UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA
NÚCLEO DE COMPUTAÇÃO - NUCOMP

Algoritmos e Estruturas de Dados II

Busca em largura largura em grafos

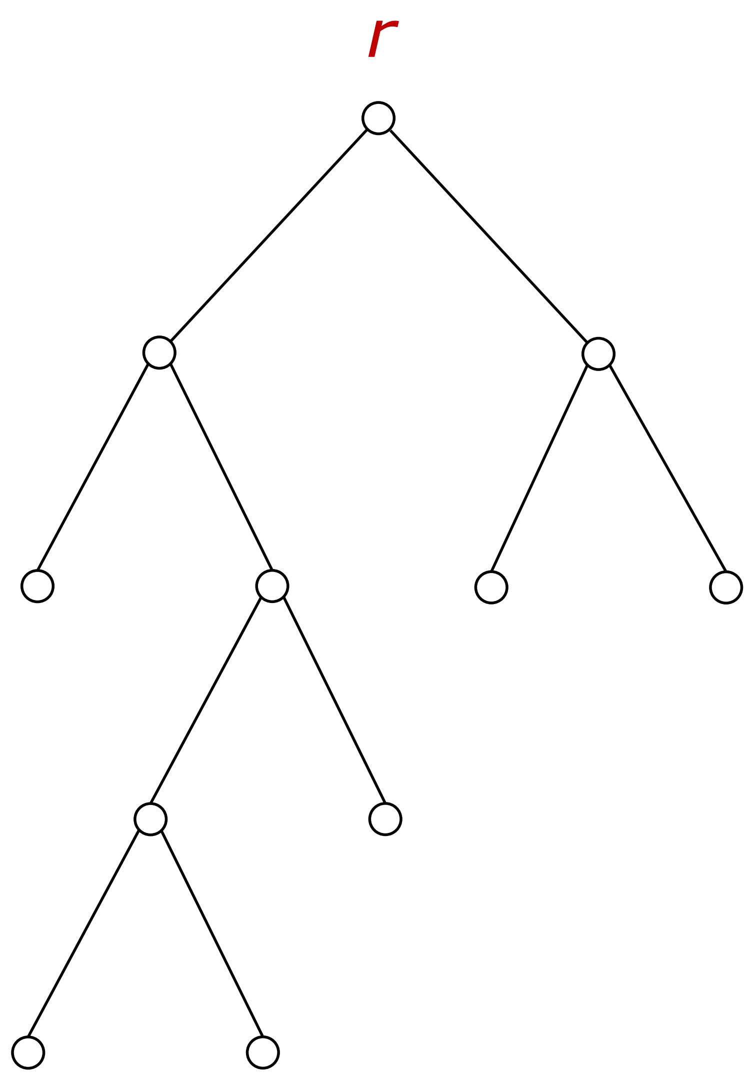
Prof. Flávio José Mendes Coelho
fcoelho@uea.edu.br

Busca em largura

(*breadth-first search - BFS*)

Busca em largura

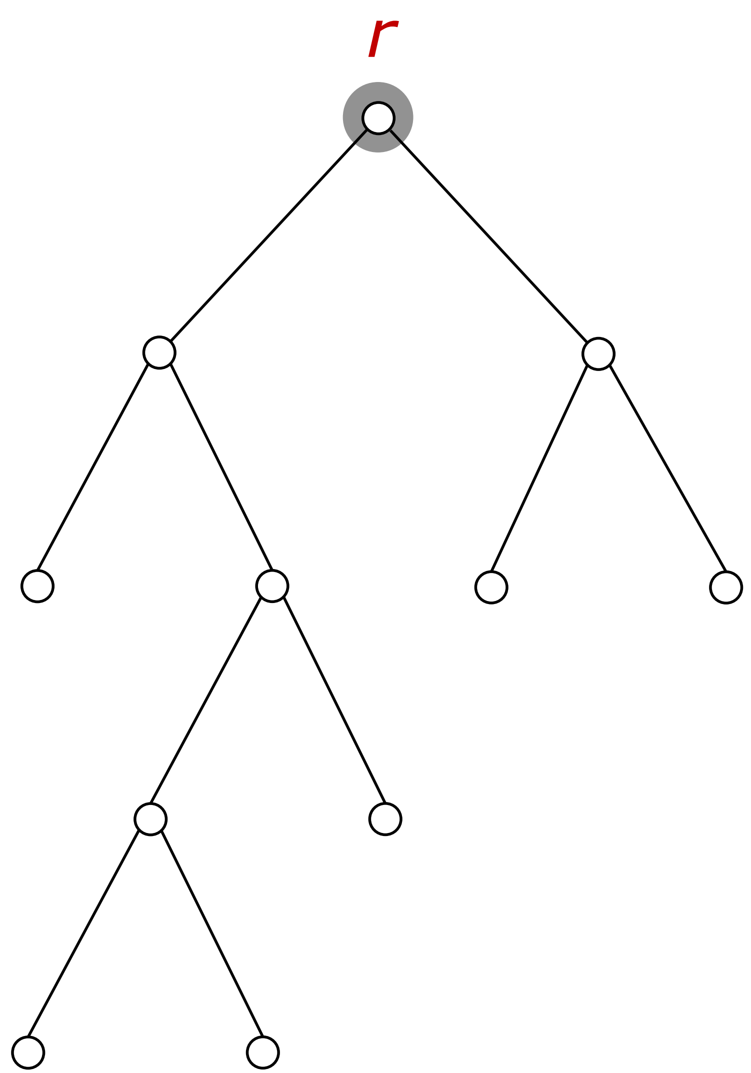
Ideia em árvores



Busca em largura

Ideia em árvores

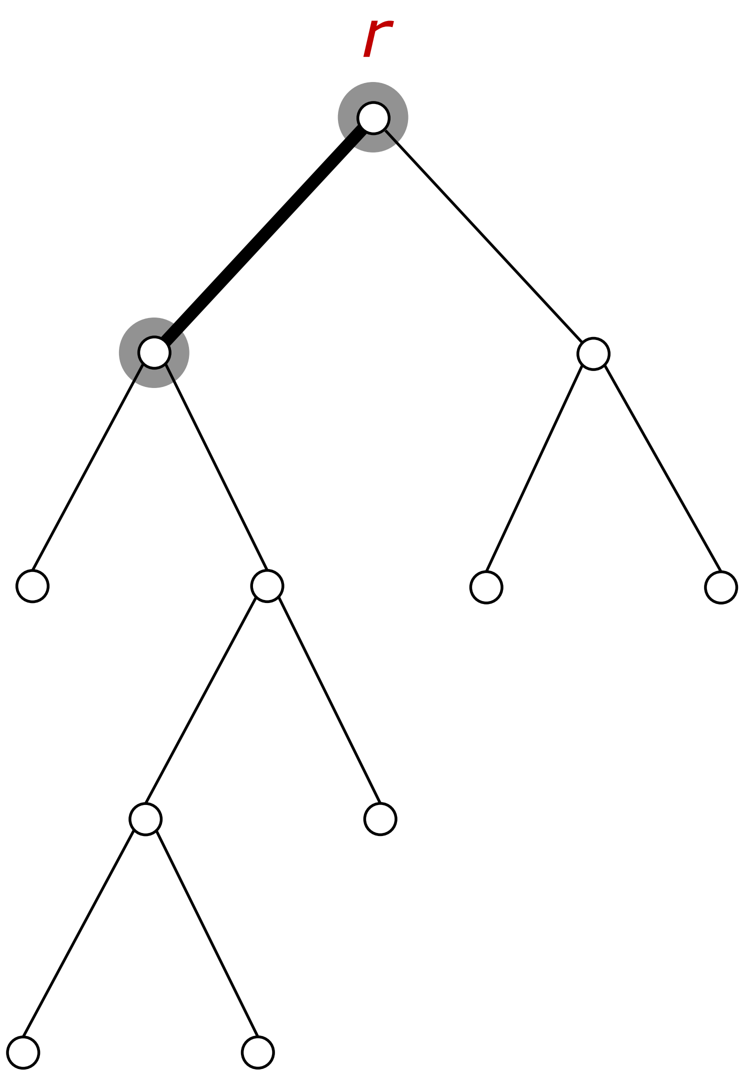
1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ de raiz *r* (largura $k = 1$).



Busca em largura

Ideia em árvores

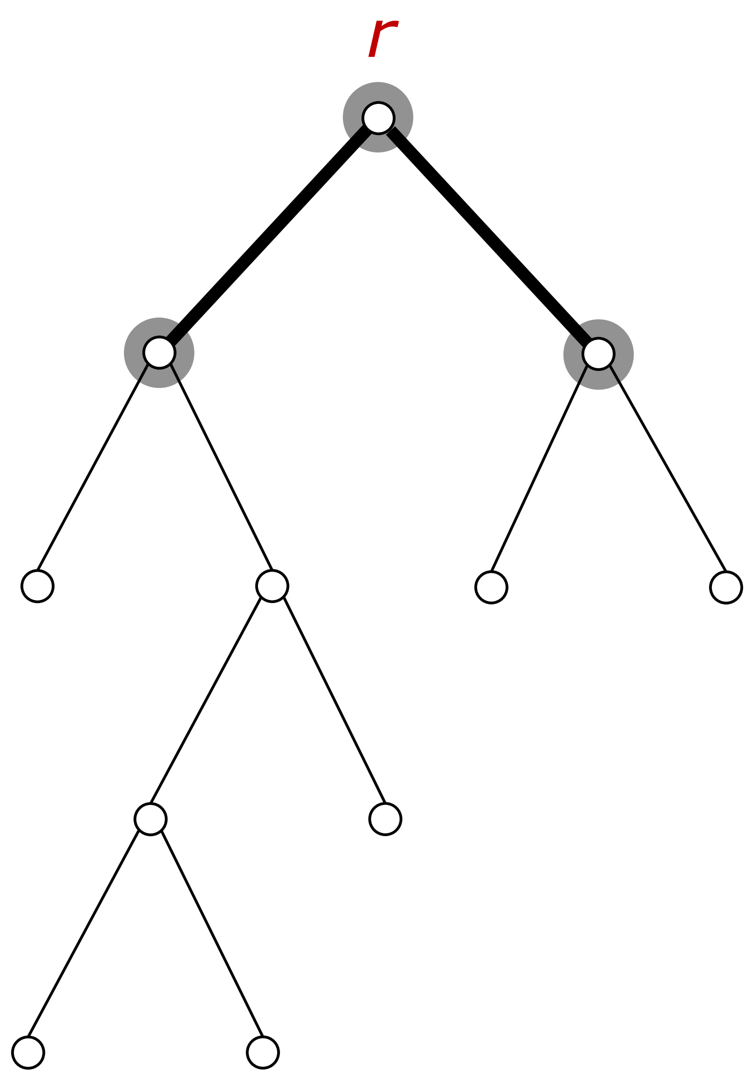
1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).



Busca em largura

Ideia em árvores

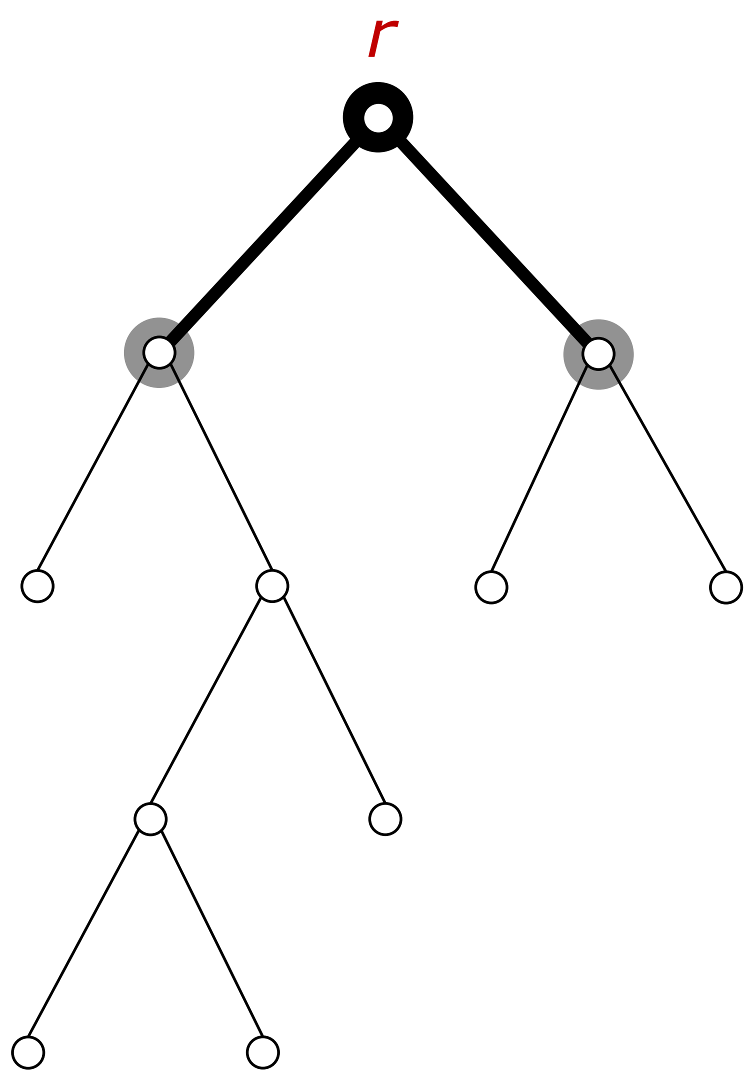
1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).



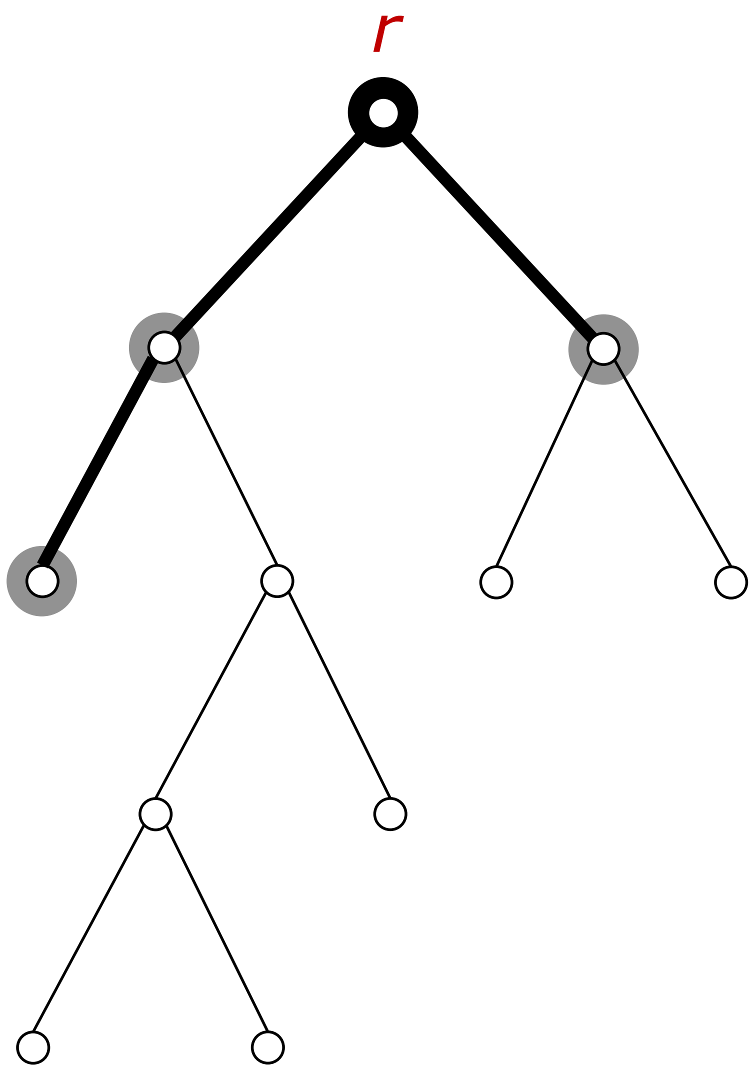
Busca em largura

Ideia em árvores

1. Partindo da raiz r, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz r (largura $k = 1$).



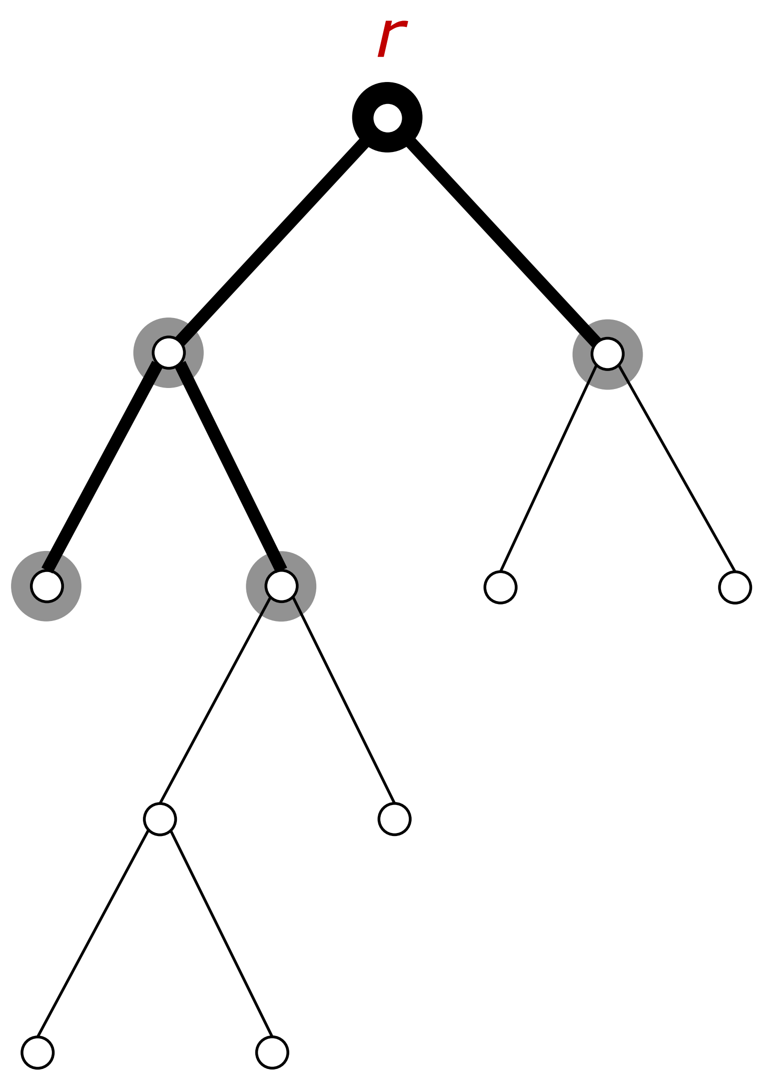
Busca em largura



Ideia em árvores

1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.

Busca em largura



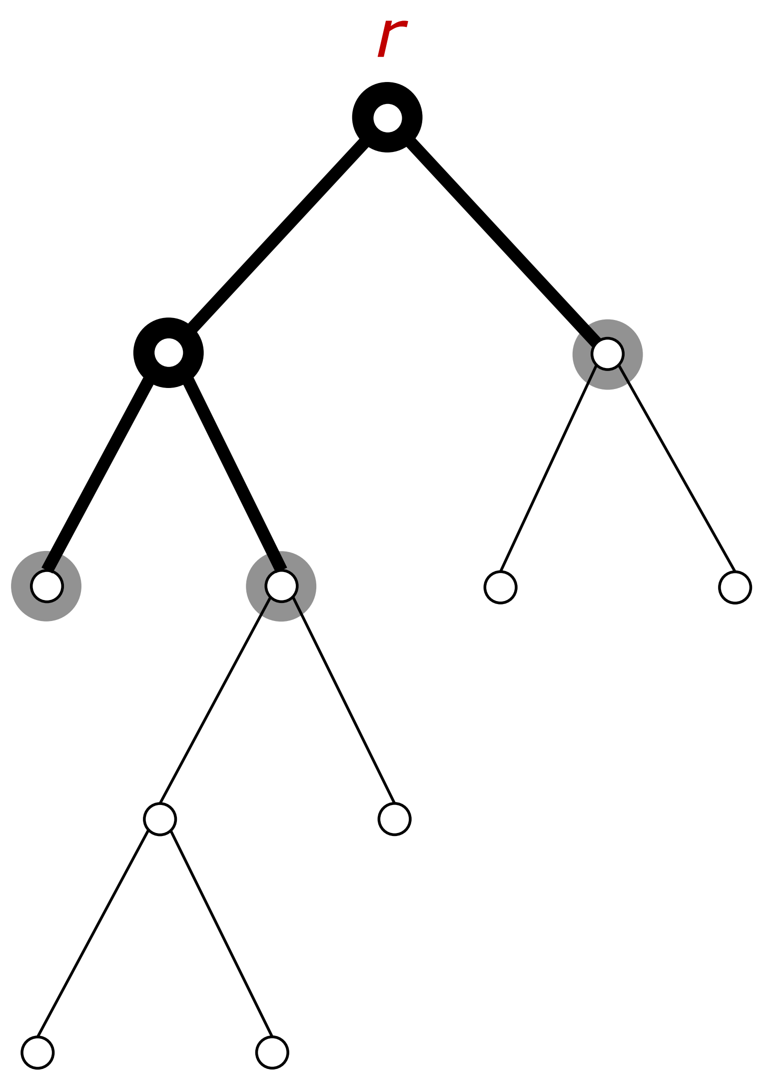
Ideia em árvores

1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.

Busca em largura

Ideia em árvores

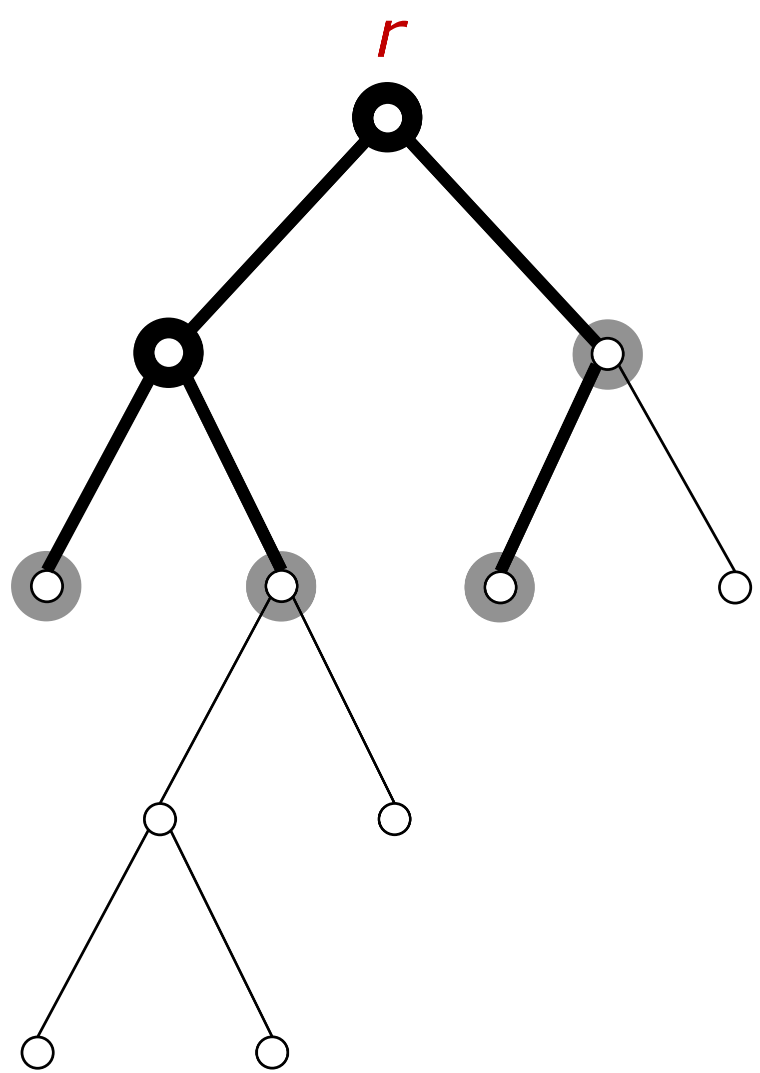
1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.



