BFS

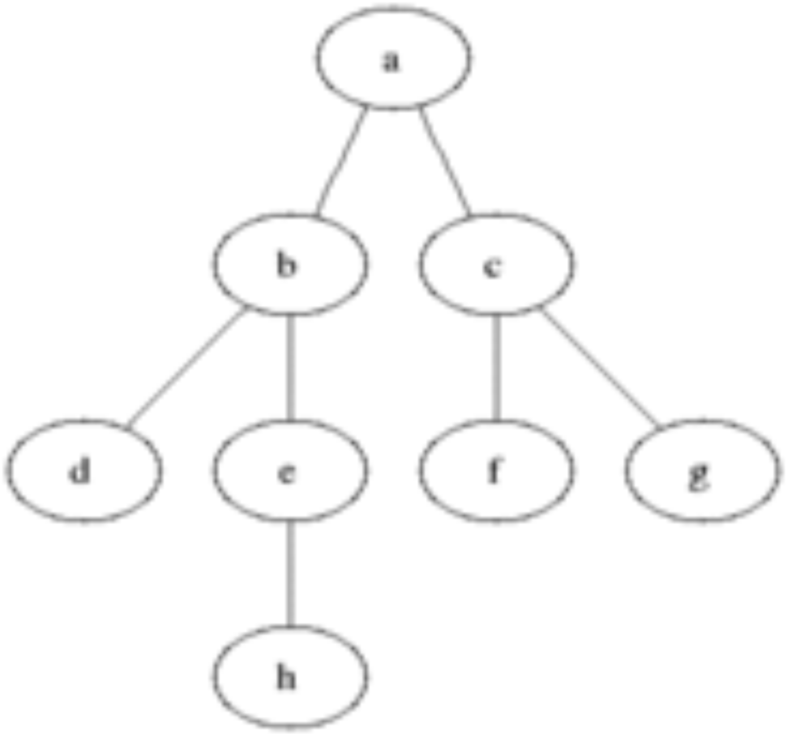
comp-420

Breadth-First Search

- Uno de los algoritmos más simples para recorrer gráficas.
- Se usa como arquetipo para muchos algoritmos importantes con gráficas.
- Dada una gráfica G=(V,E) y un vértice fuente s, BFS explora las aristas de G sistemáticamente para descubrir cada vértice alcazable desde s.
- El algoritmo calcula la distancia (minimo número de aristas) desde s hasta cualquier vértice alcanzable en G.
- Genera un árbol de anchura (BF tree) con raíz s y que contiene todos los vértices alcanzables en G.
- Para cualquier vértice v alcanzable desde s, el camino en el árbol desde s hasta v corresponde al camino más corto de s a v en G.

Breadth-First Search

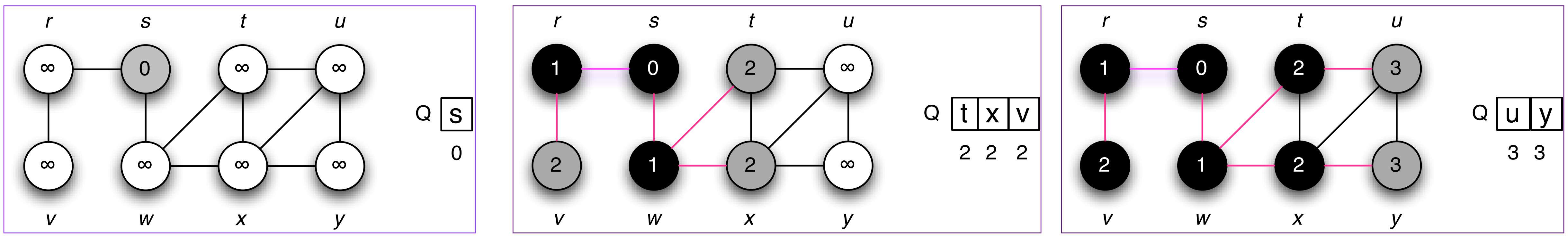
- Supongamos una representación utilizando listas de adyacencia.
- El color de cada vértice u se almacena en la variable color[u].
- El predecesor de u se almacena en la variable π[u].
- La distancia de la fuente s al vértice u calculada por el algoritmo se almacena en d[u].
- Se mantiene también una cola Q (first-in, first-out) que maneja el conjunto de vértices grises.



