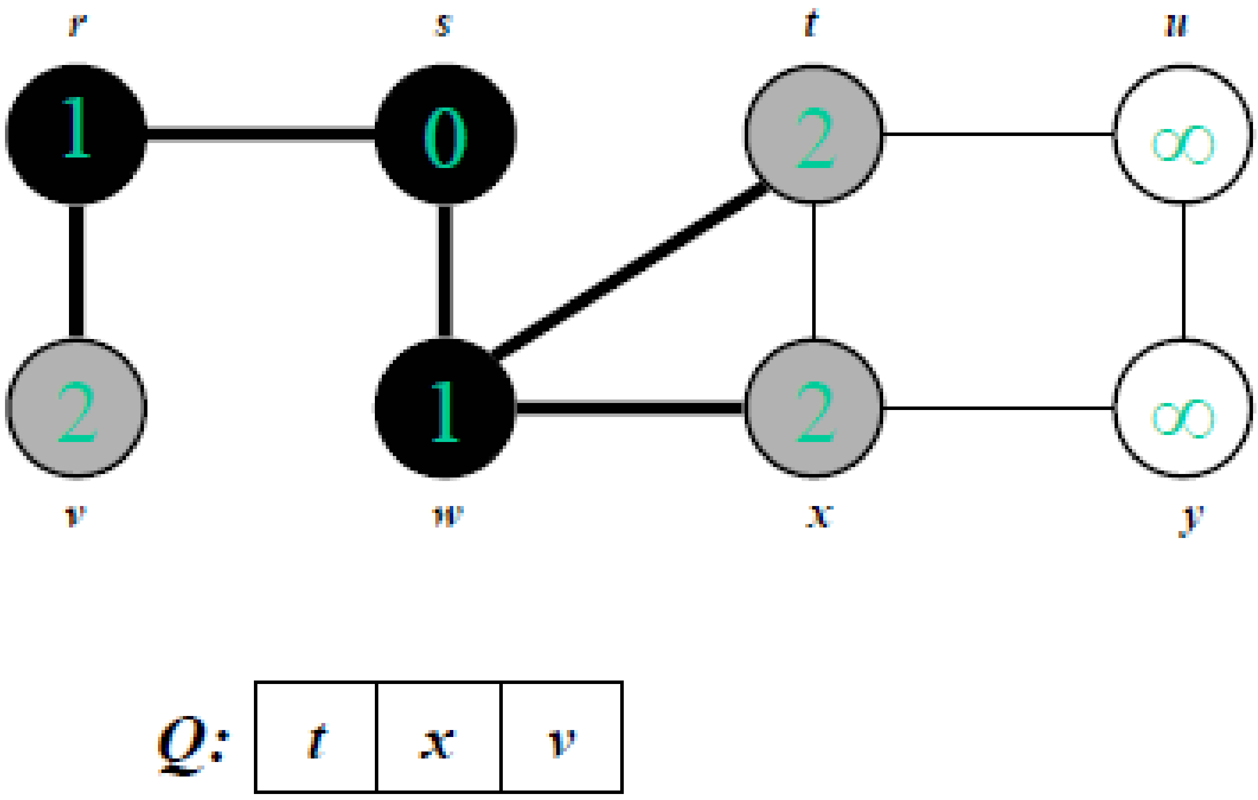
ALGORITMO



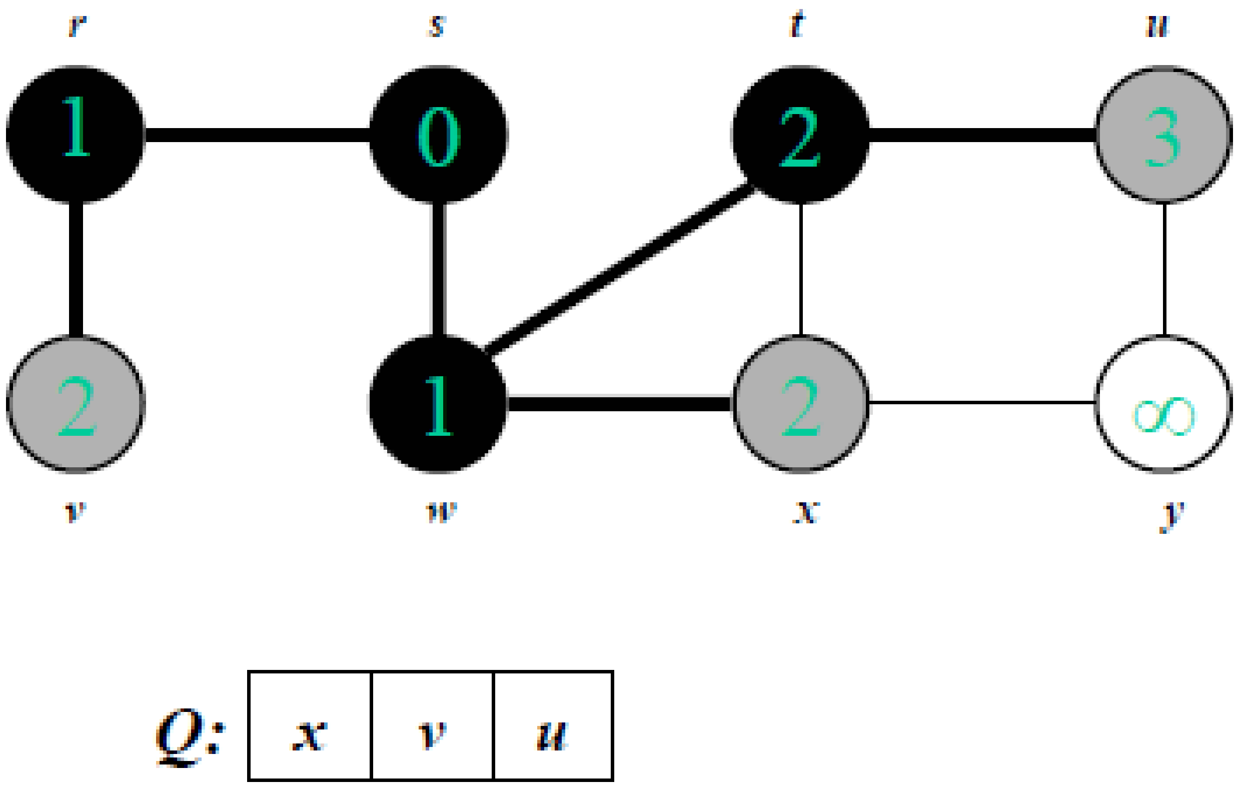
ALGORITMO



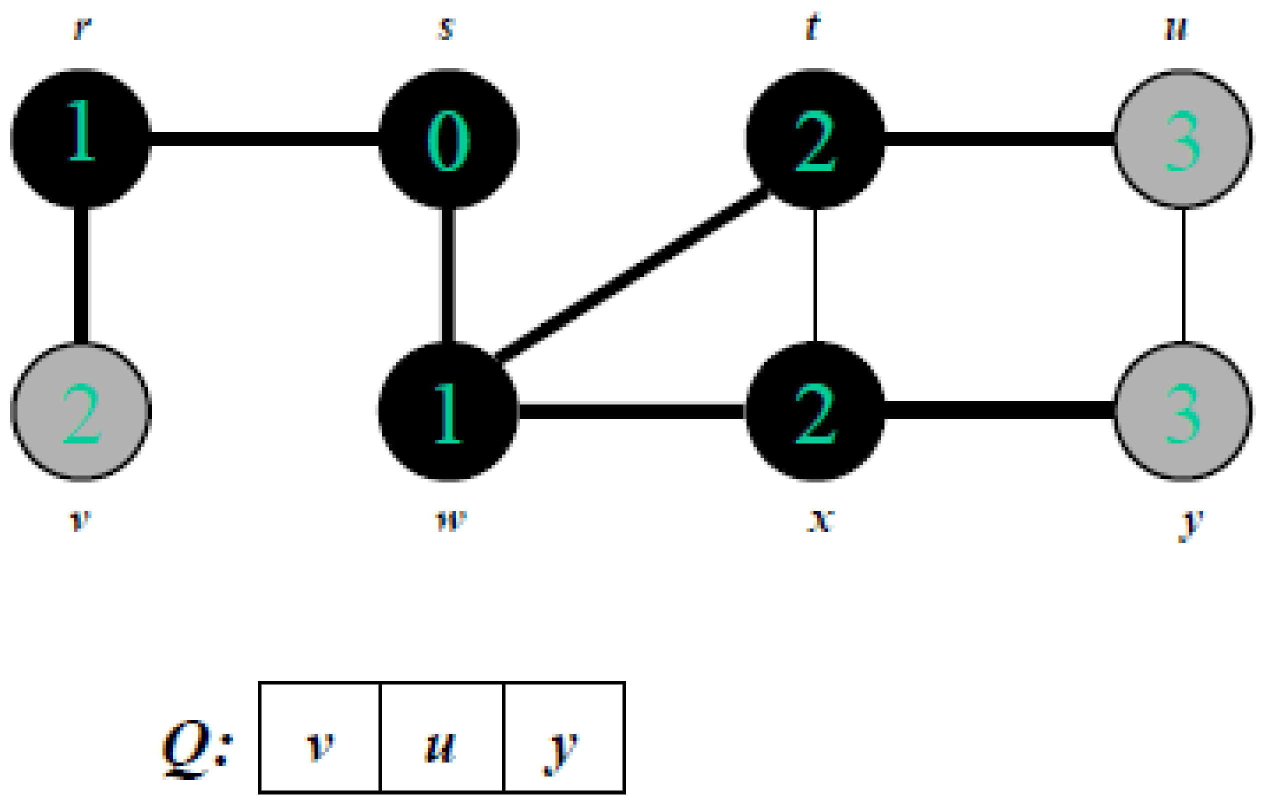
ALGORITMO



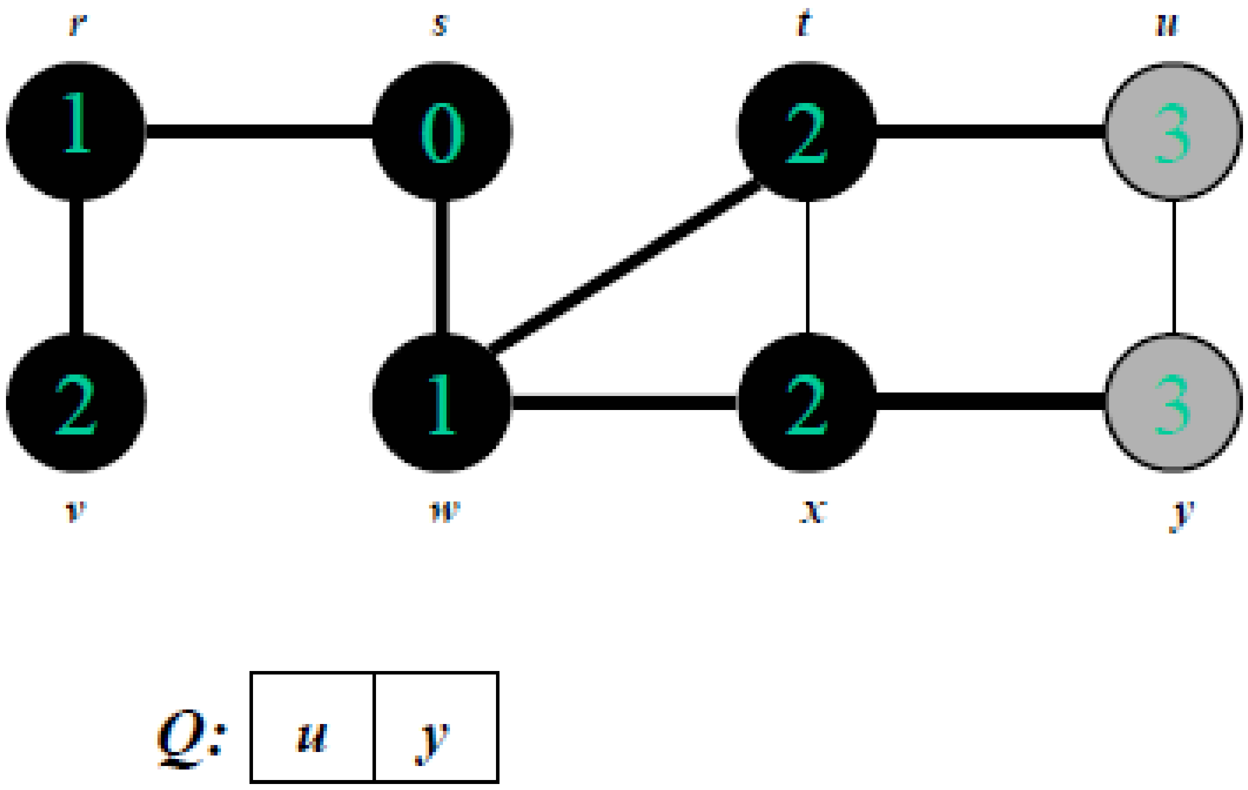
ALGORITMO



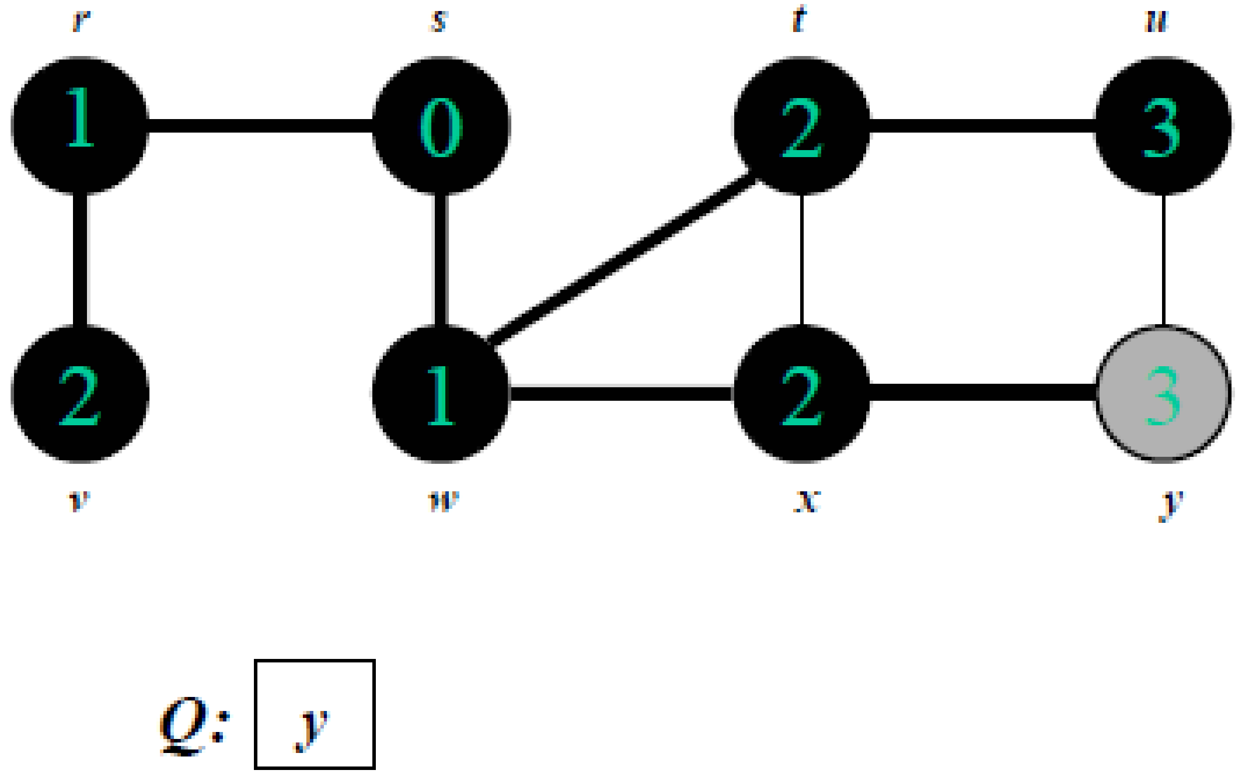
ALGORITMO



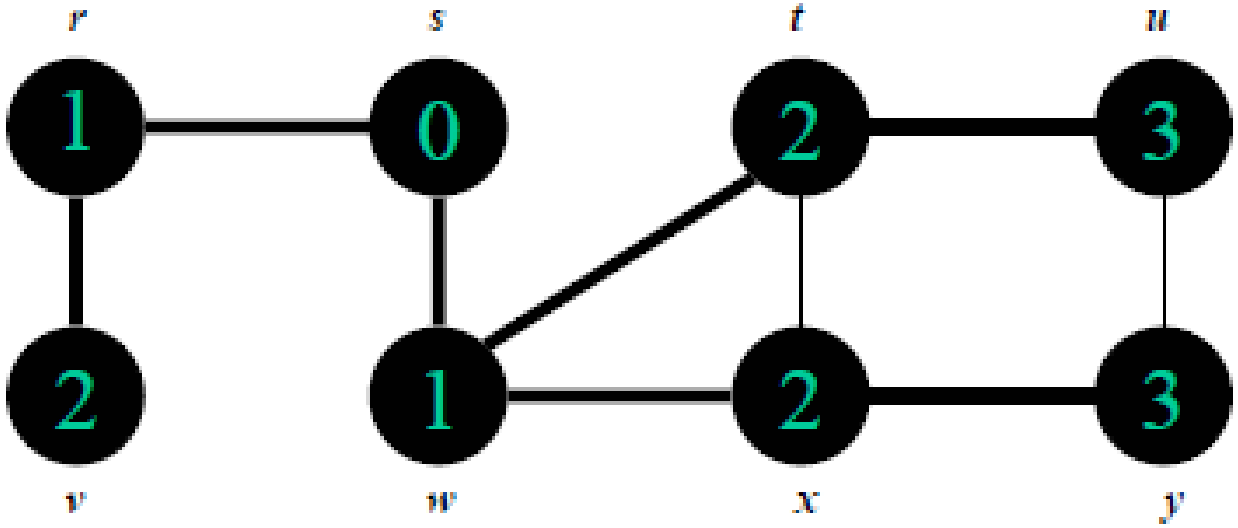