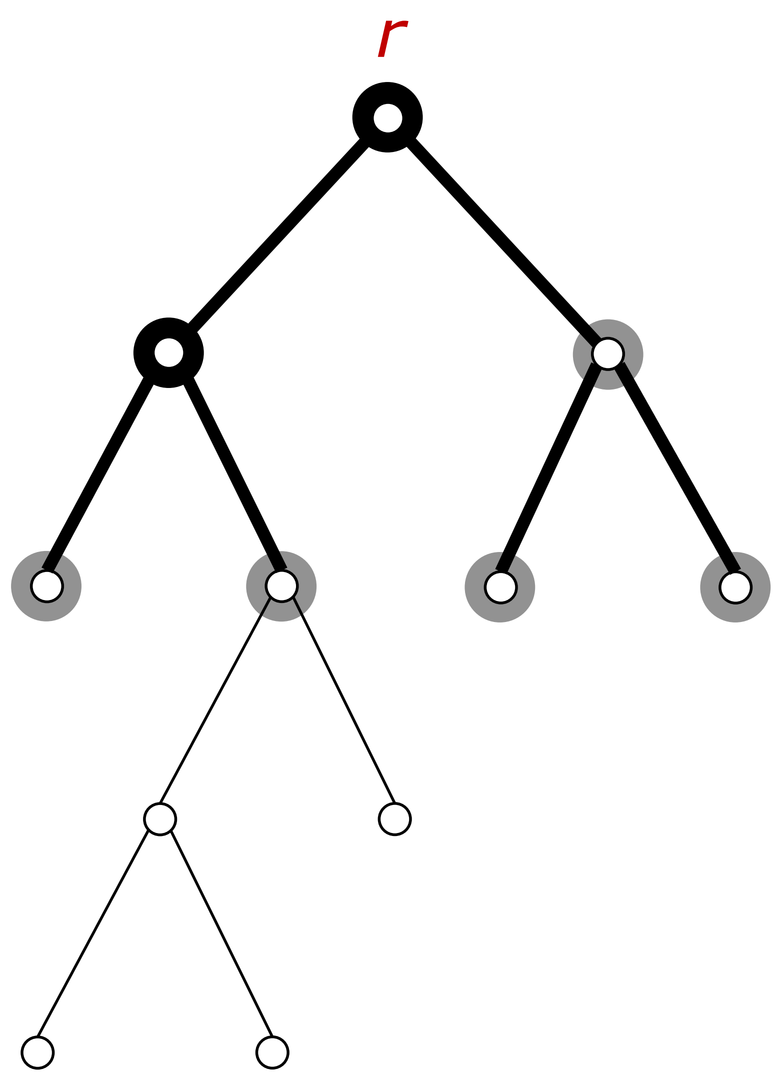
Busca em largura

Ideia em árvores

1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.



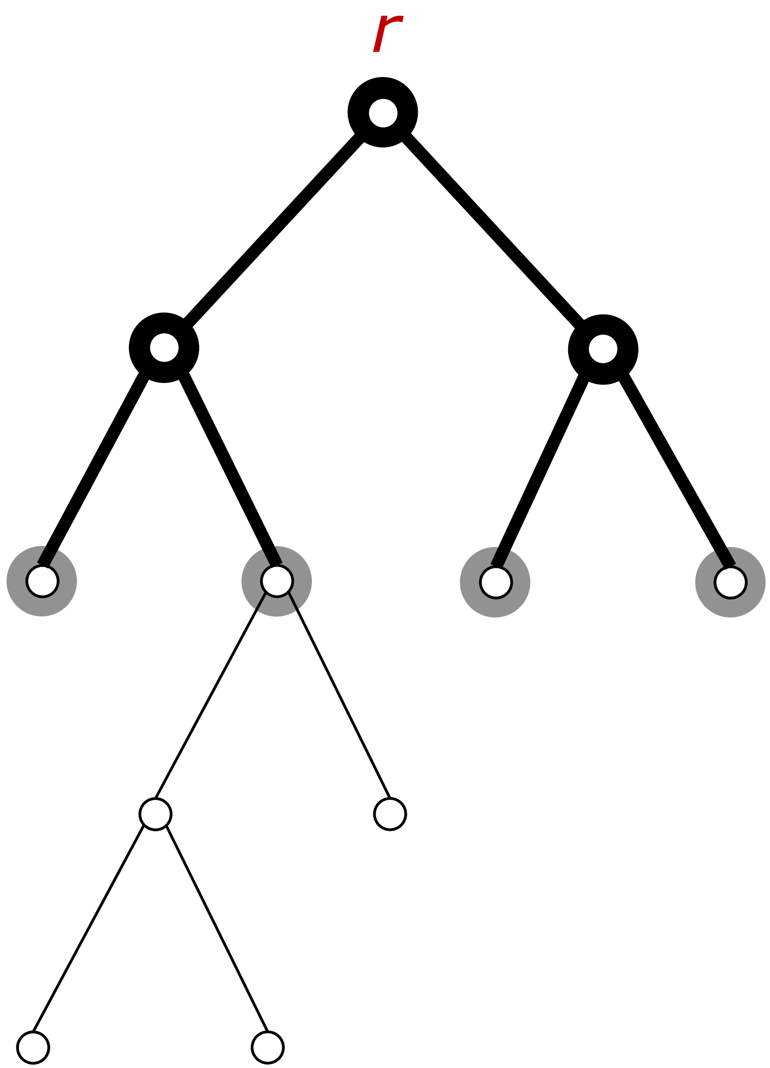
Busca em largura



Ideia em árvores

1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.

Busca em largura



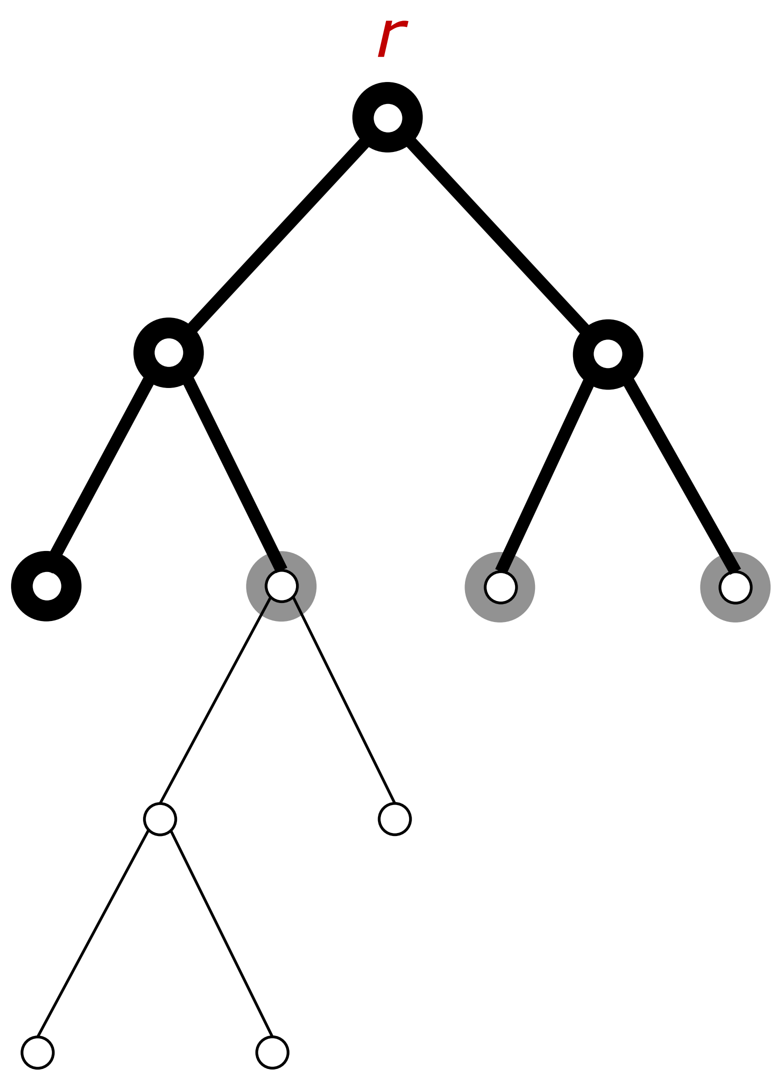
Ideia em árvores

1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.

Busca em largura

Ideia em árvores

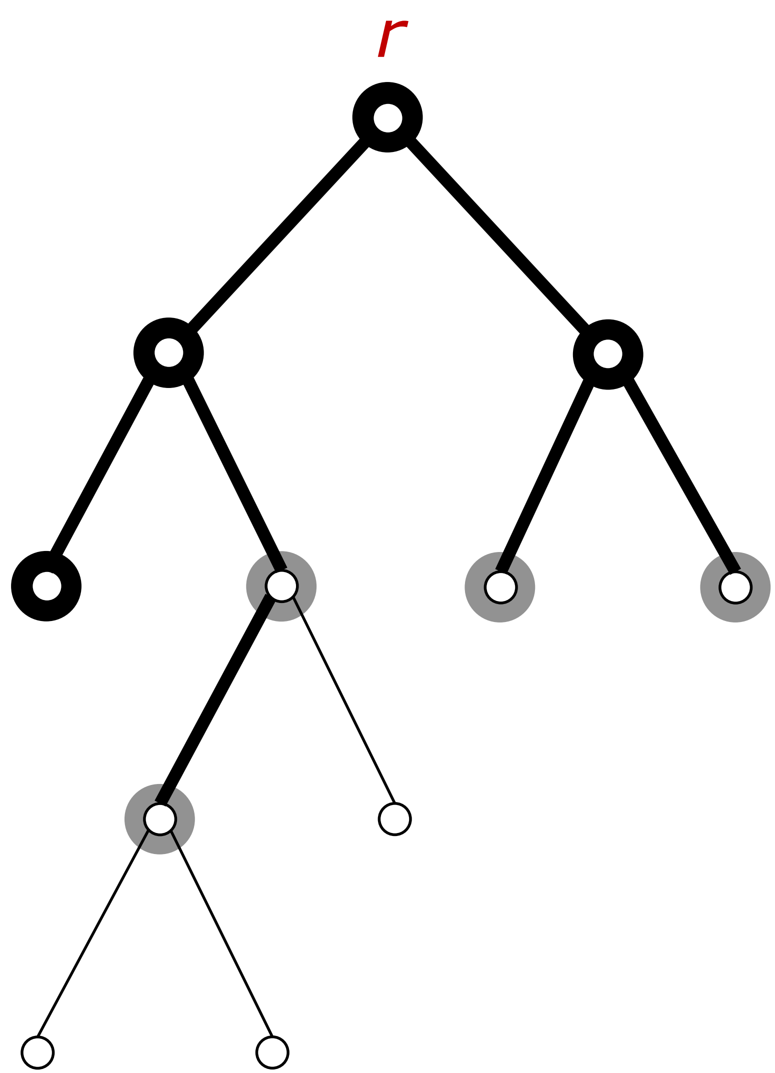
1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.



Busca em largura

Ideia em árvores

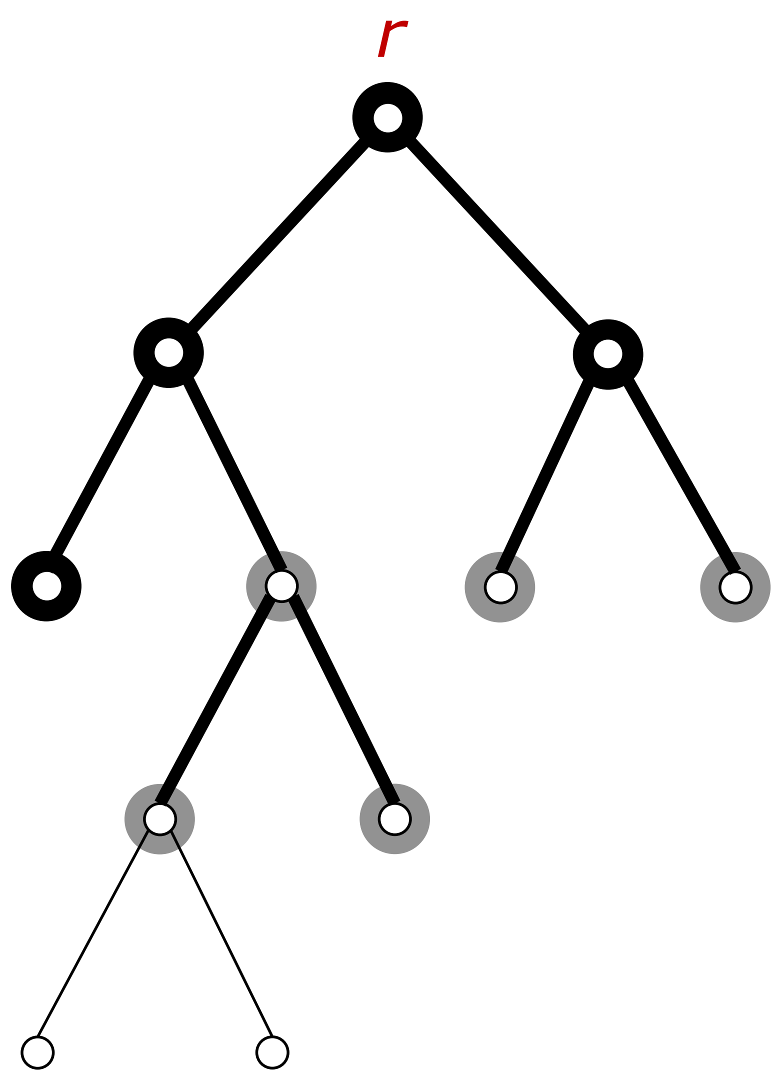
1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.



Busca em largura

Ideia em árvores

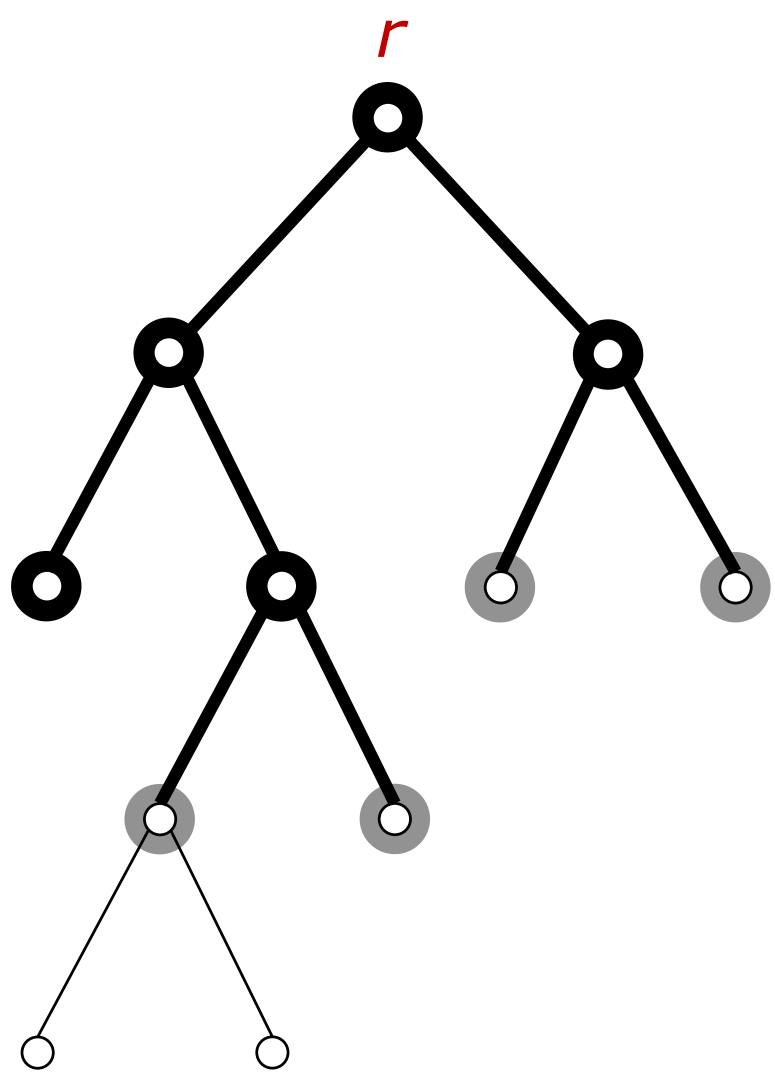
1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.



Busca em largura

Ideia em árvores

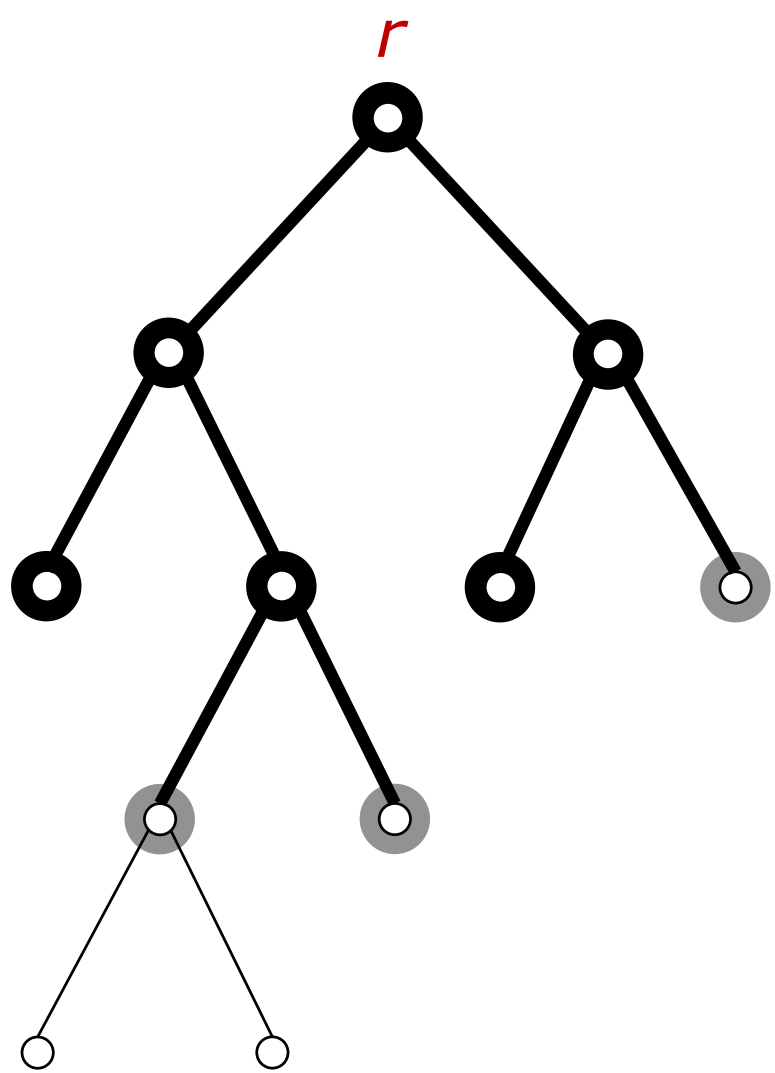
1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.



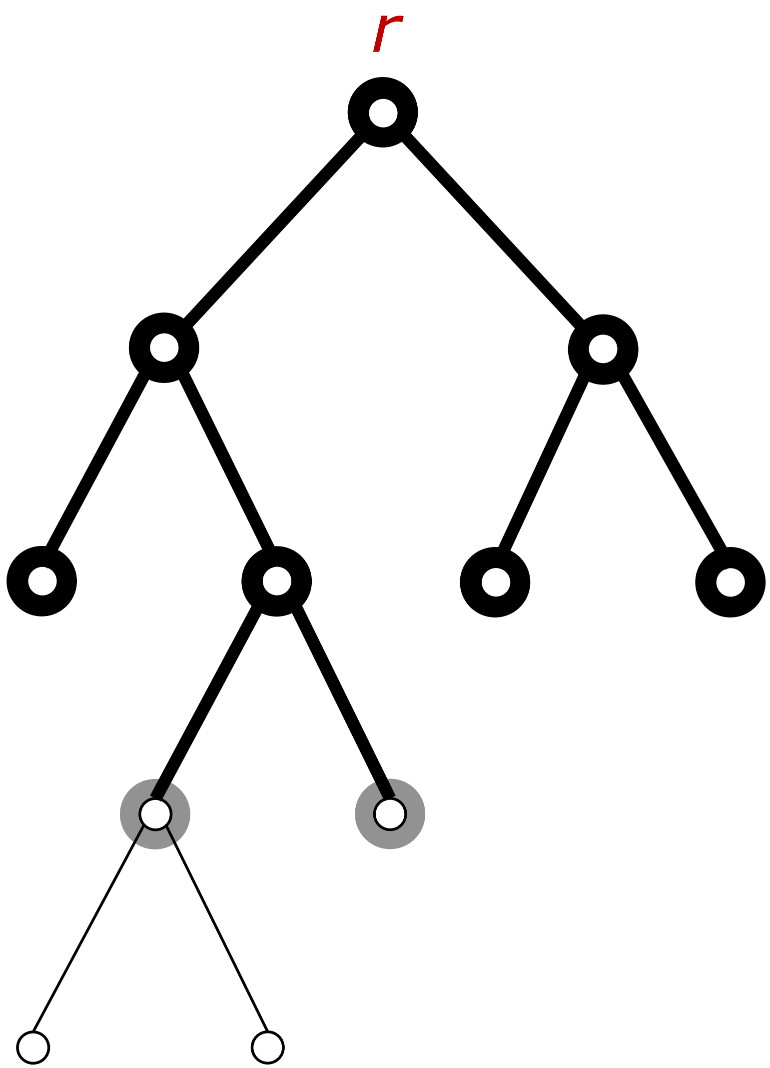
Busca em largura

Ideia em árvores

1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.