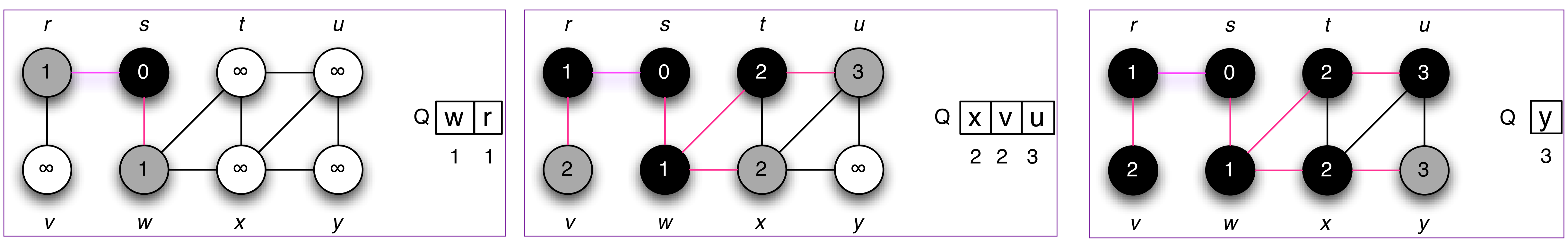
BFS(G,s)

for each vertex $u \in V[G] - \{s\}$
do $color[u] \leftarrow$ WHITE
$d[u] \leftarrow \infty$
$\pi[u] \leftarrow$ NIL
$color[s] \leftarrow$ GRAY
$d[s] \leftarrow 0$
$\pi[s] \leftarrow$ NIL
$Q \leftarrow \emptyset$
ENQUEUE(Q, s)
while $Q \neq \emptyset$
do $u \leftarrow$ DEQUEUE(Q)
for each $v \in Adj[u]$
do if $color[v] =$ WHITE
then $color[v] \leftarrow$ GRAY
$d[v] \leftarrow d[u] + 1$
$\pi[v] \leftarrow u$
ENQUEUE(Q, v)
$color[u] \leftarrow$ BLACK

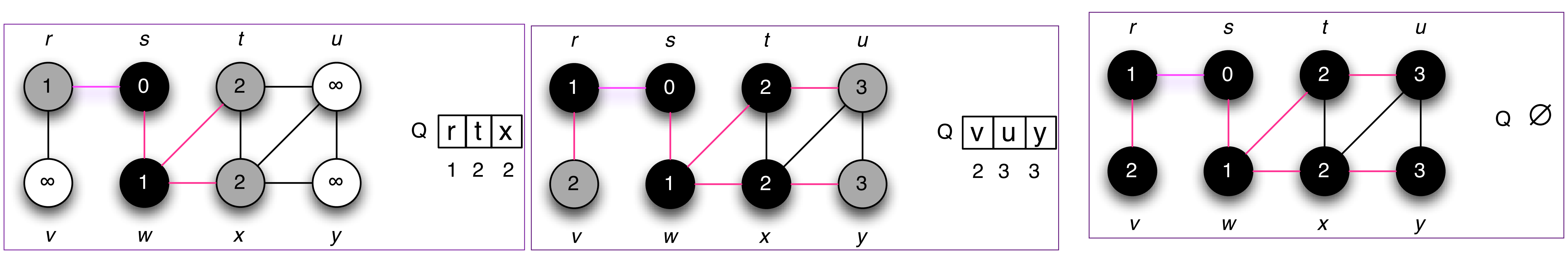
Breadth-first search



Líneas 1-9



Ciclo 11-17



Breadth-First Search

- En el test de la línea 10, la cola Q consta de ¿qué vertices?
  - el conjunto de vértices grises.
- El resultado de BFS depende solamente del orden en que los vecinos de un nodo dado son visitados en la línea 12 : El BF-tree puede variar pero las distancias d calculadas no.
- Pueden ver una buena implementación de gráficas y BFS en C++ en la librería Boost Graph Library.