ALGORITMO



ALGORITMO



ALGORITMO



Q: Ø

COMPLEXIDADE DO ALGORITMO

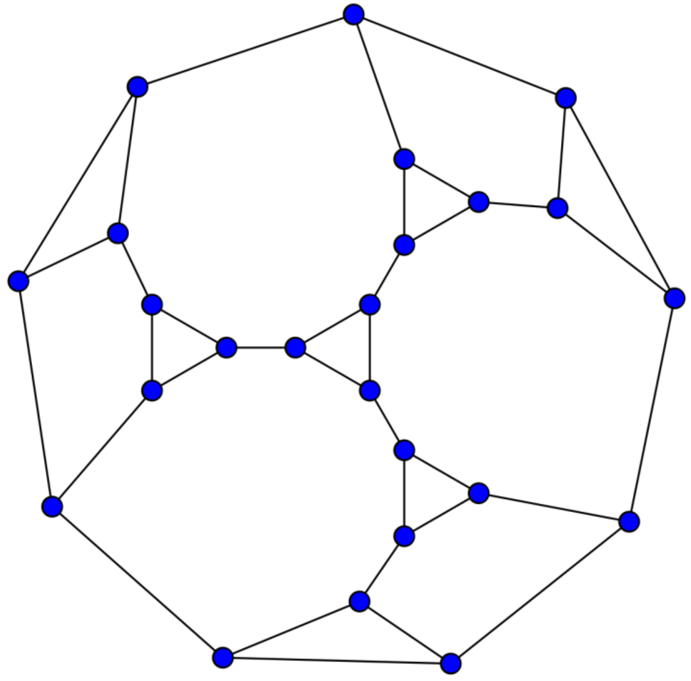
- Considerando um grafo representado em listas de adjacência
- No pior caso, a complexidade de tempo pode ser representada por O(E+V)
  - 1º laço é da ordem de |V| – (Para todas os vértices do grafo, atribuir ‘branco’)
  - No 2º laço a fila é acessada “V” vezes (pinta todos os vértices)
    - Para cada vértice, acessa todos os adjacentes. As operações nos adjacentes são O(1).
    - Esse segundo laço é executado E vezes

Por isso, a complexidade é O(V+E), ou seja – o conjunto de maior cardinalidade (arestas ou vértices)

COMPLEXIDADE DO ALGORITMO

- Considerando uma Matriz de Adjacência
  - Para "pintar" todas os vértices de branco : V
  - Para descobrir a existência de todas as arestas é preciso percorrer a matriz inteira, ou seja V^2

USOS



- Co-Authorship Distance Computation (https://www.csauthors.net/distance)
- Menor caminho entre 1 nó raiz e os demais nós do grafo
- Erdos, kevin bacon (https://oracleofbacon.org/)
- https://www.urionlinejudge.com.br/judge/pt/problems/view/2249 (Erdos)
- Dengue (OBI 2001) – Moodle*

LINK PARA VISUALIZADOR

- [A Visual Guide to Graph Traversal Algorithms by Workshape.io]