Busca em largura



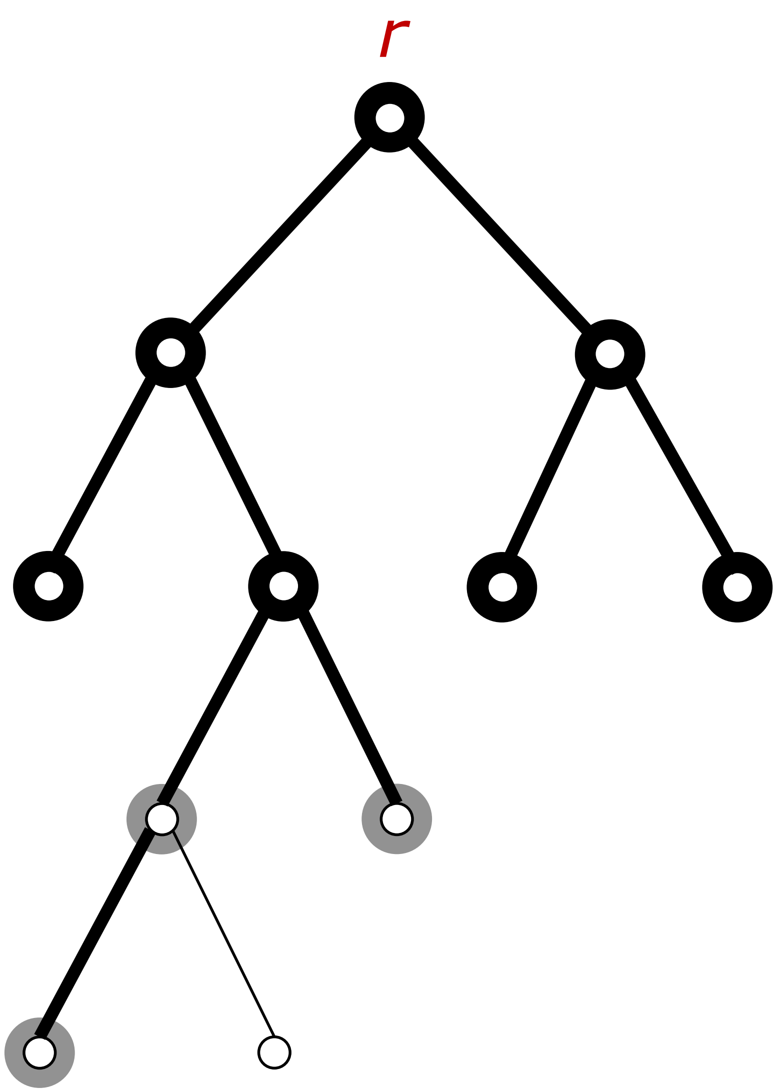
Ideia em árvores

1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.

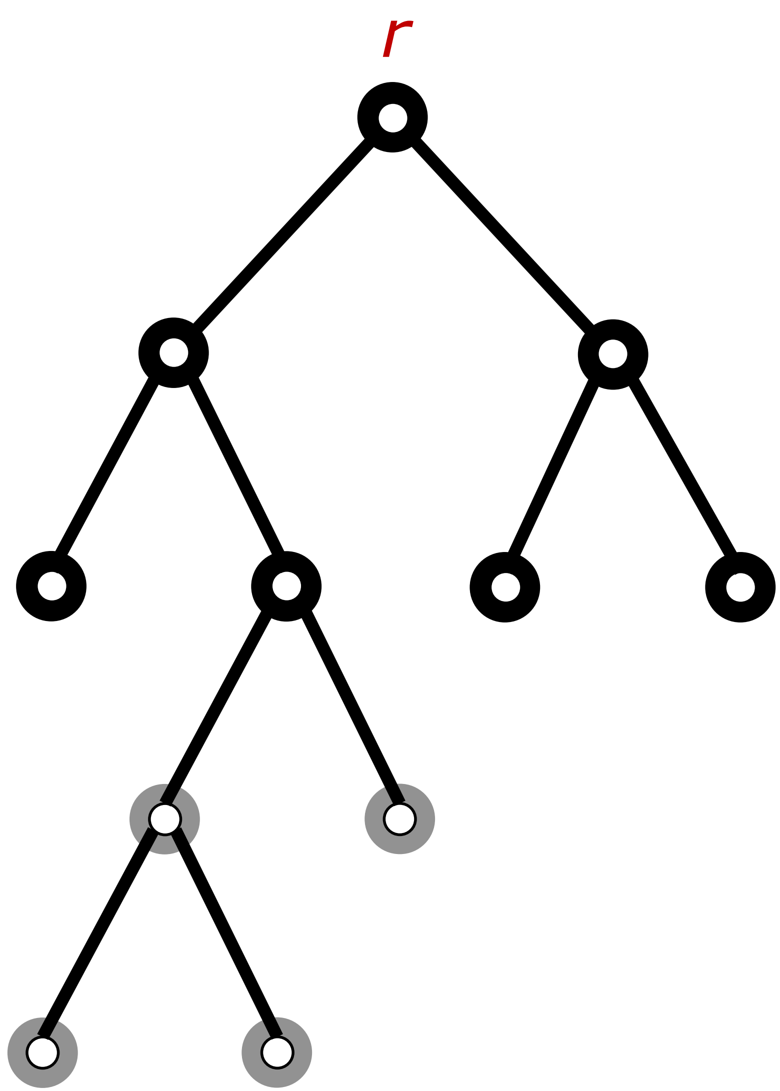
Busca em largura

Ideia em árvores

1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.



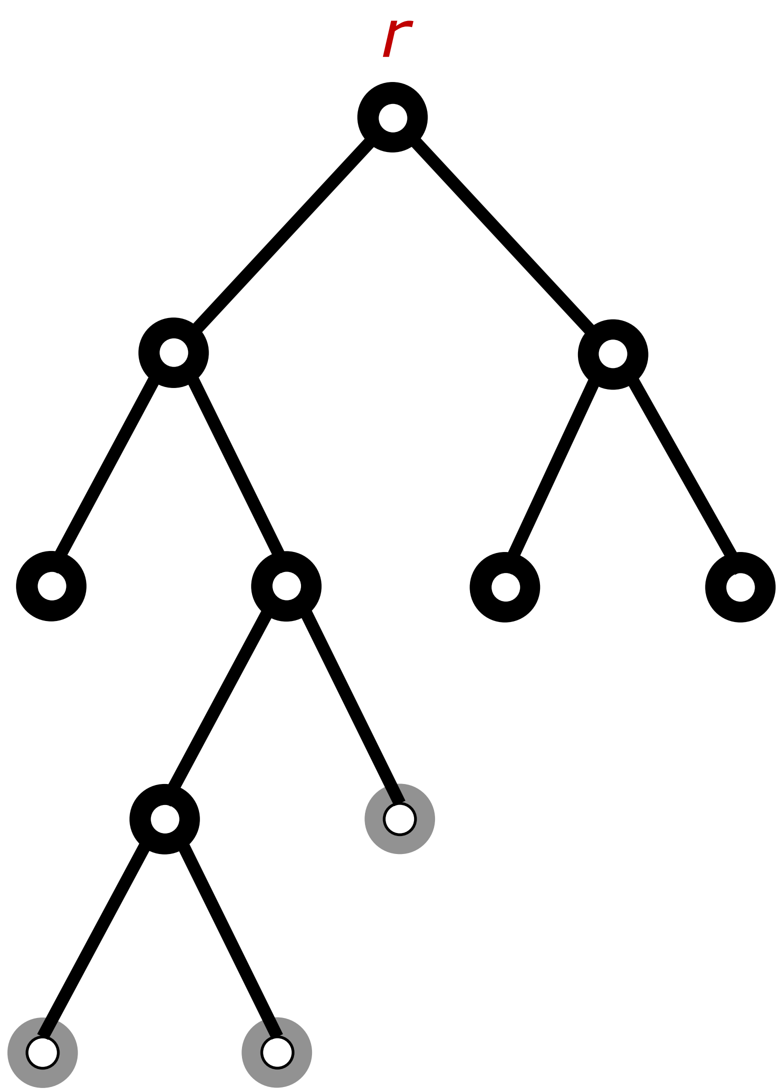
Busca em largura



Ideia em árvores

1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.

Busca em largura



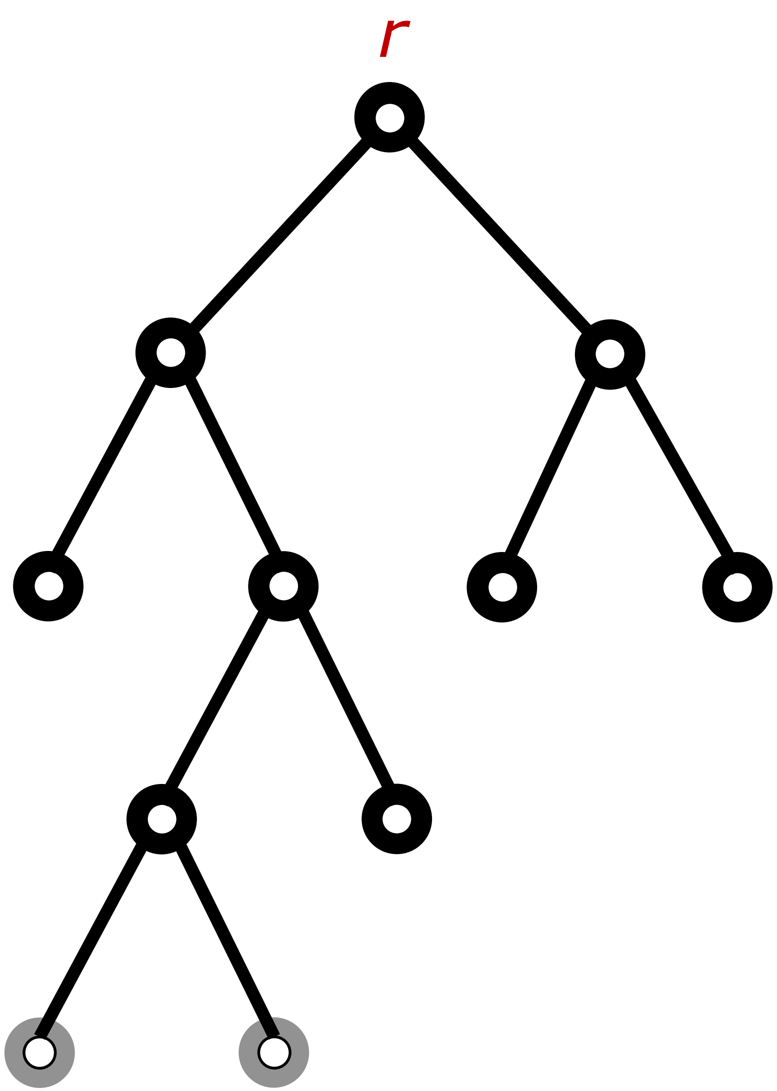
Ideia em árvores

1. Partindo da raiz r, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz r (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.

Busca em largura

Ideia em árvores

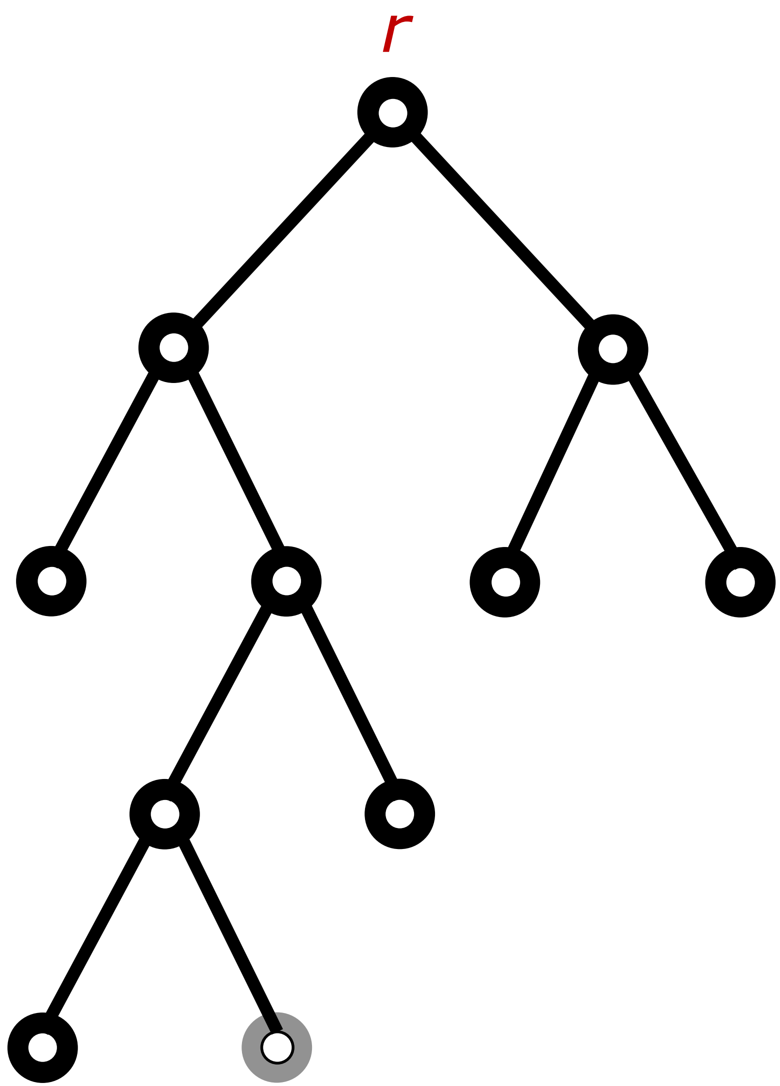
1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.



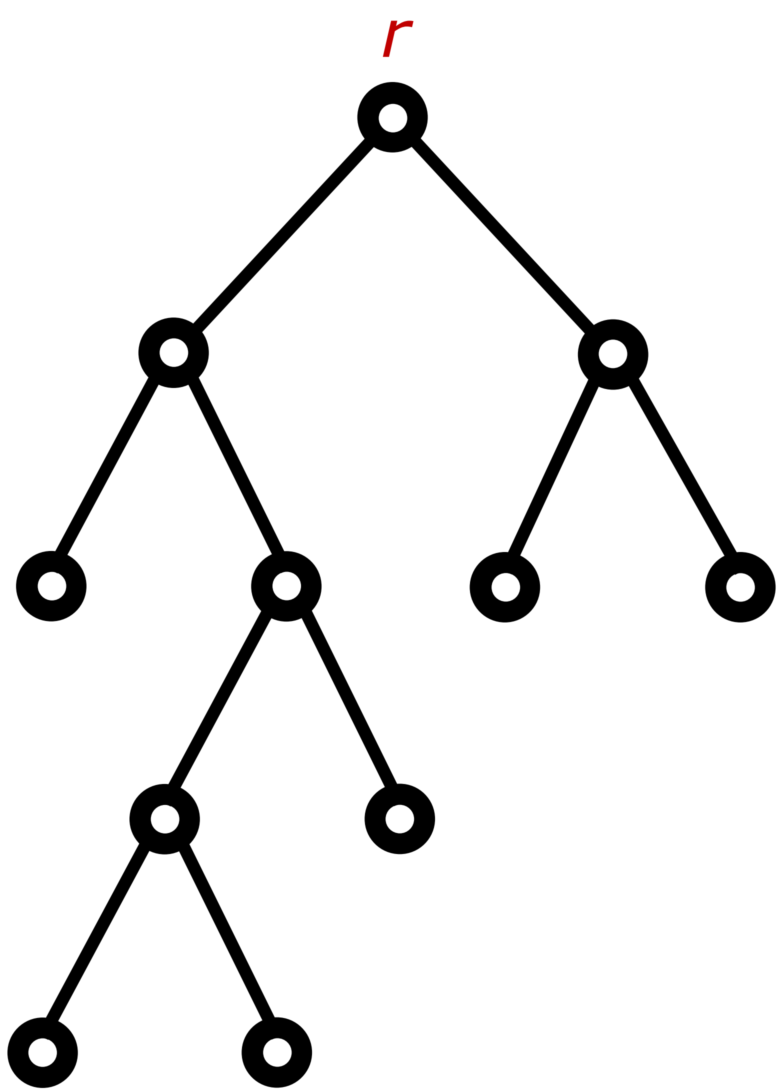
Busca em largura

Ideia em árvores

1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.



Busca em largura



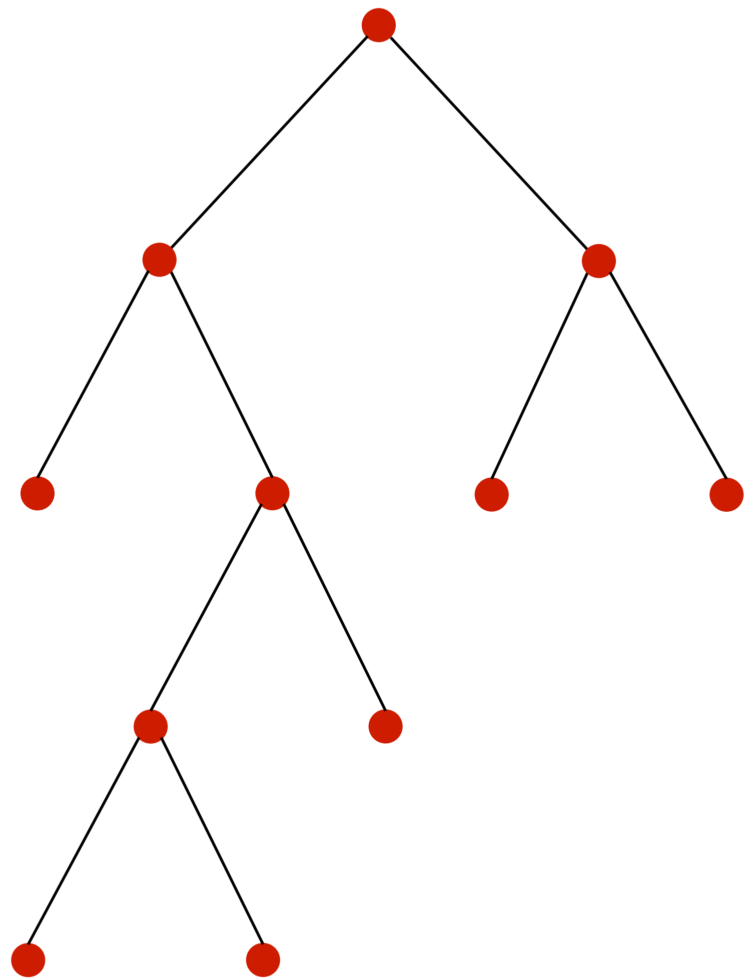
Ideia em árvores

1. Partindo da raiz *r*, o algoritmo visita (descobre) os nós a uma distância $k = 1$ da raiz *r* (largura $k = 1$).
2. Depois, visita os nós a uma distância $k + 1$, e assim por diante, até o nó mais profundo.

Busca em largura

Ideia em árvores

O algoritmo utiliza uma **fila** para memorizar os vértices cujos filhos devem ser visitados posteriormente.

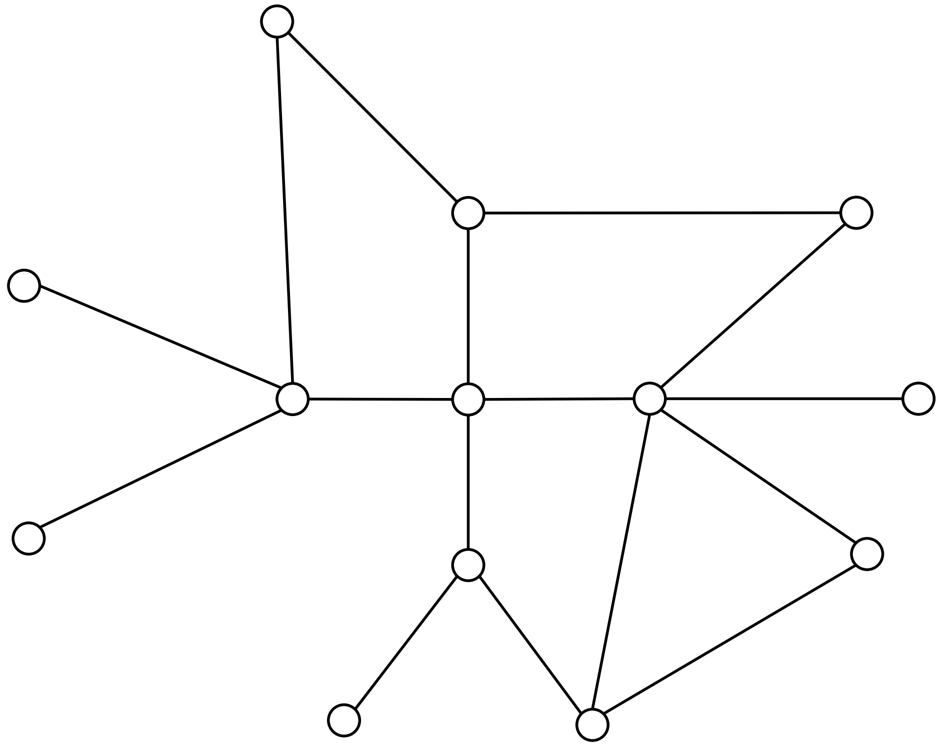


Busca em largura

Ideia em grafos

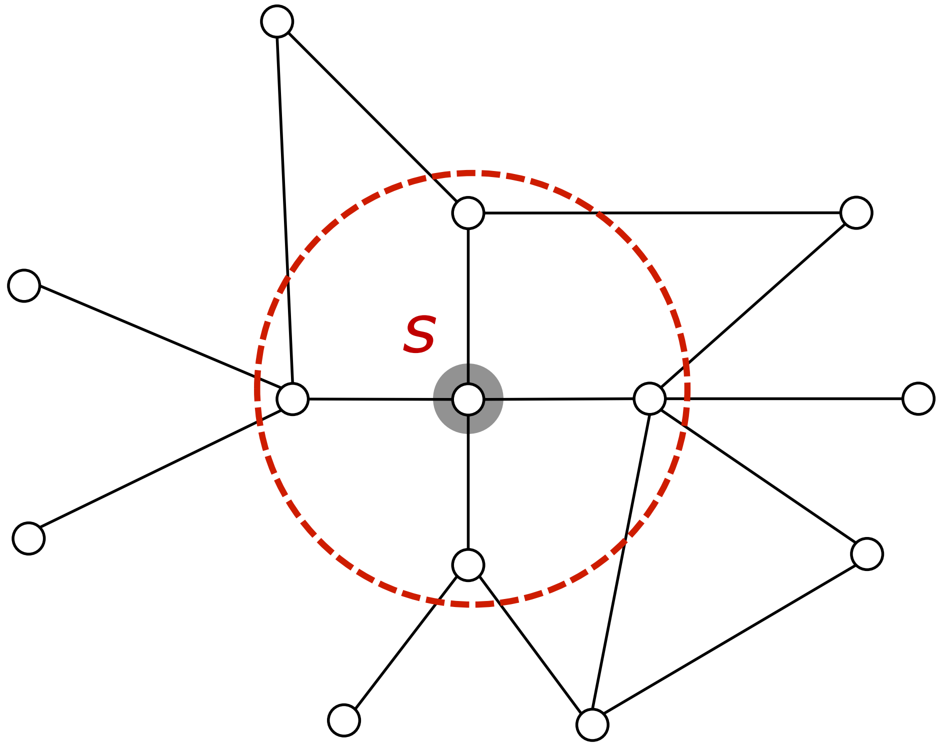
1. Partindo de um vértice-origem ***s*** do grafo, o algoritmo visita (descobre) todos os vértices a uma distância $k = 1$ do vértice s.
2. Depois, visita os nós a uma distância $k + 1$ de s, e assim por diante, até os vértices mais distantes de s.