Implementación de BFS

- Suponiendo una estructura de nodo:

```
struct Vertex {
  ...
  std::vector<int> out;
  ...
};
```

- y un arreglo de vértices (el algoritmo usa los índices del arreglo para manejar los vértices).

```
std::vector <Vertex> graph(vertices);
```

- el algoritmo comienza en start y regresa true si hay un camino dirigido desde start hasta end.

```cpp
bool BFS( const std::vector<Vertex>& graph, int start, int end ){
  std::queue<int> next;
  std::vector<int> parent(graph.size(),-1);
  next.push(start);
  while( !next.empty()) {
    int u = next.front();
    next.pop();
    // Aqui examinamos al vertice u en el grafo, por ejemplo:
    if( u == end )
      return true;
    for( std::vector<int>::const_iterator j= graph[u].out.begin(); j++){
      // mirar a los vecinos
      int v = *j;
      if( parent[v] == -1){
        // si v no ha sido visitado
        parent[v] = u;
        next.push(v);
      }
    }
  }
  return false;
}
```

BFS: tiempo de cálculo

- Depués de la inicialización ningún nodo será dibujado de blanco, por lo que la prueba de la línea 13 asegura que cada vértice es metido y por lo tanto sacado a/de la cola a lo más una vez.
- Las operaciones ENQUEUE y DEQUEUE toman $O(1)$.
- El tiempo total tomado por las operaciones de la cola es:
  - $O(V)$.
- La lista de adyacencia para cada vértice se recorre a lo más una vez. ¿cuándo?
  - cuando sale el vértice de Q.
- Ya que la suma de las longitudes de todas las listas de adyacencia es $O(E)$, el tiempo total requerido para recorrer las listas es $O(E)$.

BFS: tiempo de cálculo

- ¿Cuántas veces se pone un nodo en la cola? (operación ENQUEUE).
- ¿Cuánto tiempo se necesita para inicializar el problema?
  - O(V).
- ¿Cuál es el timpo total de ejecución de BFS?
  - O(V+E)

BFS: caminos más cortos

- BFS encuentra la distancia más corta a cada vértice alcanzable en una gráfica $G=(V,E)$ a partir de una fuente s.
- Definimos la distancia más corta $\delta(s,v)$ de s a v como el mínimo número de aristas en cualquier camino del vértice s al vértice v.
- Si no existe un camino de s hasta v entonces $\delta(s,v) = \infty$.
- Un camino de largo $\delta(s,v)$ desde s hasta v se conoce como el camino más corto en esa gráfica.

Caminos más cortos: propiedades

- Lema 1: Sea $G=(V,E)$ un grafo dirigido o no dirigido, y sea $s \in V$ un vértice arbitrario. Entonces para cualquier arista $(u,v) \in E$:

$$\delta(s,v) < \delta(s,u)+1$$

- Prueba:
    - si u es alcanzable desde s, entonces también v.
    - en este caso, el camino más corto desde s hasta v no puede ser más largo que el camino más corto a partir de s hasta u, seguido por la arista (u,v) y la desigualdad se mantiene.
    - En caso que no exista camino entre s y u, entonces $\delta(s,u) = \infty$ y la desigualdad se mantiene.

Caminos más cortos: propiedades

- Lema 2: Sea $G=(V,E)$ un grafo dirigido o no dirigido, y supongamos que BFS se ejecuta en G desde un vértice fuente $s \in V$. Al término de la ejecución, para cada vértice $v \in V$, el valor $d[v]$ calculado satisface:

$$d[v] \geq \delta(s,v)$$

- Prueba:
  - Usando inducción en el número de operaciones ENQUEUE:
  - La base de la inducción es la situación que ocurre inmediatamente después que s es puesta en Q en la línea 9 de BFS.
  - La hipótesis inductiva se mantiene ya que $d[s]=0=\delta(s,s)$ y $d[v]=\infty \geq \delta(s,v)$ para todo $v \in V - \{s\}$.