Busca em largura



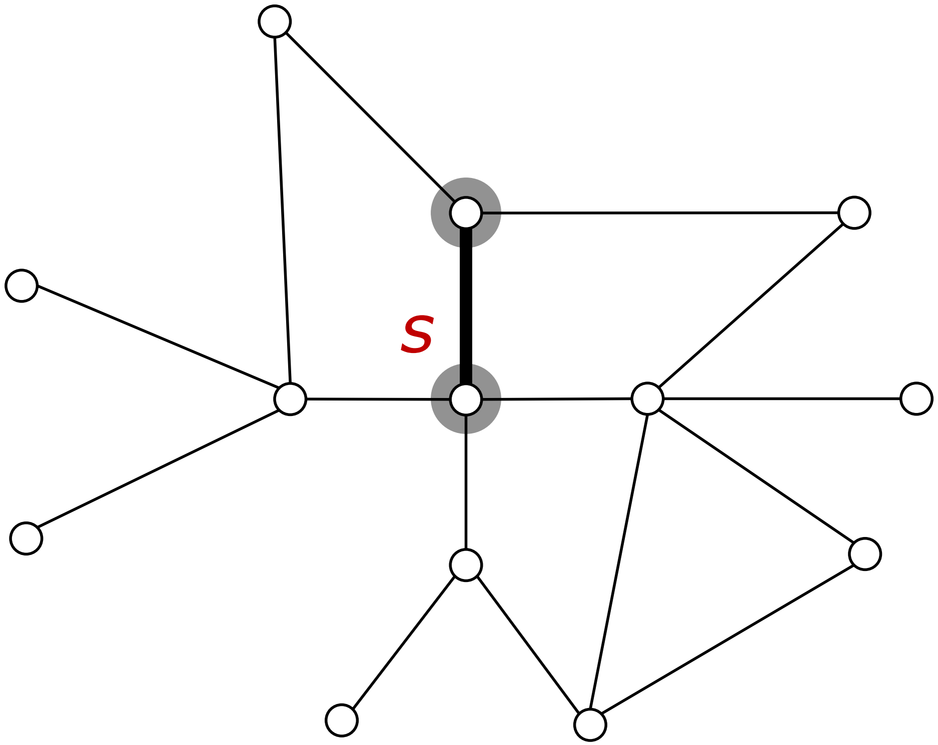
Busca em largura

Escolha um vértice *s* de origem.



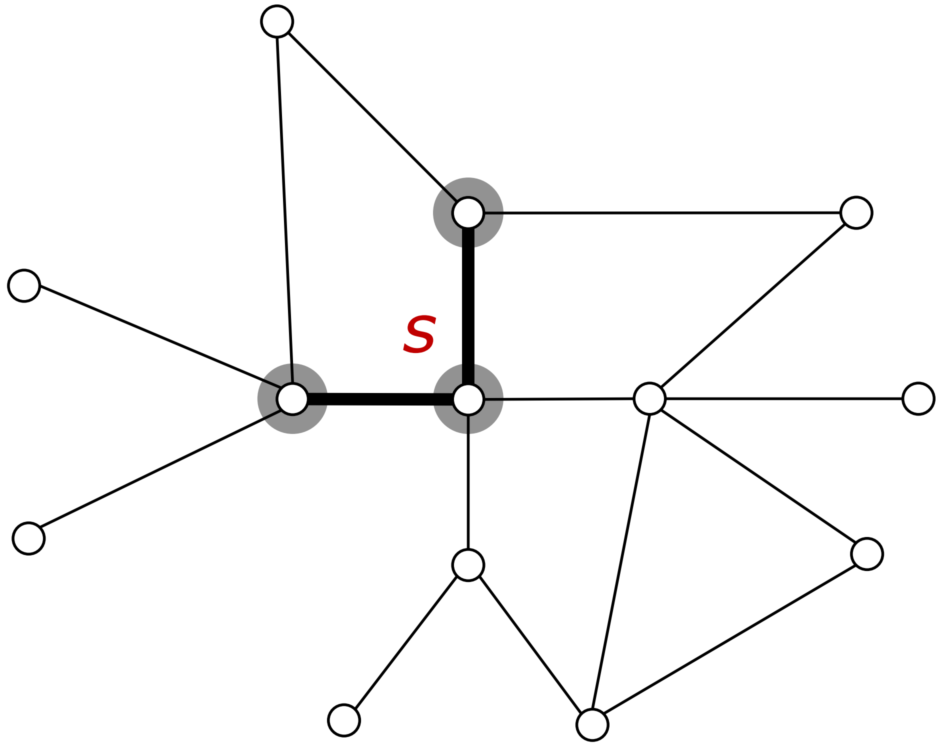
Busca em largura

Visite todos os seus adjacentes.



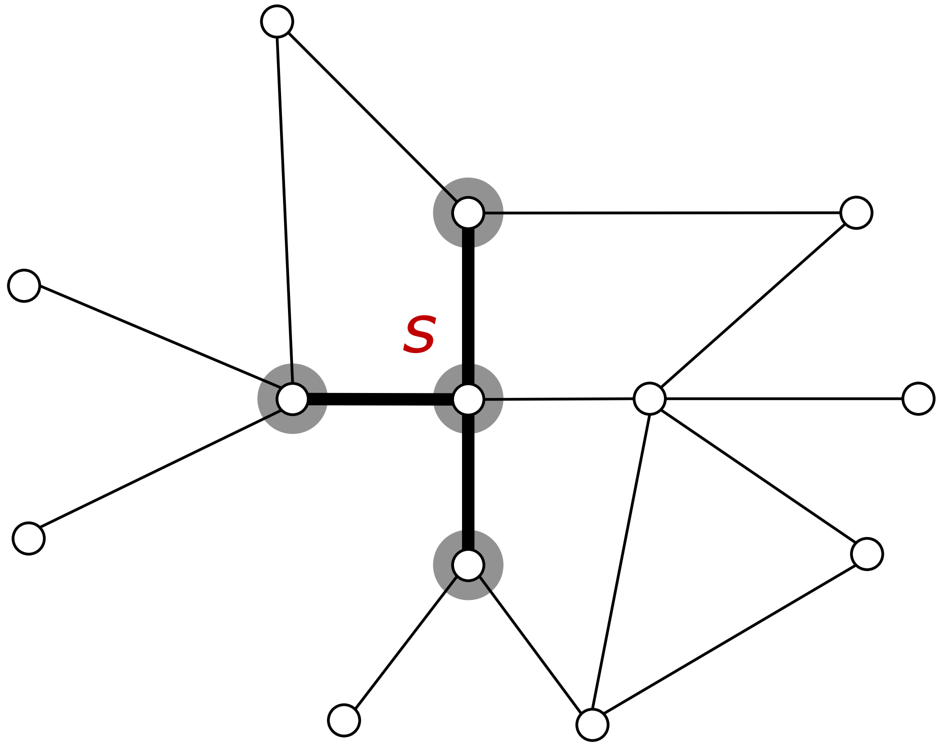
Busca em largura

Visite todos os seus adjacentes.



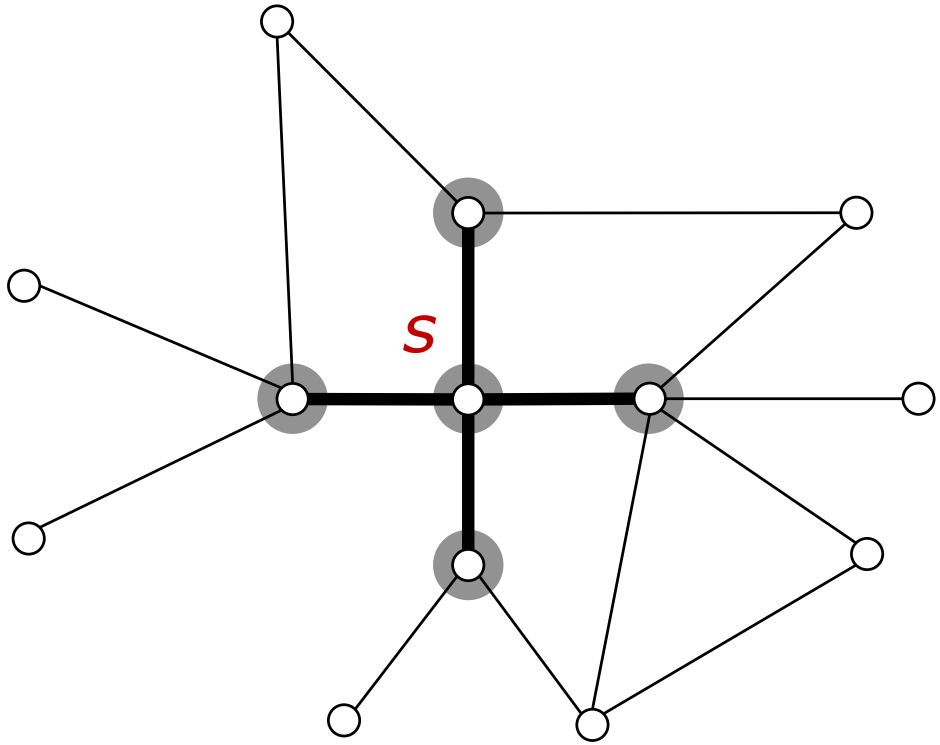
Busca em largura

Visite todos os seus adjacentes.



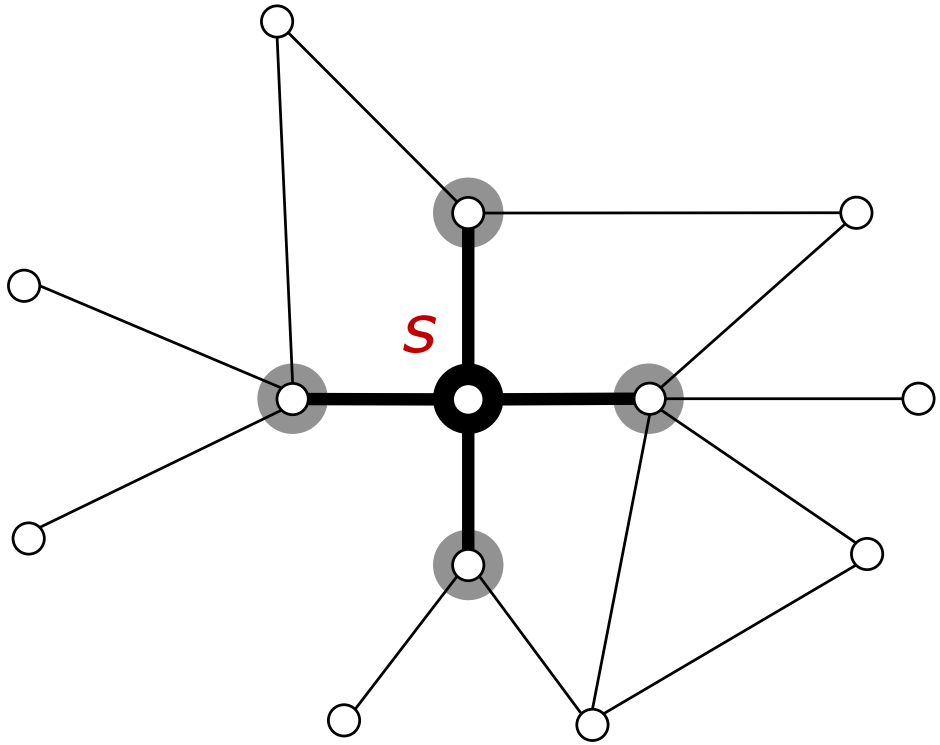
Busca em largura

Visite todos os seus adjacentes.

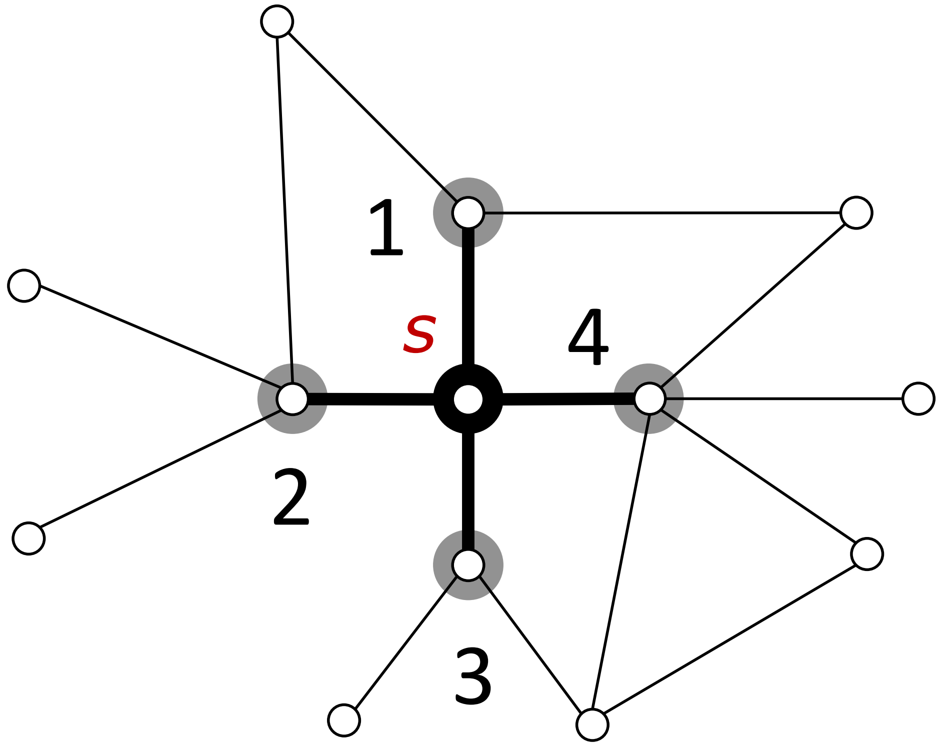


Busca em largura

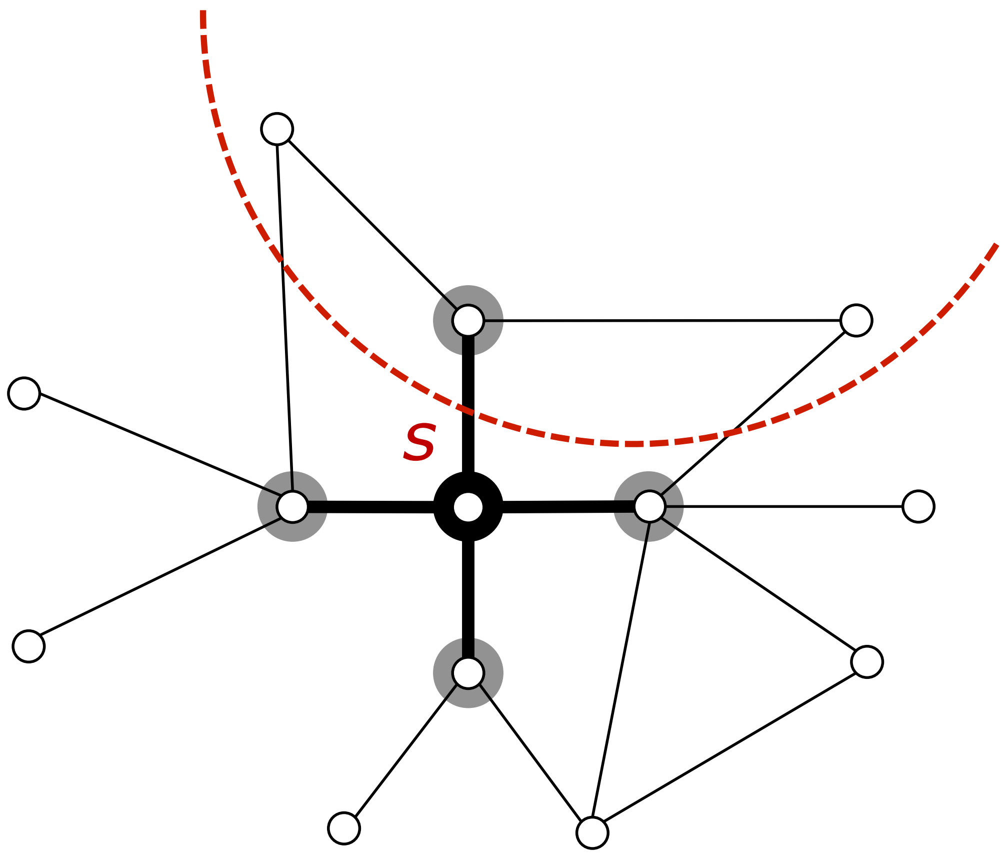
Pinte *s* de **preto** após visitar seus adjacentes.



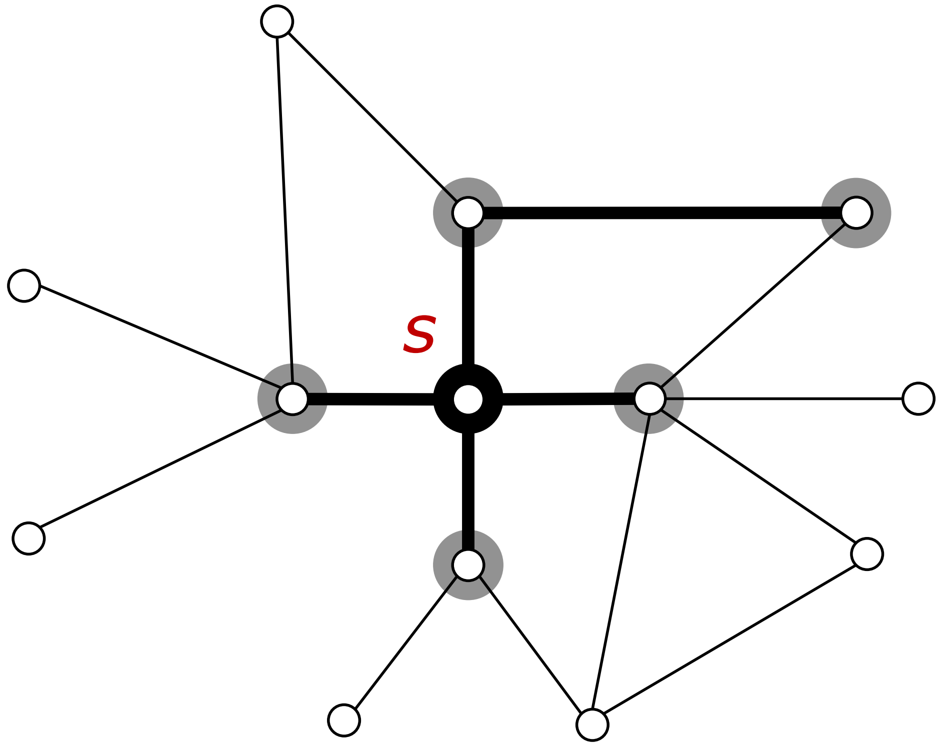
Busca em largura



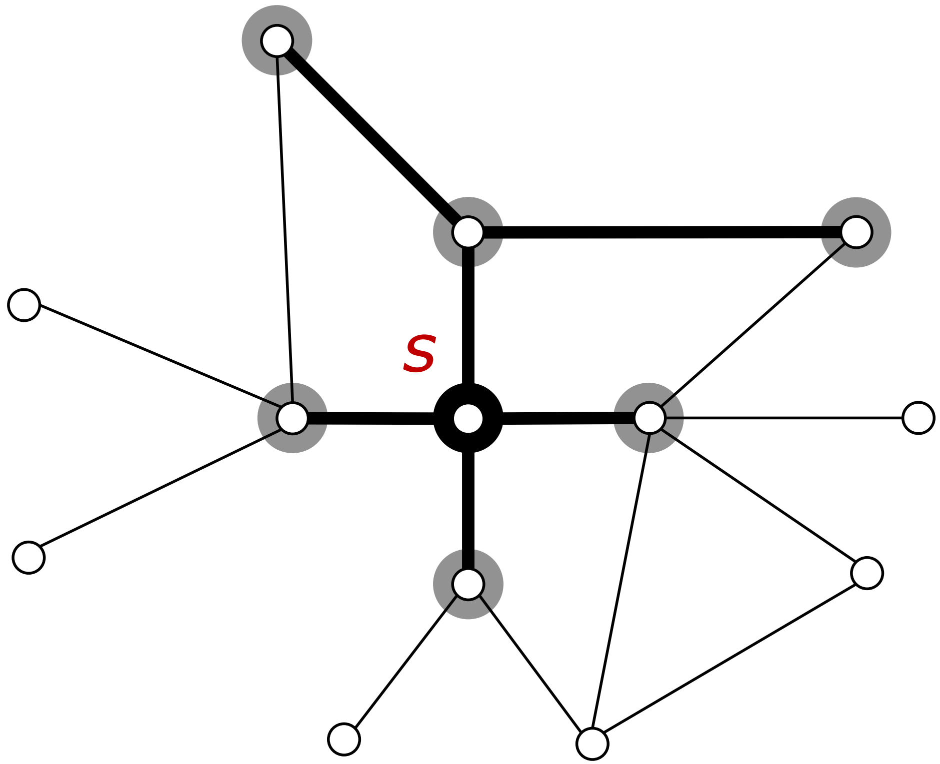
Busca em largura



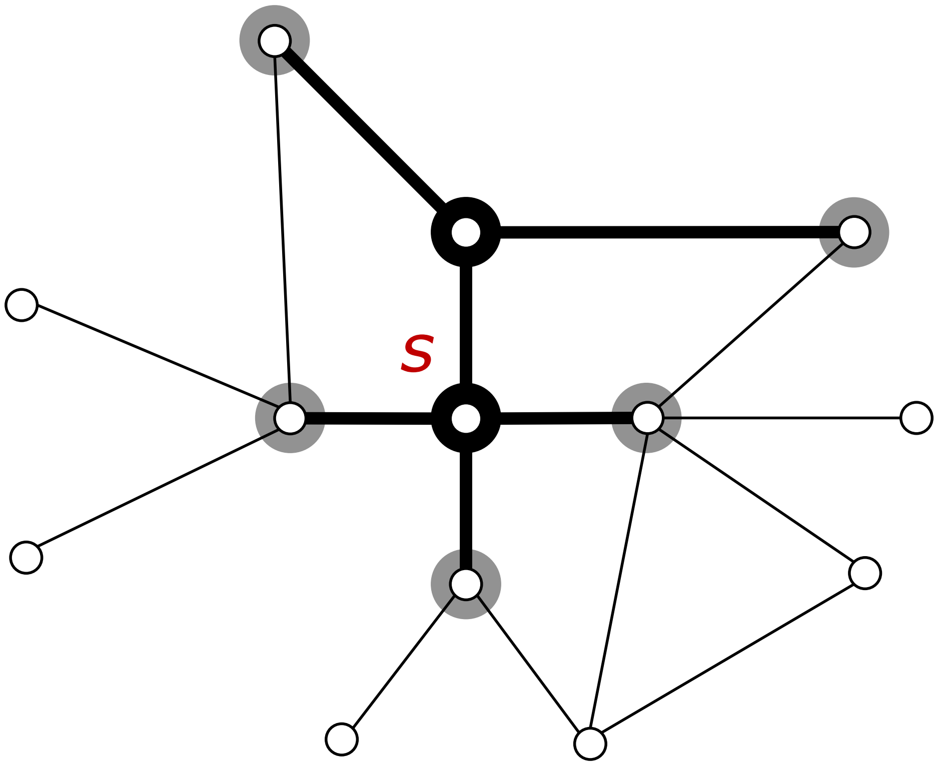
Busca em largura



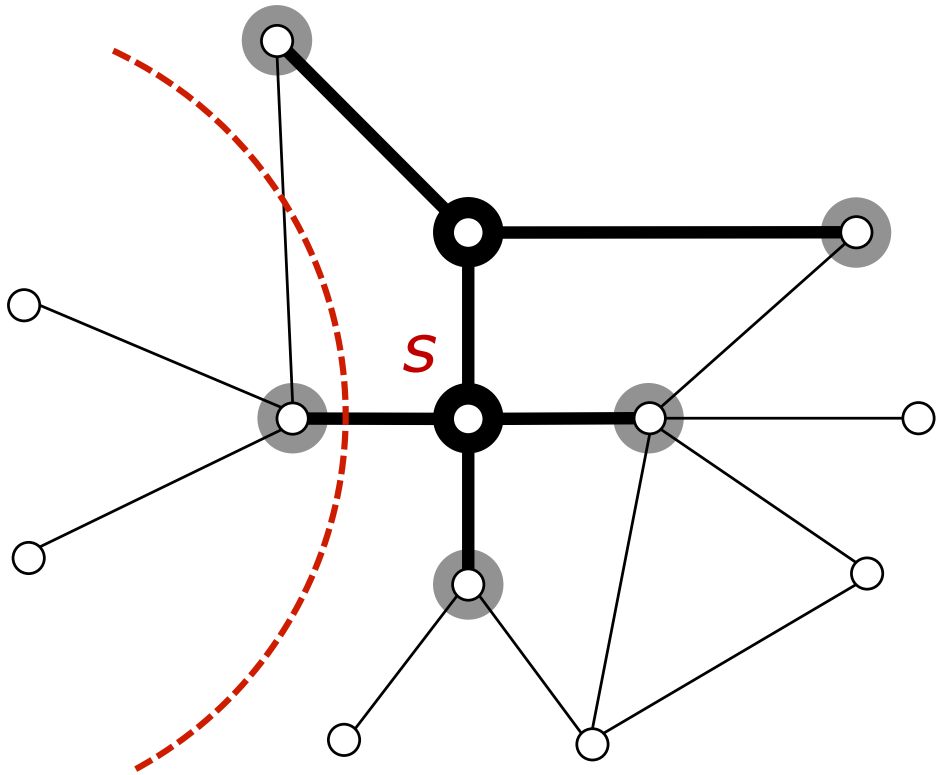
Busca em largura



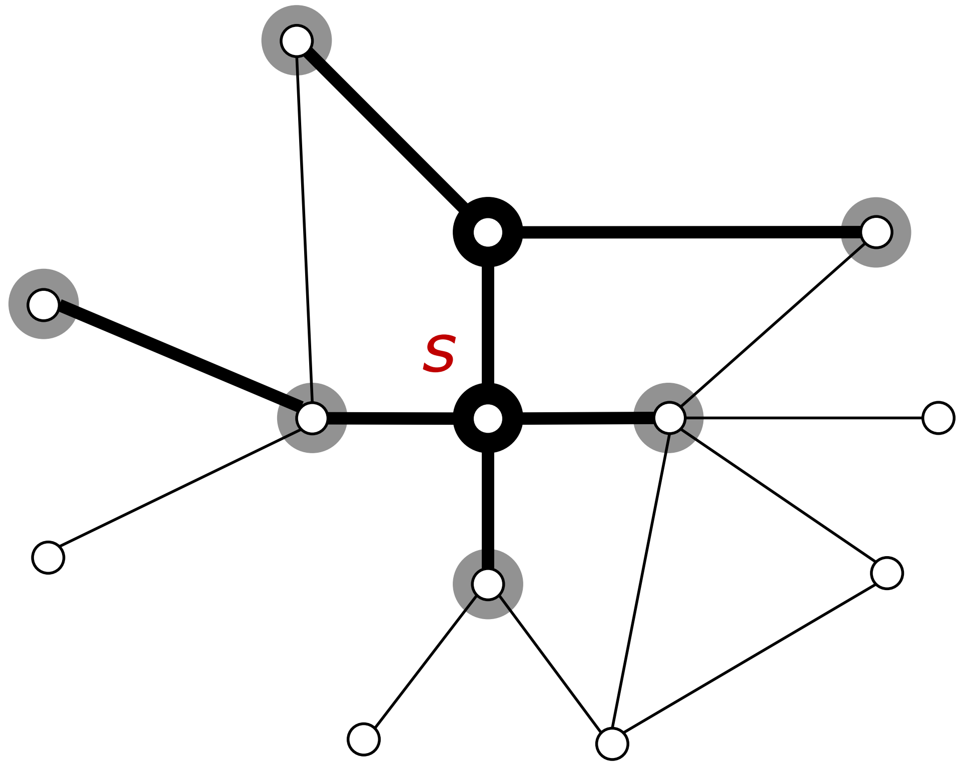
Busca em largura



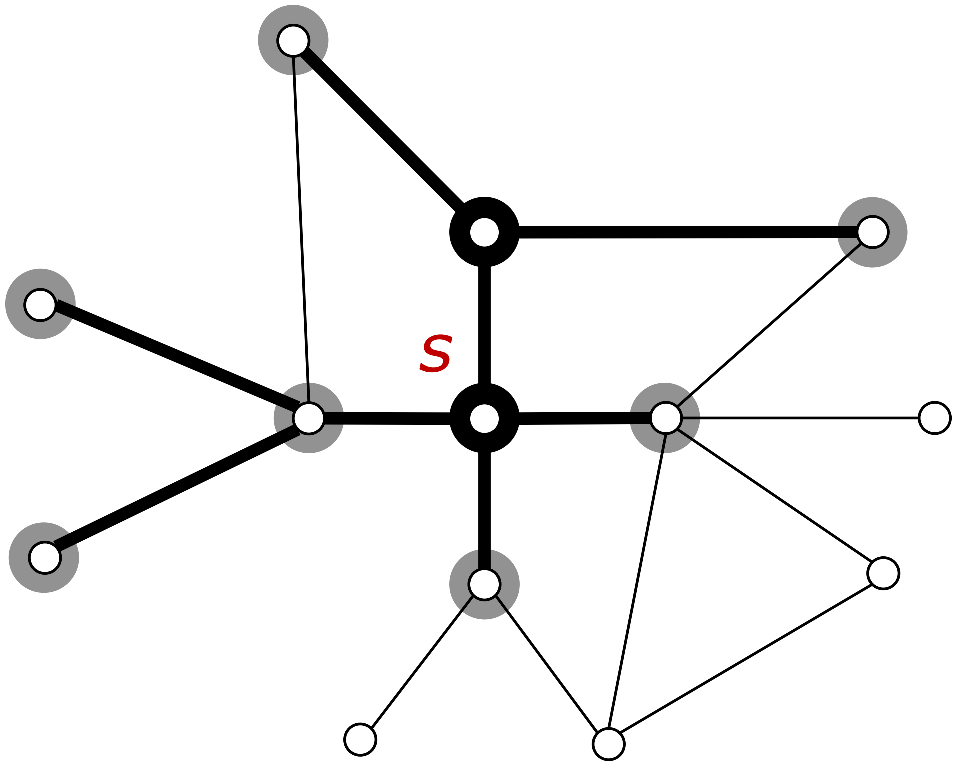
Busca em largura



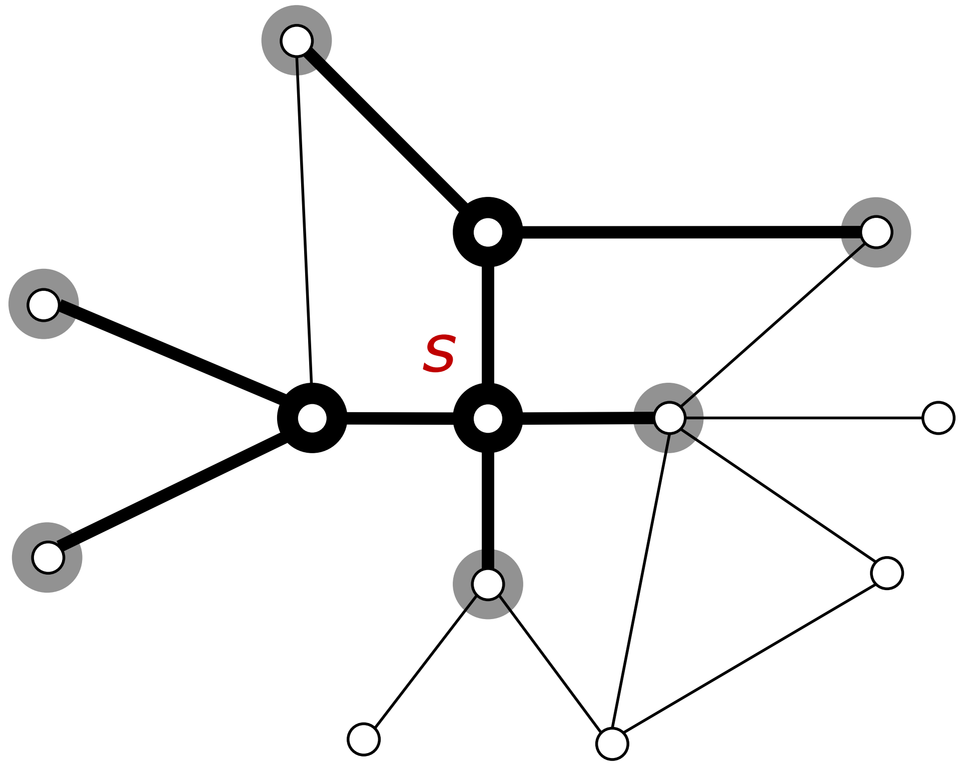
Busca em largura



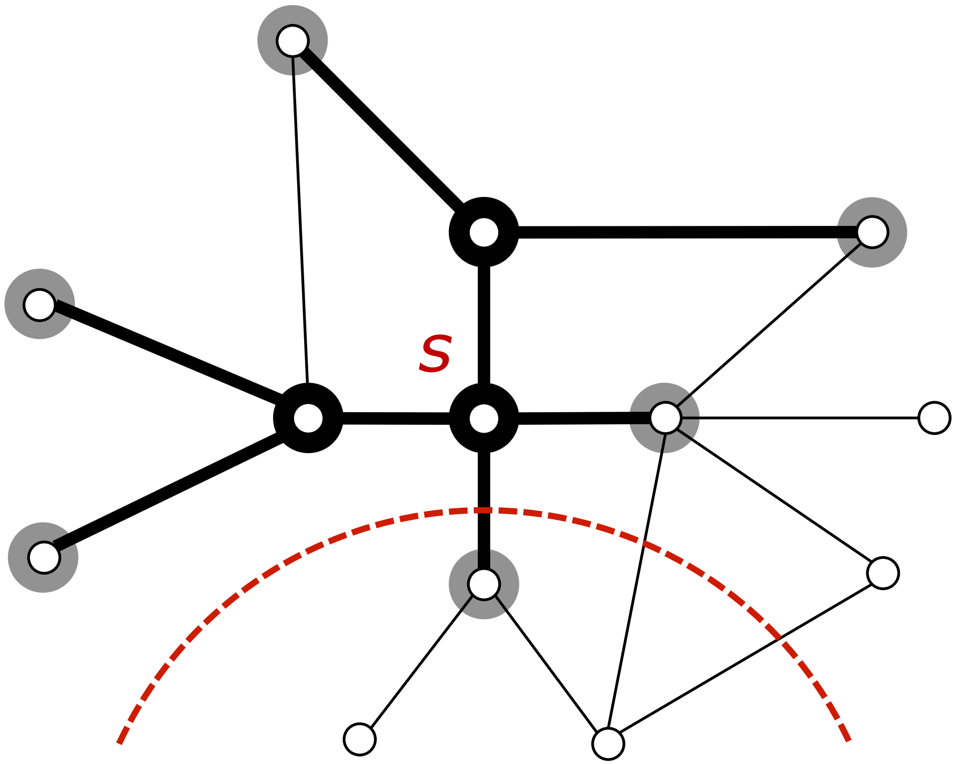
Busca em largura



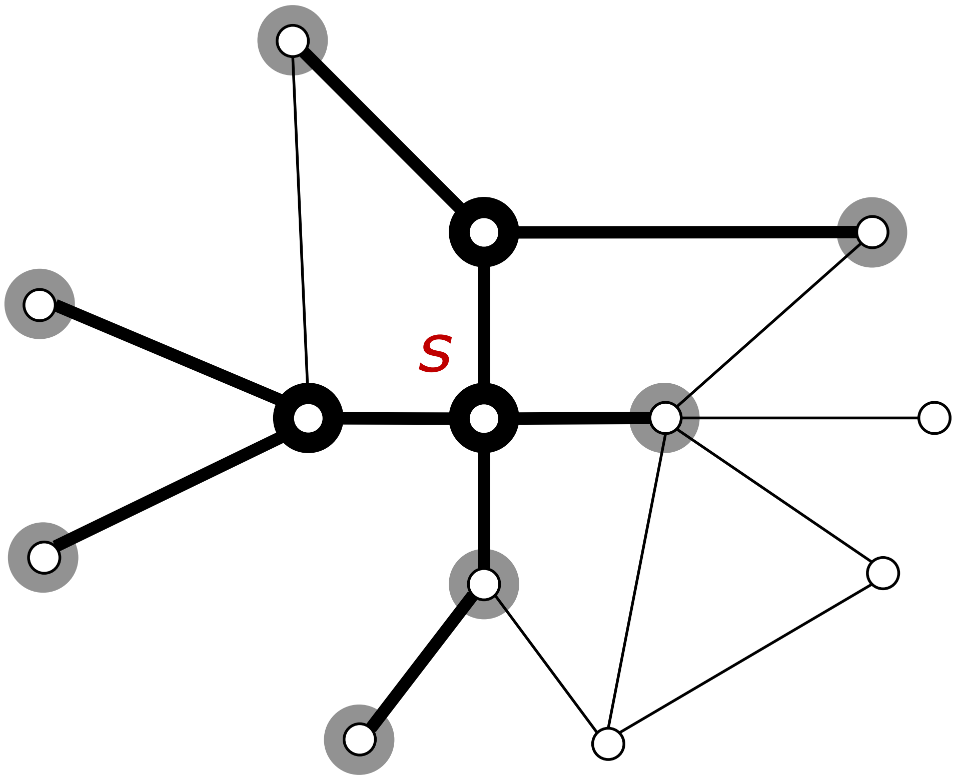
Busca em largura



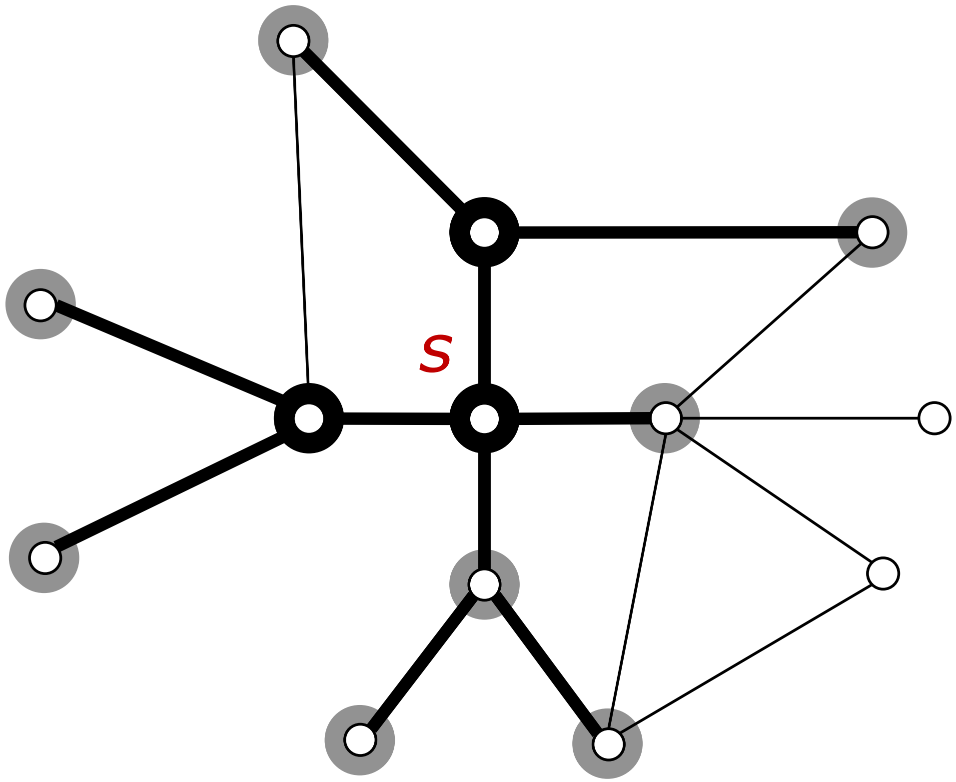
Busca em largura



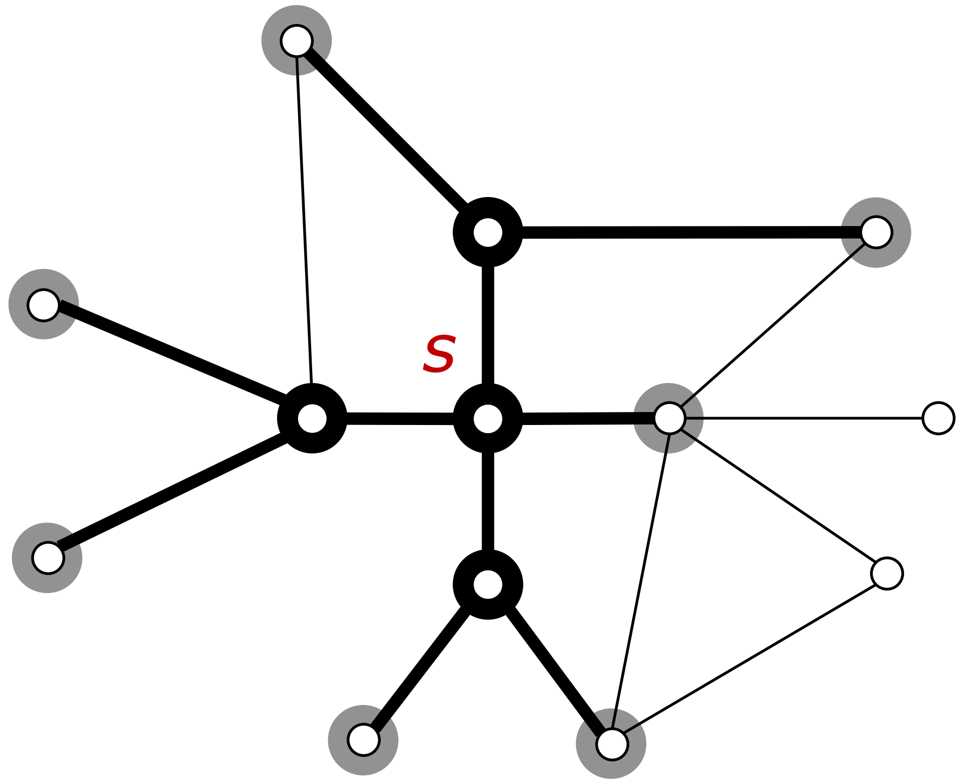
Busca em largura



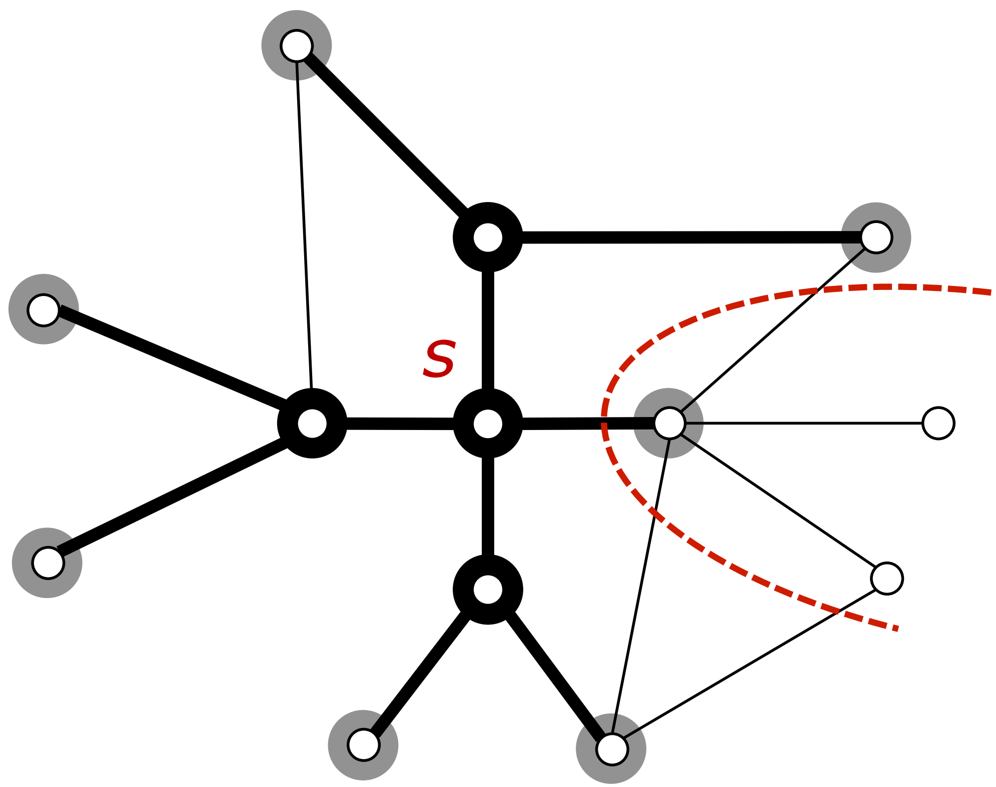
Busca em largura



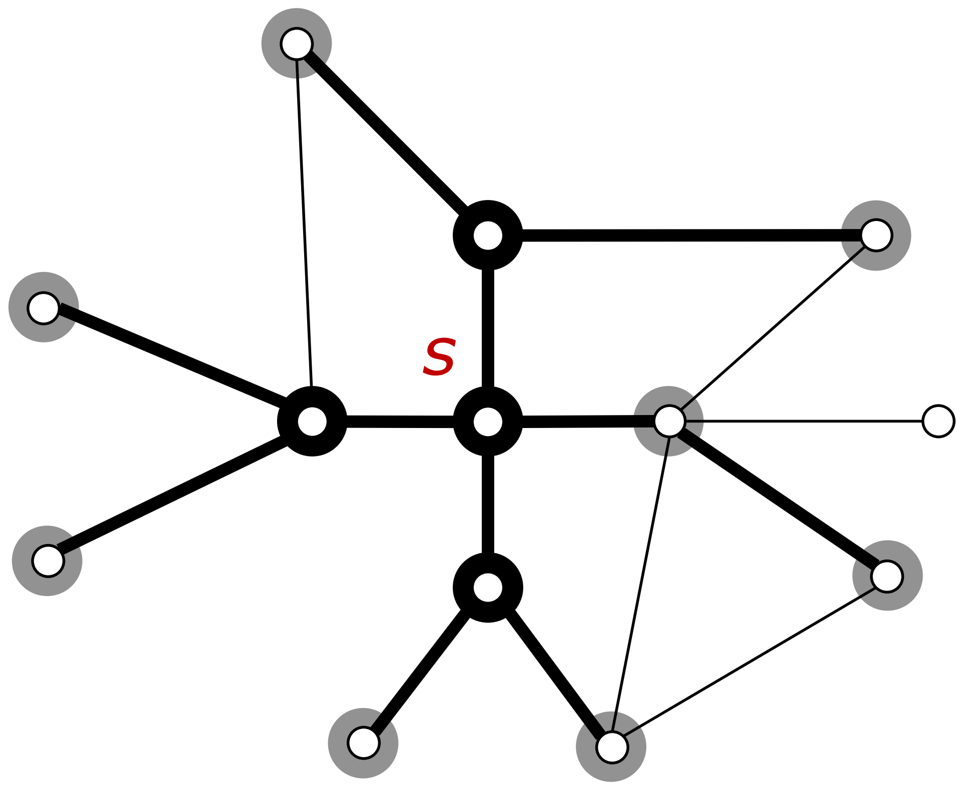
Busca em largura



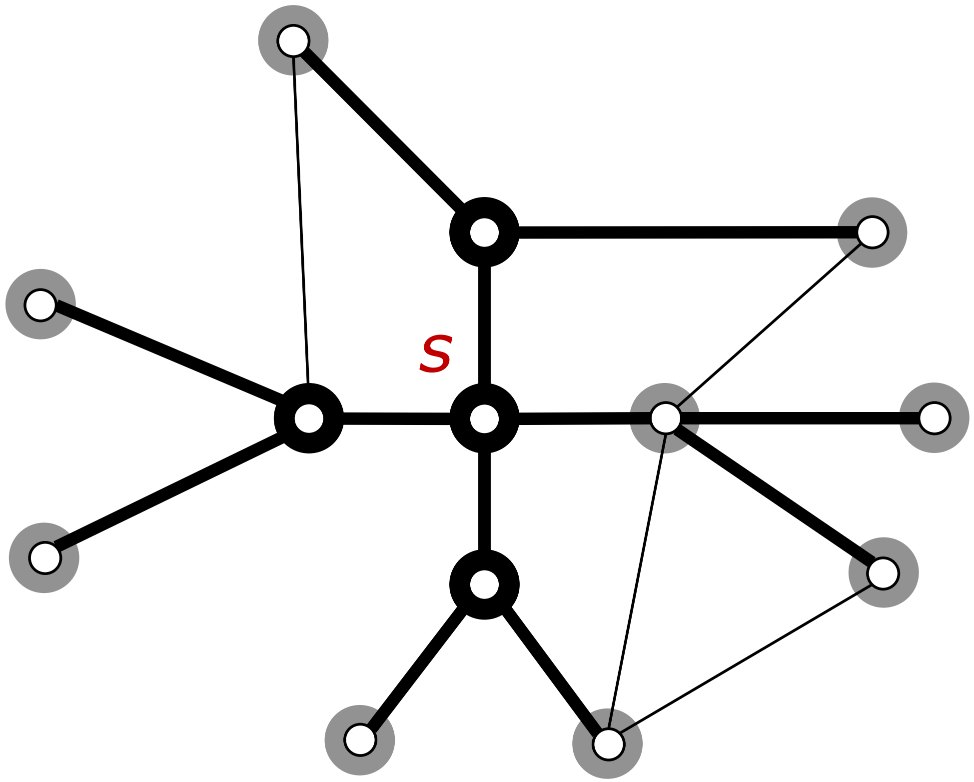
Busca em largura



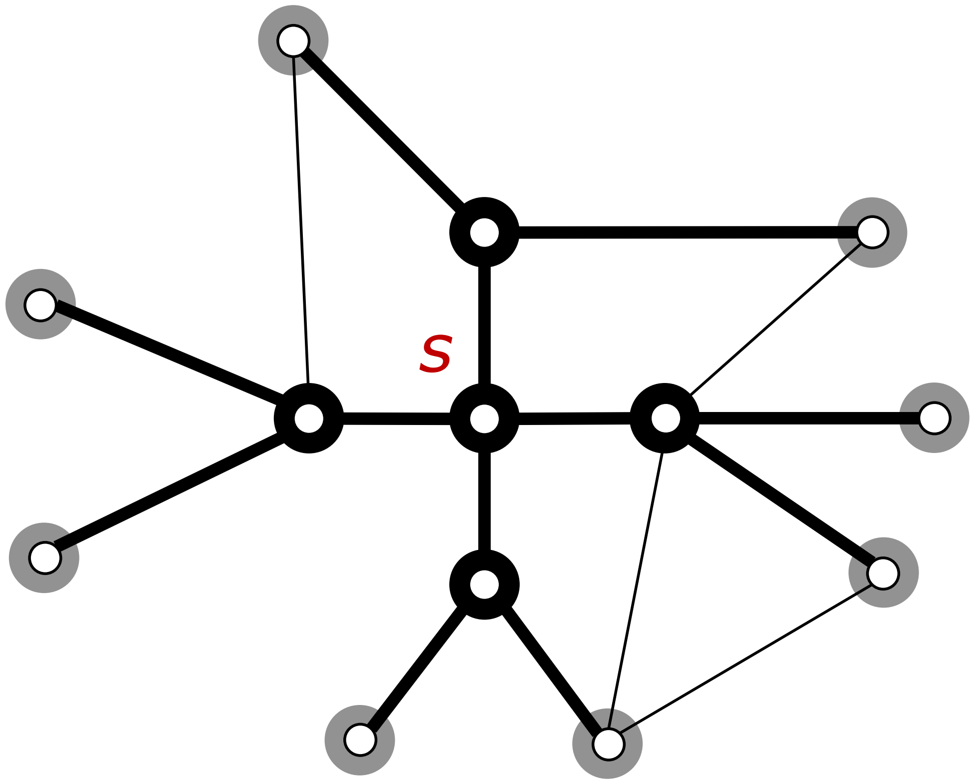
Busca em largura



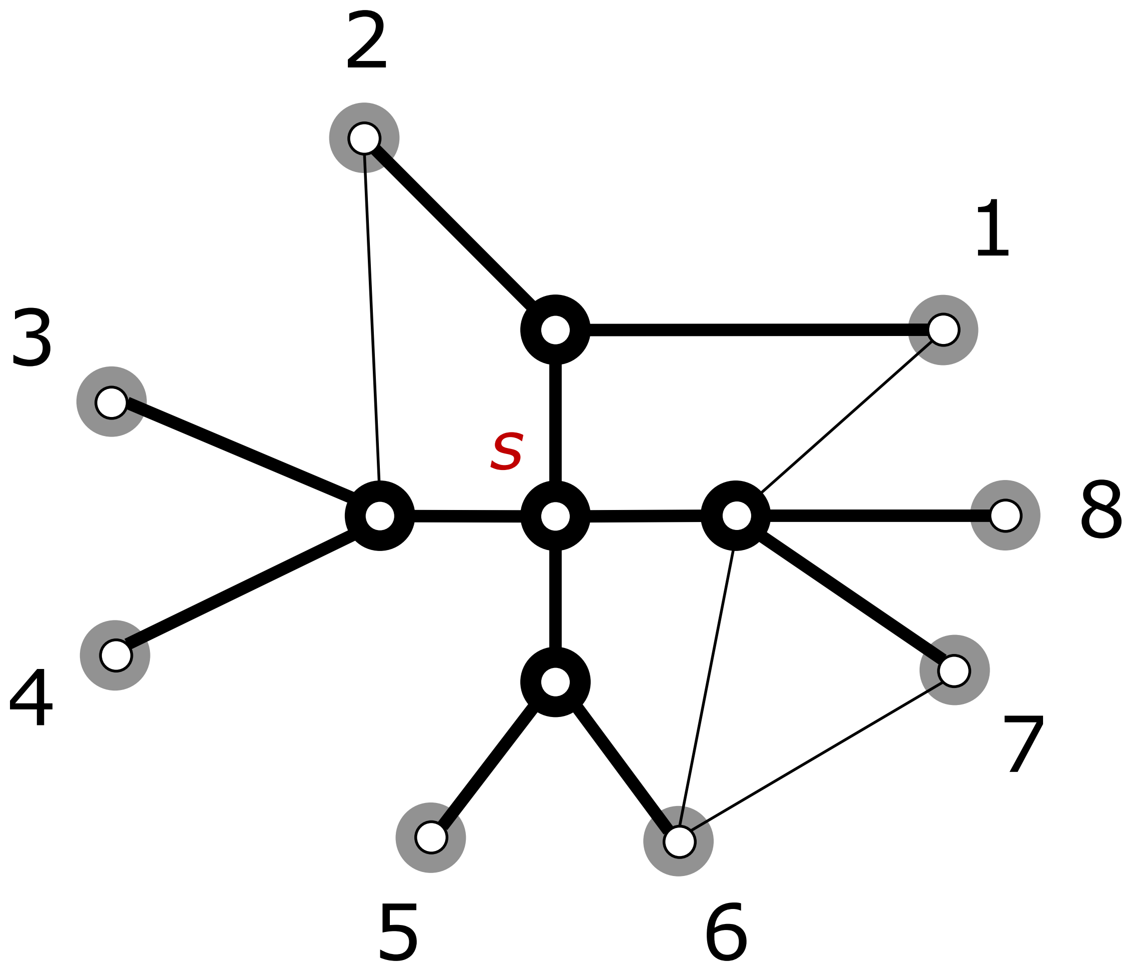
Busca em largura



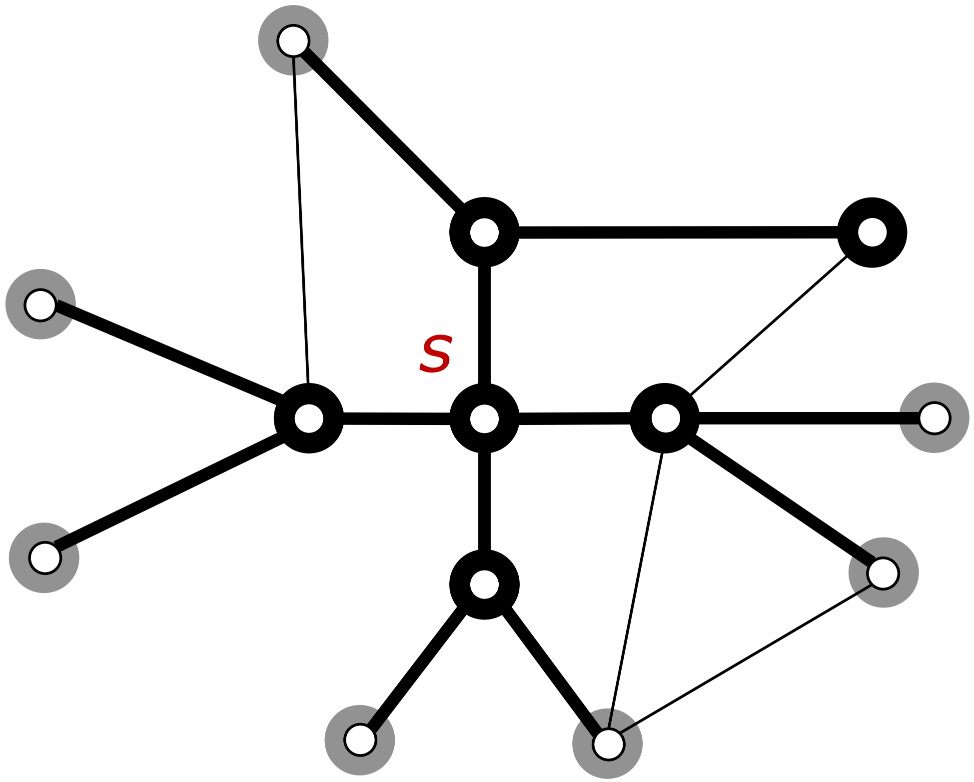
Busca em largura



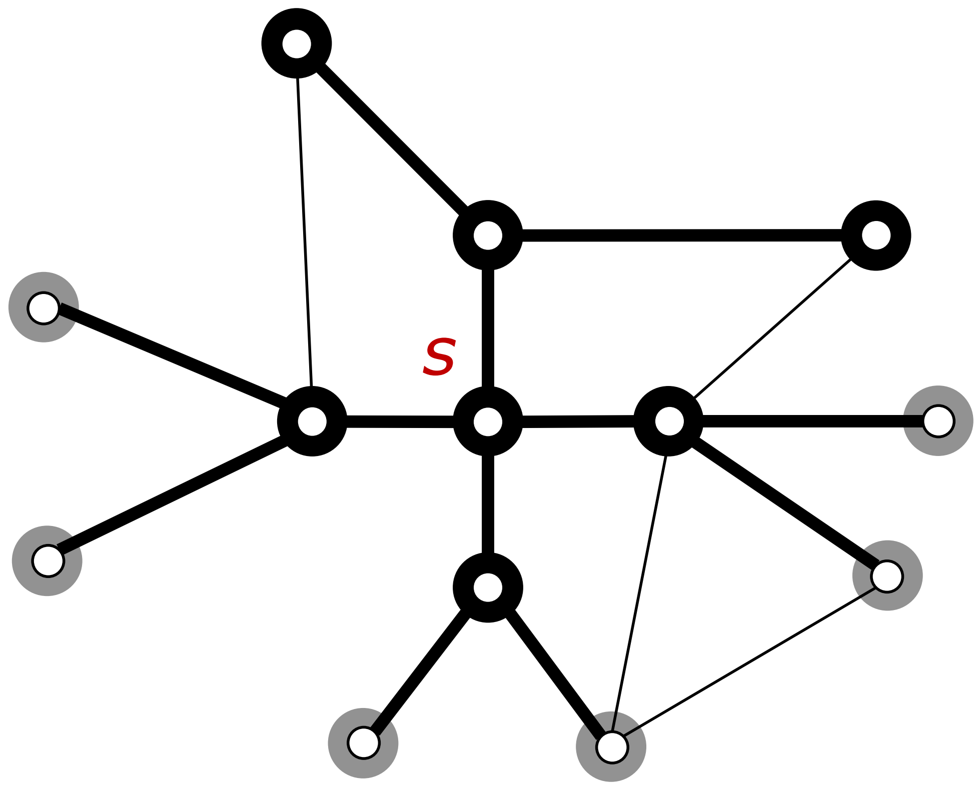
Busca em largura



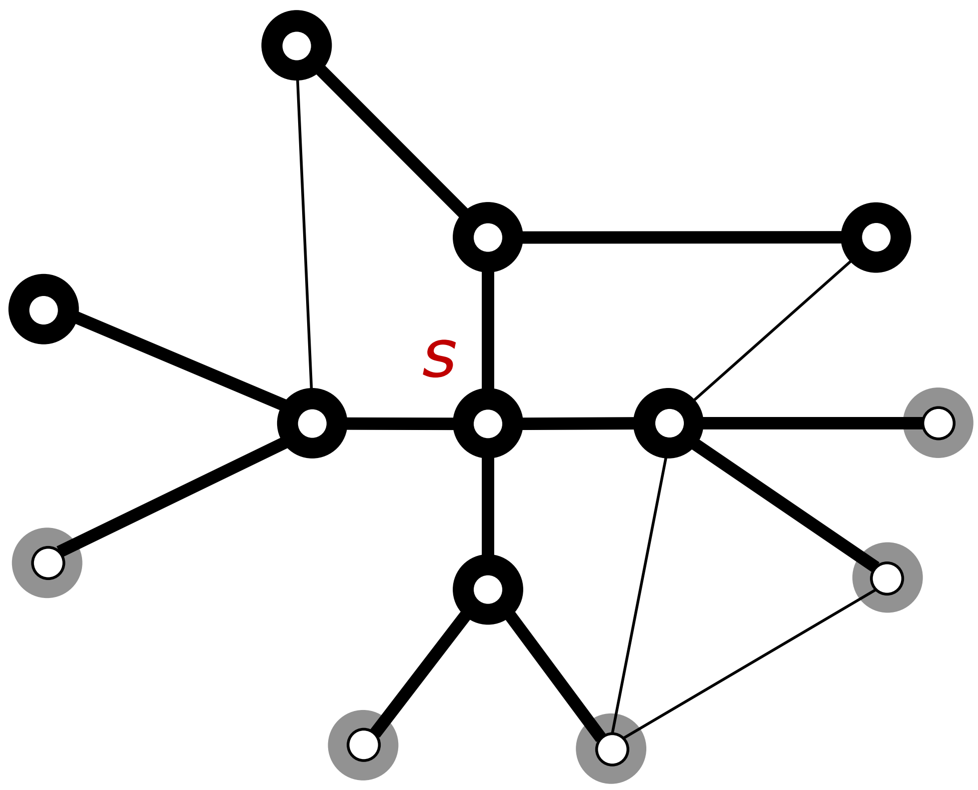
Busca em largura



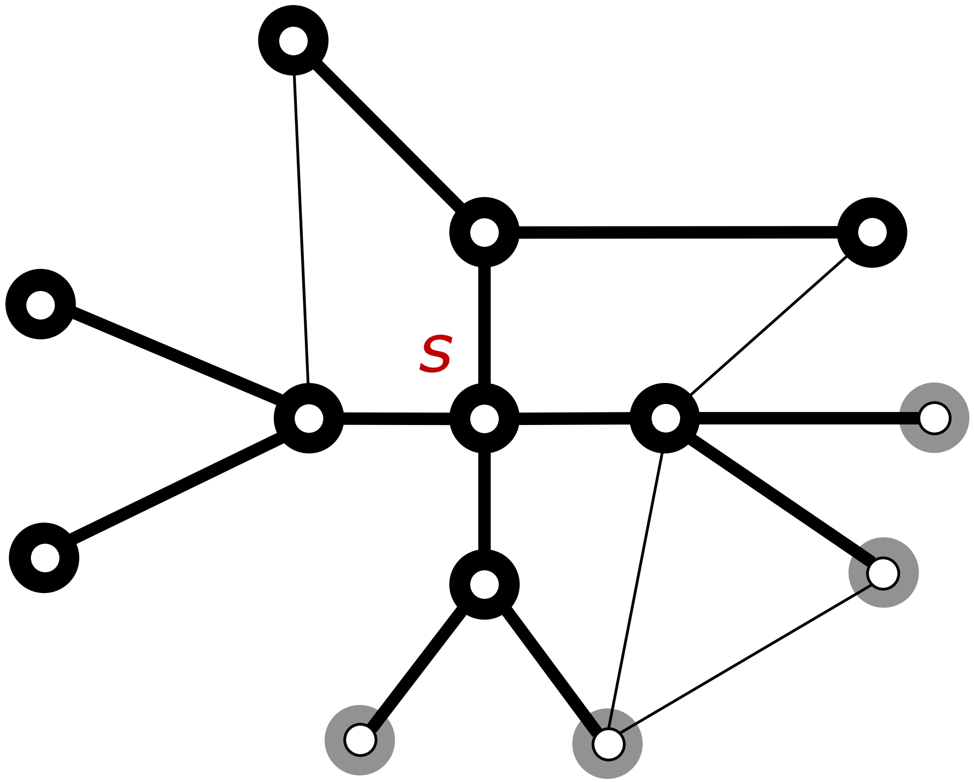
Busca em largura



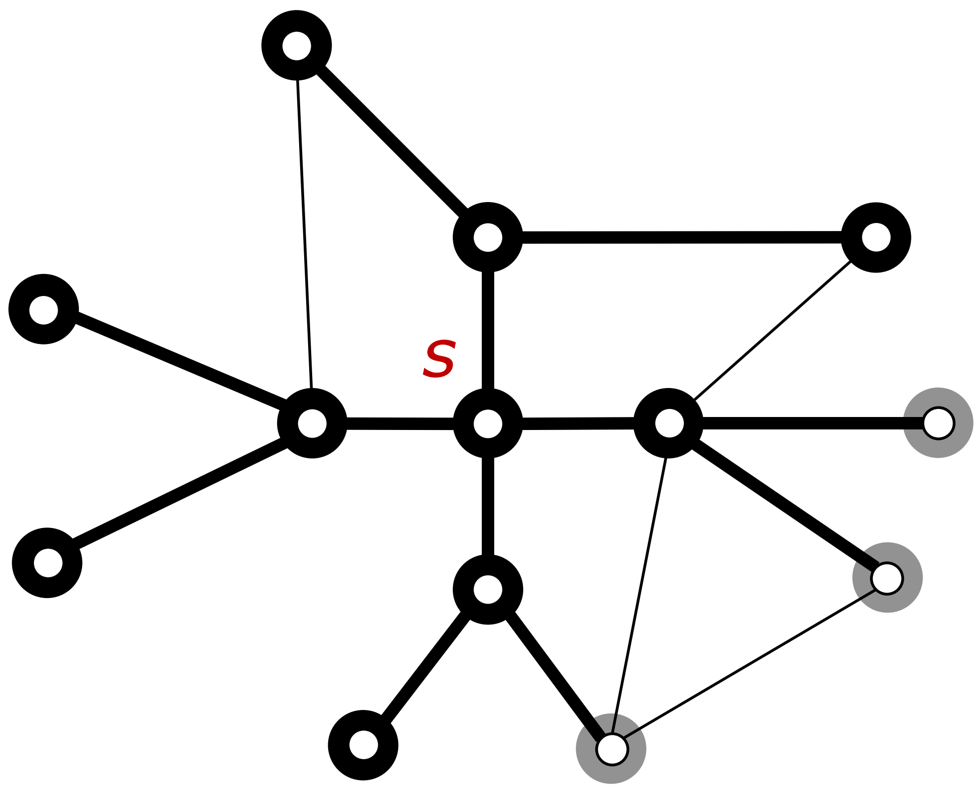
Busca em largura



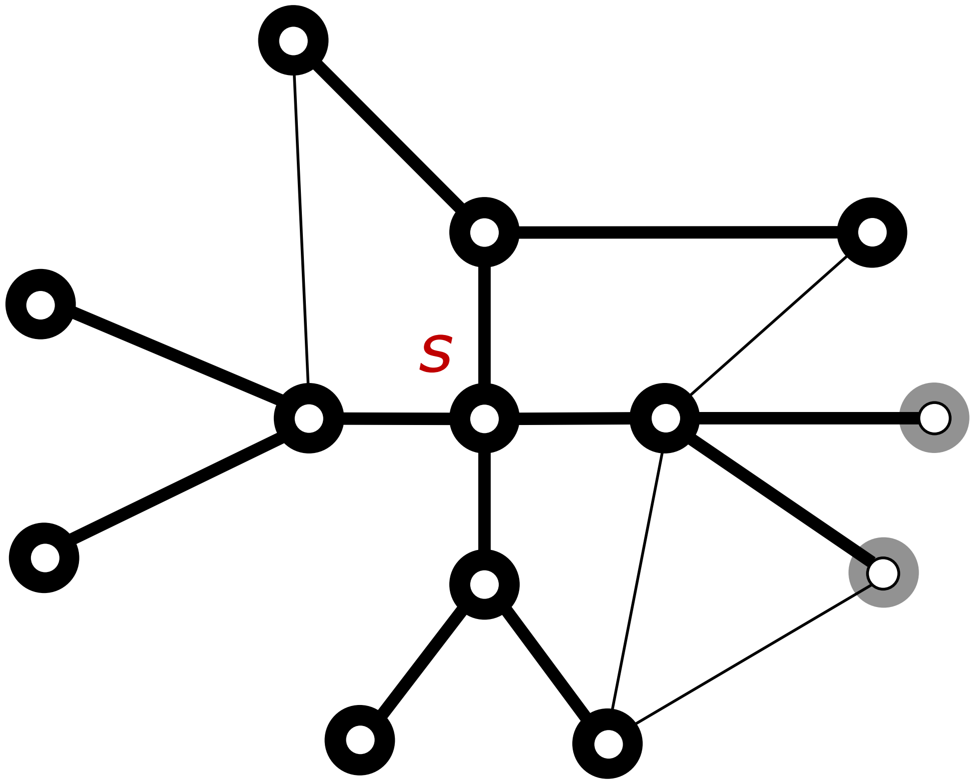
Busca em largura



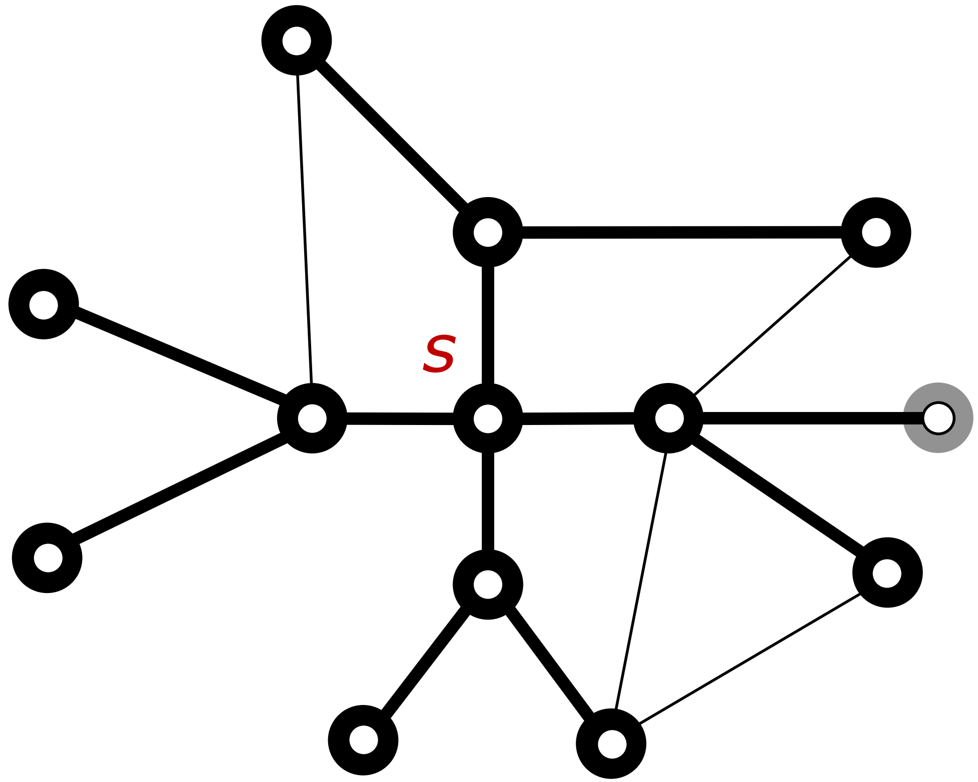
Busca em largura



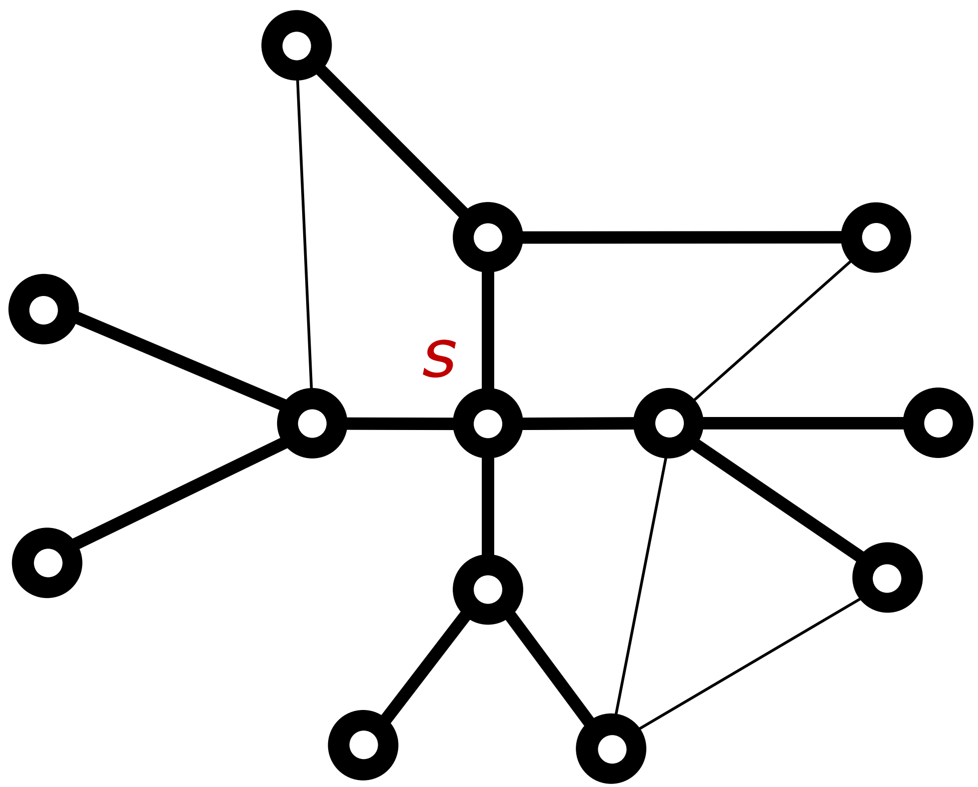
Busca em largura



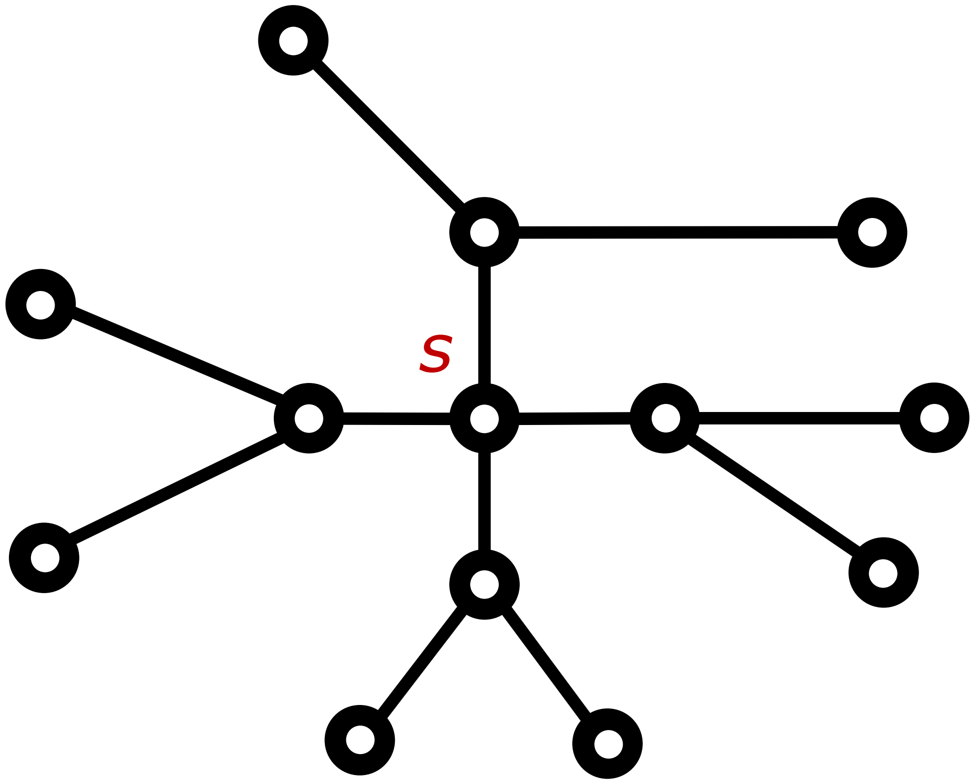
Busca em largura



Busca em largura

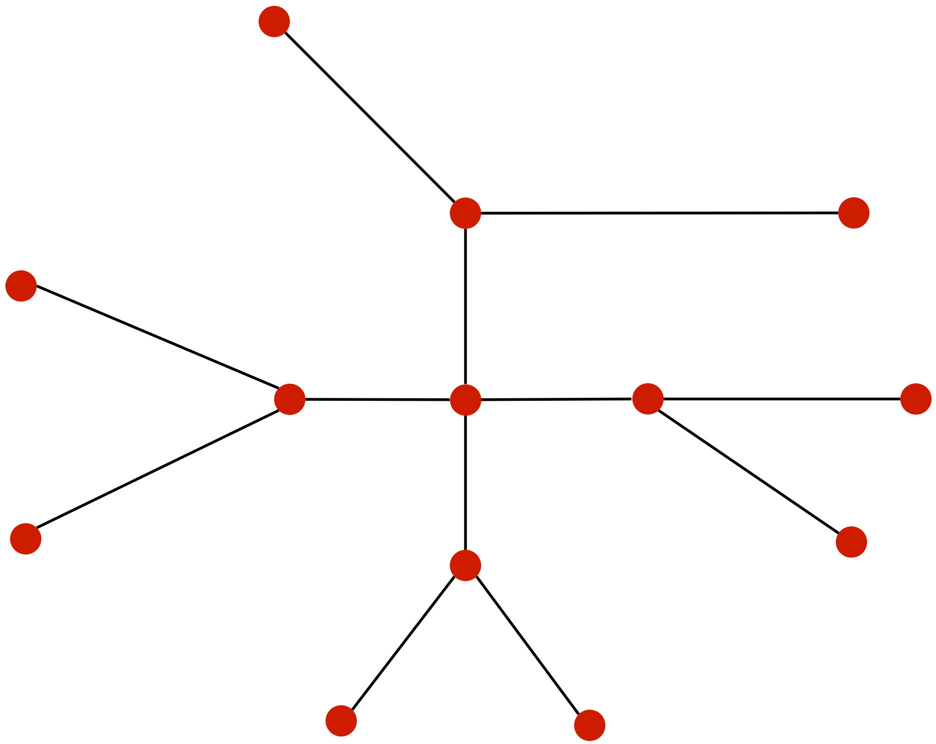


Busca em largura