Caminos más cortos: propiedades

- para el paso inductivo, consideramos un vértice blanco v que es descubierto durante la búsqueda a partir de un vértice u.
- la hipótesis inductiva supone que $d[u] \geq \delta(s,u)$.
- De la asignación realizada en la línea 15 y por el Lema 1, obtenemos:

$$\begin{aligned} d[v] &= d[u] + 1 \\ &\geq \delta(s,u) + 1 \\ &\geq \delta(s,v). \end{aligned}$$

- el vértice v es puesto en la cola y nunca se vuelve a poner en Q porque ahora es gris y la cláusula then de las líneas 14 a 17 se ejecuta solo para vértices blancos.
- El valor de $d[v]$ nunca vuelve a cambiar, manteniendo la hipótesis inductiva.

Caminos más cortos: propiedades

- Lema 3: Supongamos la ejecución de BFS en la gráfica G=(V,E).
- La cola Q contiene los vértices $\langle v_1, v_2, \ldots, v_r \rangle$ donde v_1 es el primer elemento y v_r el último elemento de Q. Entonces:

$$d[v_r] \leq d[v_1] + 1 \quad \text{y} \quad d[v_i] \leq d[v_{i+1}] \text{ para } i = 1, 2, \ldots, r-1.$$

- Prueba:
    - Usando inducción en el número de operaciones en la cola.
    - Inicialmente, cuando Q solo tiene a s, el lema se mantiene.
    - Para el paso inductivo, debemos probar que el lema se mantiene después de las operaciones ENQUEUE y DEQUEUE en un vértice.

Caminos más cortos: propiedades

- Si el primer elemento v_1 en Q sale, v_2 se convierte en la nueva cabeza.
- Por la hipótesis inductiva $d[v_1] \leq d[v_2]$, y tenemos que $d[v_r] \leq d[v_1]+1 \leq d[v_2]+1$
- Las demás desigualdades se mantienen y el lema se mantiene también con v_2 a la cabeza.
- Al poner un vértice v en Q, se convierte en v_{r+1}. En este momento hemos quitado al vértice u de Q y la nueva cabeza v_1 mantiene $d[v_1] \geq d[u]$.
- Entonces $d[v_{r+1}] = d[v] = d[u]+1 \leq d[v_1]+1$.
- Por la hipótesis inductiva también tenemos $d[v_r] \leq d[u]+1$ y por eso $d[v_r] \leq d[u]+1 = d[v] = d[v_r+1]$, y las demás desigualdades se mantienen.

Caminos más cortos: propiedades

- Corolario 1: Supongamos que los vértices v_i y v_j son puestos en Q durante la ejecución de BFS y que v_i fue puesto en la cola antes que v_j. Entonces:

  $$d[v_i] \leq d[v_j]$$

  al momento de poner a v_j en la cola.

- Prueba:
    - Por el Lema 3 y la propiedad de que cada vértice recibe un valor finito de d durante BFS.

Caminos más cortos

- Todo esto para probar que BFS encuentra correctamente caminos con las distancias más cortas.
- Teorema I (exactitud de BFS)
    - Sea una gráfica $G=(V,E)$ un grafo dirigido o no dirigido, supongamos que BFS se ejecuta en G para un vértice $s \in V$.
    - Durante la ejcución BFS descubre cada vértice $v \in V$ alcanzable desde la fuente s y a su terminación $d[v]=\delta(s,v)$ para todo $v \in V$.
    - Además, para cualquier vértice $v \neq s$ alcanzable desde s, uno de los caminos más cortos hasta v es un camino más corto desde s hasta $\pi[v]$ siguiendo la arista $(\pi[v],v)$.