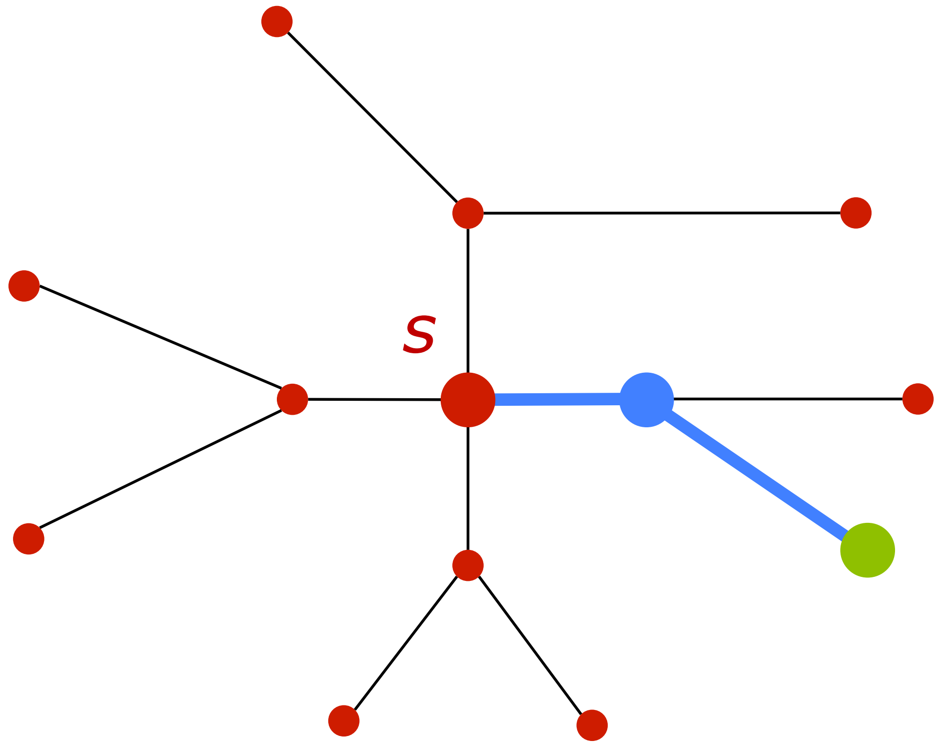
Busca em largura

Descobre uma árvore de busca em largura.



Busca em largura

Descobre um caminho mínimo de *s* até qualquer vértice.



Busca em largura

Elementos do algoritmo

Cada vértice:

1. Inicia **BRANCO**.
2. Muda para **CINZA** quando é descoberto.
3. Muda para **PRETO** quando todos seus vértices adjacentes já foram visitados.

Busca em largura

Elementos do algoritmo

A árvore de busca é dada por:

$G_\pi = (V, E_\pi)$, onde

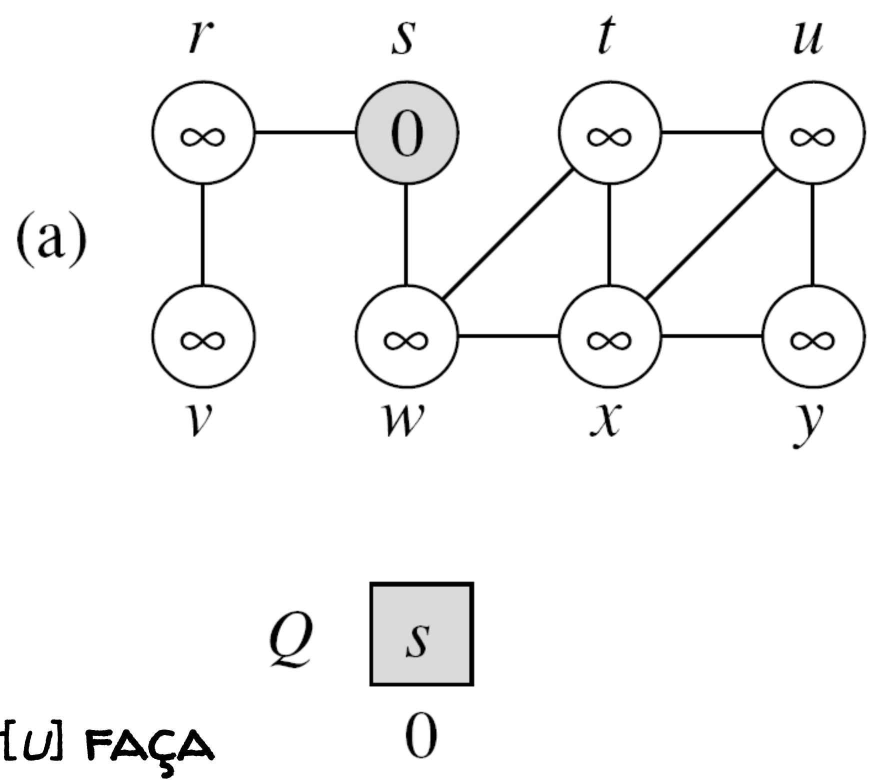
$E_\pi = \{(v.\pi, v) | : v \in V$ e $v.\pi \neq$ NULO$\}$ e

$v.\pi$ armazena o vértice predecessor de v.

$u.d$: acumula a "distância" da origem s até u.

Algoritmo BFS(G, s)

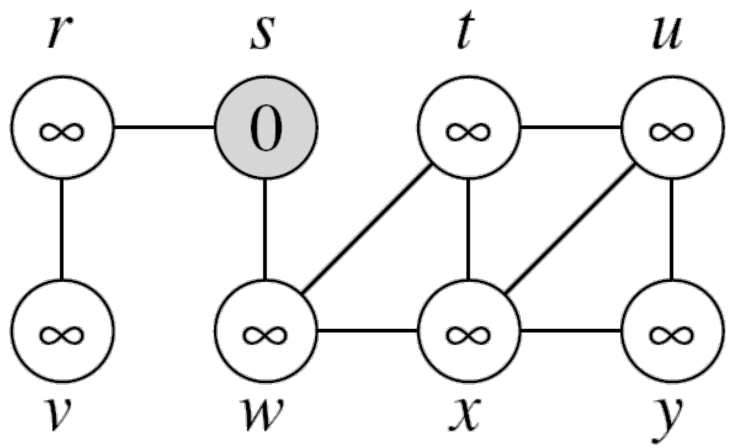
```
para cada vértice u ∈ G.V – {s} faça
    u.cor ← BRANCO
    u.d ← ∞
    u.π ← NULO
s.cor ← CINZA
s.d ← 0
s.π ← NULO
Q ← Ø
ENFILEIRA(Q, s)
enquanto Q ≠ Ø faça
    u ← DESENFILEIRA (Q)
    para cada vértice v ∈ G.adj[u] faça
        se v.cor = BRANCO então
            v.cor ← CINZA
            v.d ← u.d + 1
            v.π ← u
            ENFILEIRA(Q, v)
    u.cor ← PRETO
```

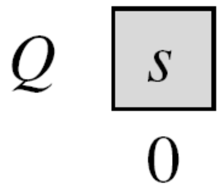


Algoritmo BFS(G, s)

```
para cada vértice u ∈ G.V − {s} faça
    u.cor ← BRANCO
    u.d ← ∞
    u.π ← NULO
s.cor ← CINZA
s.d ← 0
s.π ← NULO
Q ← ∅
ENFILEIRA(Q, s)
enquanto Q ≠ ∅ faça
    u ← DESENFILEIRA (Q)
    para cada vértice v ∈ G.adj[u] faça
        se v.cor = BRANCO então
            v.cor ← CINZA
            v.d ← u.d + 1
            v.π ← u
            ENFILEIRA(Q, v)
    u.cor ← PRETO
```

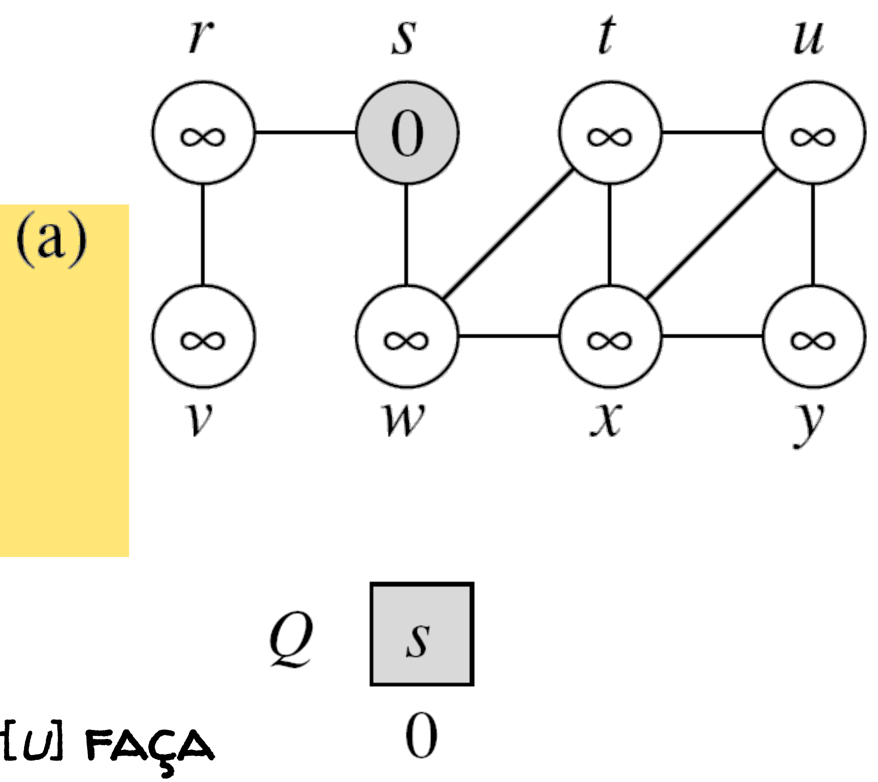
(a)





Algoritmo BFS(G, s)

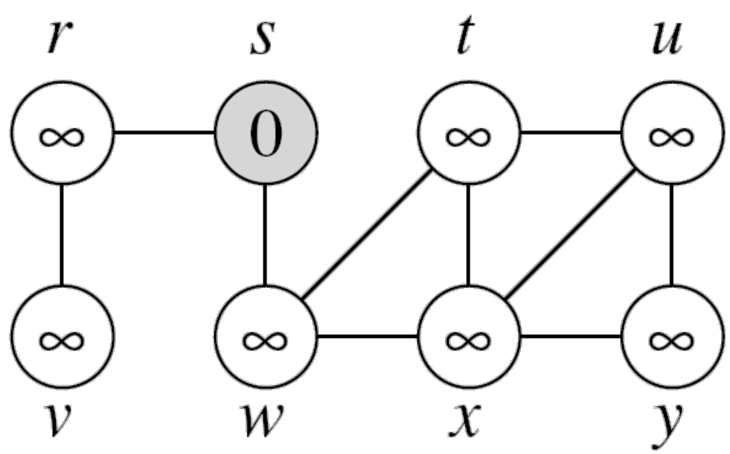
para cada vértice $u \in G.V - \{s\}$ faça
  $u.cor \leftarrow$ BRANCO
  $u.d \leftarrow \infty$
  $u.\pi \leftarrow$ NULO
$s.cor \leftarrow$ CINZA
$s.d \leftarrow 0$
$s.\pi \leftarrow$ NULO
$Q \leftarrow \emptyset$
ENFILEIRA(Q, s)
enquanto $Q \neq \emptyset$ faça
  $u \leftarrow$ DESENFILEIRA (Q)
  para cada vértice $v \in G.Adj[u]$ faça
    se $v.cor$ = BRANCO então
      $v.cor \leftarrow$ CINZA
      $v.d \leftarrow u.d + 1$
      $v.\pi \leftarrow u$
      ENFILEIRA(Q, v)
  $u.cor \leftarrow$ PRETO

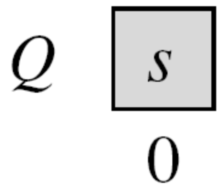


ALGORITMO BFS(G, s)

```
PARA CADA VÉRTICE u ∈ G.V − {s} FAÇA
        u.cor ← BRANCO
        u.d ← ∞
        u.π ← NULO
s.cor ← CINZA
s.d ← 0
s.π ← NULO
Q ← Ø
ENFILEIRA(Q, s)
ENQUANTO Q ≠ Ø FAÇA
        u ← DESENFILEIRA (Q)
        PARA CADA VÉRTICE v ∈ G.Adj[u] FAÇA
                SE v.cor = BRANCO ENTÃO
                        v.cor ← CINZA
                        v.d ← u.d + 1
                        v.π ← u
                        ENFILEIRA(Q, v)
        u.cor ← PRETO
```

(a)

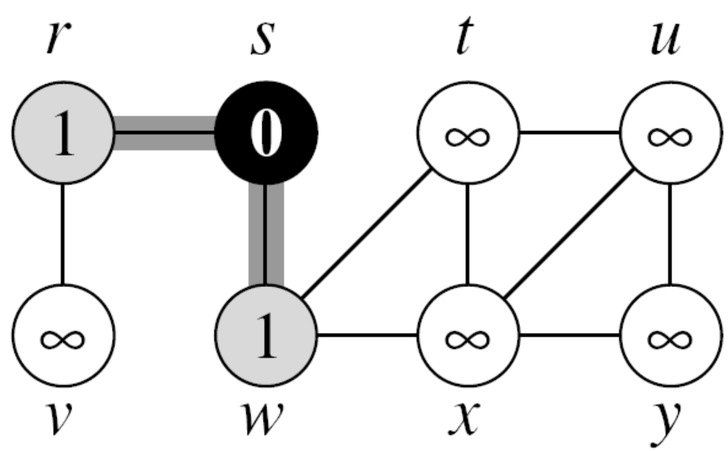


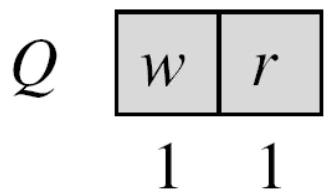


Algoritmo BFS(G, s)

```
para cada vértice u ∈ G.V − {s} faça
    u.cor ← BRANCO
    u.d ← ∞
    u.π ← NULO
s.cor ← CINZA
s.d ← 0
s.π ← NULO
Q ← Ø
ENFILEIRA(Q, s)
enquanto Q ≠ Ø faça
    u ← DESENFILEIRA (Q)
    para cada vértice v ∈ G.Adj[u] faça
        se v.cor = BRANCO então
            v.cor ← CINZA
            v.d ← u.d + 1
            v.π ← u
            ENFILEIRA(Q, v)
    u.cor ← PRETO
```

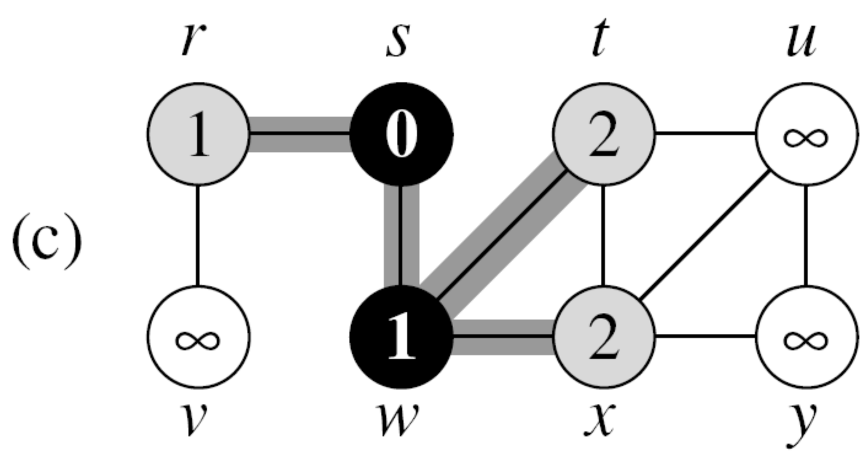
(b)

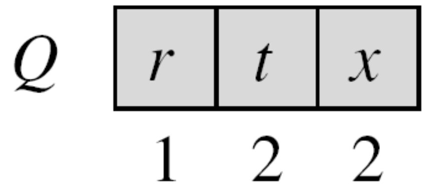




Algoritmo BFS(G, s)

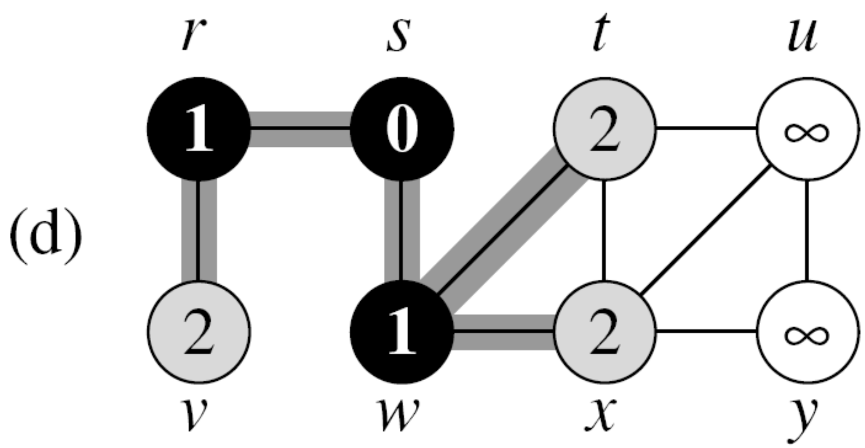
```
para cada vértice u ∈ G.V − {s} faça
        u.cor ← BRANCO
        u.d ← ∞
        u.π ← NULO
s.cor ← CINZA
s.d ← 0
s.π ← NULO
Q ← ∅
ENFILEIRA(Q, s)
enquanto Q ≠ ∅ faça
        u ← DESENFILEIRA (Q)
        para cada vértice v ∈ G.Adj[u] faça
                se v.cor = BRANCO então
                        v.cor ← CINZA
                        v.d ← u.d + 1
                        v.π ← u
                        ENFILEIRA(Q, v)
        u.cor ← PRETO
```

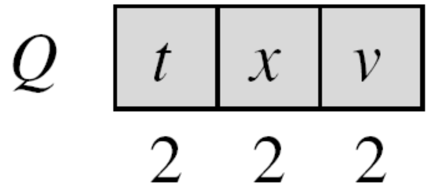




Algoritmo BFS(G, s)

para cada vértice $u \in G.V - \{s\}$ faça
    $u.cor \leftarrow$ BRANCO
    $u.d \leftarrow \infty$
    $u.\pi \leftarrow$ NULO
$s.cor \leftarrow$ CINZA
$s.d \leftarrow 0$
$s.\pi \leftarrow$ NULO
$Q \leftarrow \emptyset$
ENFILEIRA(Q, s)
enquanto $Q \neq \emptyset$ faça
    $u \leftarrow$ DESENFILEIRA (Q)
    para cada vértice $v \in G.Adj[u]$ faça
        se $v.cor =$ BRANCO então
            $v.cor \leftarrow$ CINZA
            $v.d \leftarrow u.d + 1$
            $v.\pi \leftarrow u$
            ENFILEIRA(Q, v)
    $u.cor \leftarrow$ PRETO

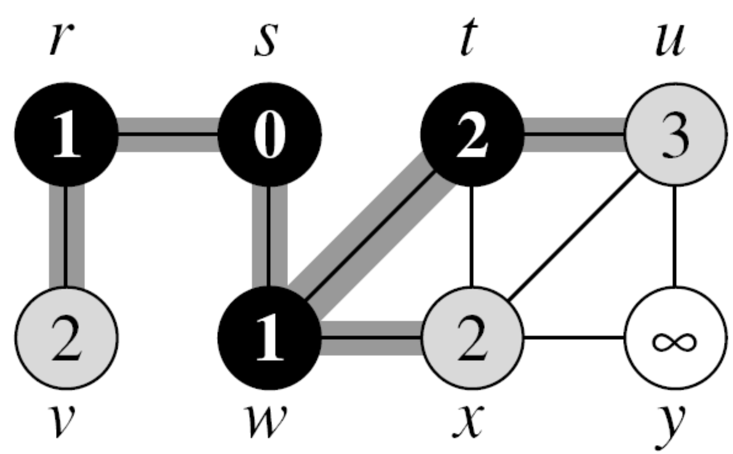


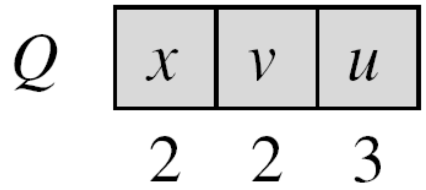


ALGORITMO BFS(G, s)

PARA CADA VÉRTICE $u \in G.V - \{s\}$ FAÇA
    $u.cor \leftarrow$ BRANCO
    $u.d \leftarrow \infty$
    $u.\pi \leftarrow$ NULO
$s.cor \leftarrow$ CINZA
$s.d \leftarrow 0$
$s.\pi \leftarrow$ NULO
$Q \leftarrow \emptyset$
ENFILEIRA(Q, s)
ENQUANTO $Q \neq \emptyset$ FAÇA
    $u \leftarrow$ DESENFILEIRA (Q)
    PARA CADA VÉRTICE $v \in G.Adj[u]$ FAÇA
        SE $v.cor$ = BRANCO ENTÃO
            $v.cor \leftarrow$ CINZA
            $v.d \leftarrow u.d + 1$
            $v.\pi \leftarrow u$
            ENFILEIRA(Q, v)
    $u.cor \leftarrow$ PRETO

(e)

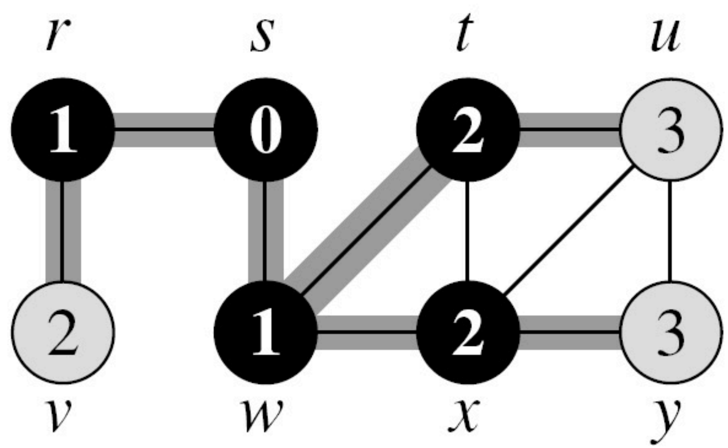




ALGORITMO BFS(G, s)

PARA CADA VÉRTICE $u \in G.V - \{s\}$ FAÇA
    u.COR ← BRANCO
    u.D ← ∞
    $u.\pi$ ← NULO
s.COR ← CINZA
s.D ← 0
$s.\pi$ ← NULO
$Q \leftarrow \emptyset$
ENFILEIRA(Q, s)
ENQUANTO $Q \neq \emptyset$ FAÇA
    u ← DESENFILEIRA (Q)
    PARA CADA VÉRTICE $v \in G$.ADJ[u] FAÇA
        SE v.COR = BRANCO ENTÃO
            v.COR ← CINZA
            v.D ← u.D + 1
            $v.\pi$ ← u
            ENFILEIRA(Q, v)
    u.COR ← PRETO

(f)



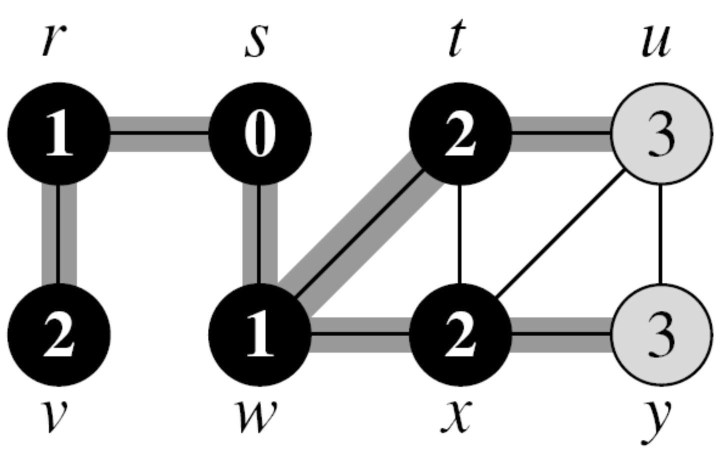
Q

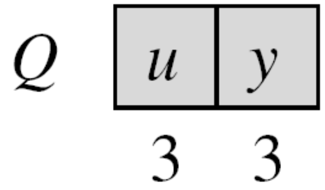
v	u	y
2	3	3

Algoritmo BFS(G, s)

```
para cada vértice u ∈ G.V − {s} faça
        u.cor ← BRANCO
        u.d ← ∞
        u.π ← NULO
s.cor ← CINZA
s.d ← 0
s.π ← NULO
Q ← ∅
ENFILEIRA(Q, s)
enquanto Q ≠ ∅ faça
        u ← DESENFILEIRA (Q)
        para cada vértice v ∈ G.adj[u] faça
                se v.cor = BRANCO então
                        v.cor ← CINZA
                        v.d ← u.d + 1
                        v.π ← u
                        ENFILEIRA(Q, v)
        u.cor ← PRETO
```

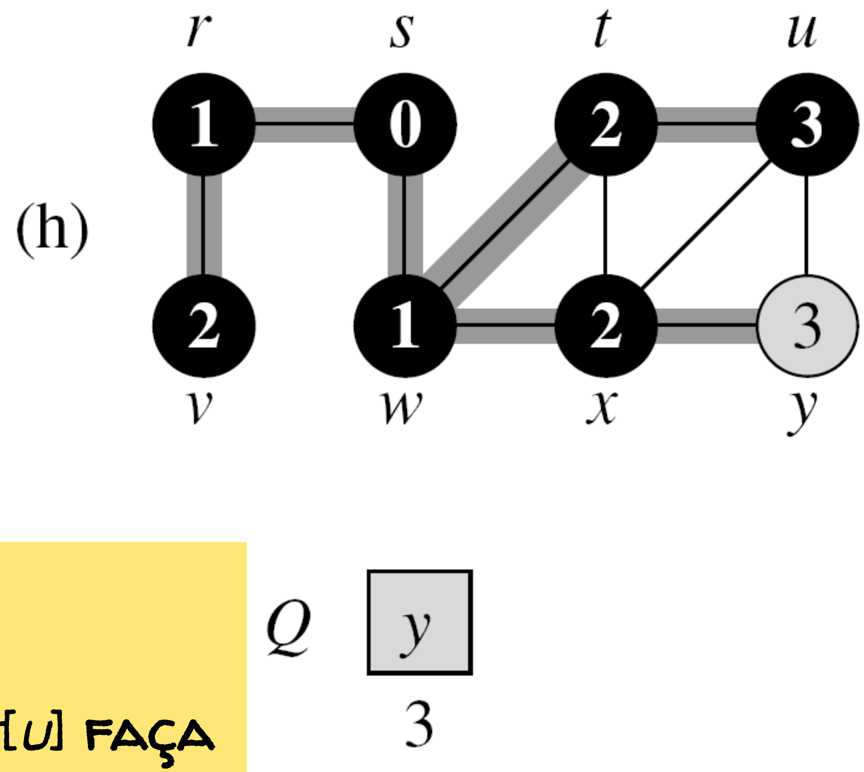
(g)





Algoritmo BFS(G, s)

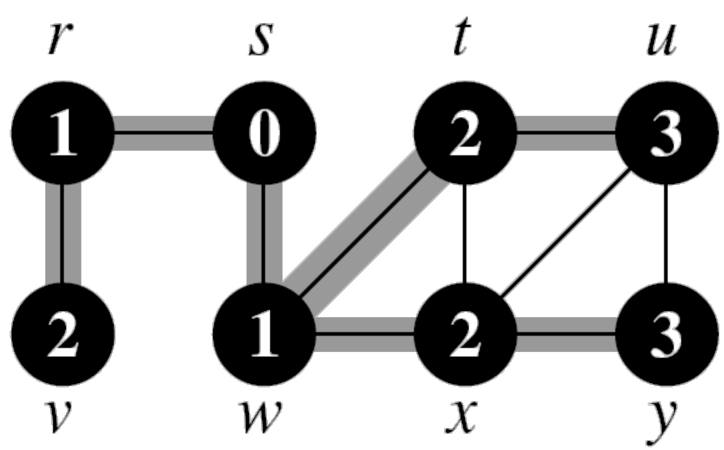
```
para cada vértice u ∈ G.V − {s} faça
    u.cor ← BRANCO
    u.d ← ∞
    u.π ← NULO
s.cor ← CINZA
s.d ← 0
s.π ← NULO
Q ← ∅
ENFILEIRA(Q, s)
enquanto Q ≠ ∅ faça
    u ← DESENFILEIRA (Q)
    para cada vértice v ∈ G.Adj[u] faça
        se v.cor = BRANCO então
            v.cor ← CINZA
            v.d ← u.d + 1
            v.π ← u
            ENFILEIRA(Q, v)
    u.cor ← PRETO
```



Algoritmo BFS(G, s)

```
para cada vértice u ∈ G.V − {s} faça
        u.cor ← BRANCO
        u.d ← ∞
        u.π ← NULO
    s.cor ← CINZA
    s.d ← 0
    s.π ← NULO
    Q ← ∅
    ENFILEIRA(Q, s)
    enquanto Q ≠ ∅ faça
        u ← DESENFILEIRA (Q)
        para cada vértice v ∈ G.Adj[u] faça
            se v.cor = BRANCO então
                v.cor ← CINZA
                v.d ← u.d + 1
                v.π ← u
                ENFILEIRA(Q, v)
        u.cor ← PRETO
```

(i)

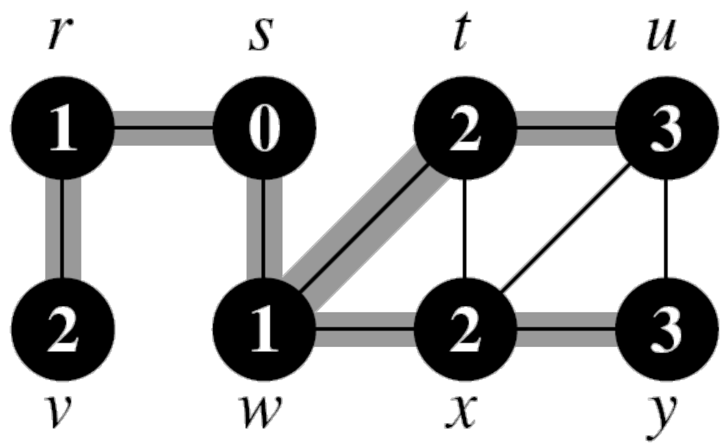


Q $\emptyset$

Algoritmo BFS(G, s)

```
para cada vértice u ∈ G.V − {s} faça
        u.cor ← BRANCO
        u.d ← ∞
        u.π ← NULO
  s.cor ← CINZA
  s.d ← 0
  s.π ← NULO
  Q ← Ø
  ENFILEIRA(Q, s)
  enquanto Q ≠ Ø faça
        u ← DESENFILEIRA (Q)
        para cada vértice v ∈ G.Adj[u] faça
              se v.cor = BRANCO então
                    v.cor ← CINZA
                    v.d ← u.d + 1
                    v.π ← u
                    ENFILEIRA(Q, v)
        u.cor ← PRETO
FIM
```

(i)



Q	Ø

Busca em largura

Propriedades da BFS:

Todos os vértices são explorados.

A busca em largura obtém o caminho mínimo de um vértice origem *s* até outro vértice qualquer *v*.

Procedimento BFS constrói uma árvore de busca em largura armazenada em G_π.

Busca em largura

Eficiência de tempo

Matriz de adjacência: $\Theta(|V|^2)$.

Lista de adjacência: $\Theta(|V| + |E|)$.

Busca em largura

Aplicações da BFS:

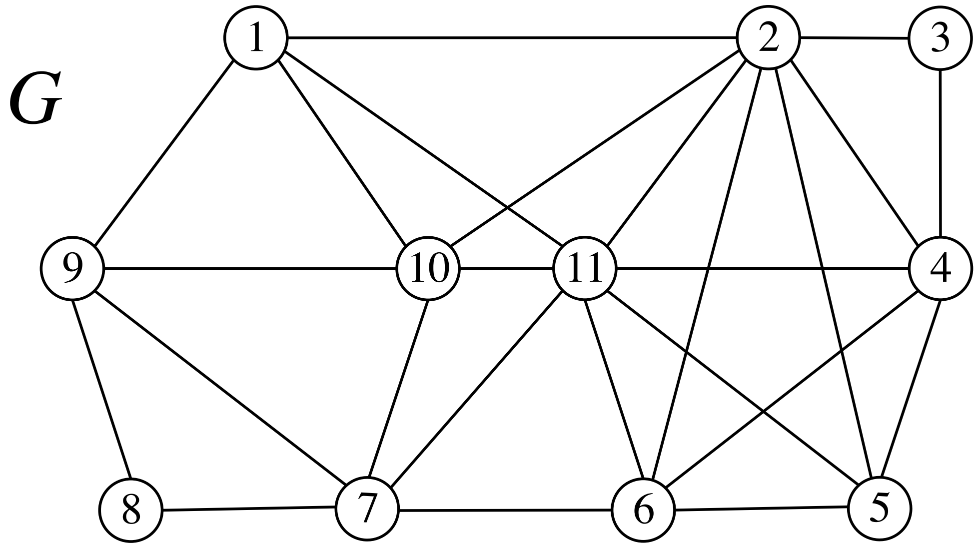
Modelo para o algoritmo de Dijkstra para caminho mais curto de origem única e o algoritmo de Prim para árvore geradora mínima.

Bibliografia

- BONDY, J.; MURTY, U. ***Graduate Texts in Mathematics series: Graph Theory***. Springer, 2008. Volume 244.
- DIESTEL, Reinhard. ***Graduate Texts in Mathematics series: Graph Theory***. New York: Springer-Verlag, 2000. Volume 173.
- CORMEN, T. H. , LEISERSON, C. E., RIVEST, R.L., STEIN, C. ***Introduction to Algorithms***, 3rd edition, MIT Press, 2009.
- SZWARCFITER, J. ***Grafos e Algoritmos Computacionais***. Rio de Janeiro: 2ª. Ed. Campus, 1986.
- ZIVIANI, Nivio. ***Projeto de Algoritmos com Implementação em Pascal e C***. 3a Edição revisada e ampliada de 2010. São Paulo: Thomson Learning, 2010.

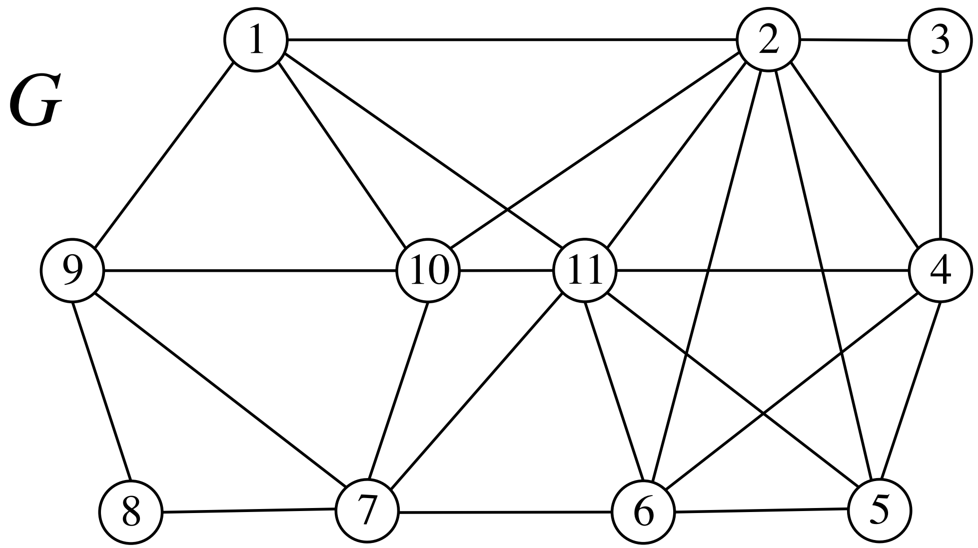
Exercícios

1. Seja G o grafo abaixo. Determine uma árvore em largura em G iniciando pelo vértice 5. Escolha sempre o menor vértice adjacente.



Exercícios

2. Determine outra árvore em largura em G iniciando pelo vértice 8. A árvore obtida é igual à árvore da questão 1?



Bibliografia

- BONDY, J.; MURTY, U. ***Graduate Texts in Mathematics series: Graph Theory***. Springer, 2008. Volume 244.
- DIESTEL, Reinhard. ***Graduate Texts in Mathematics series: Graph Theory***. New York: Springer-Verlag, 2000. Volume 173.
- CORMEN, T. H. , LEISERSON, C. E., RIVEST, R.L., STEIN, C. ***Introduction to Algorithms***, 3rd edition, MIT Press, 2009.
- SZWARCFITER, J. ***Grafos e Algoritmos Computacionais***. Rio de Janeiro: 2ª. Ed. Campus, 1986.
- ZIVIANI, Nivio. ***Projeto de Algoritmos com Implementação em Pascal e C***. 3a Edição revisada e ampliada de 2010. São Paulo: Thomson Learning, 2010.