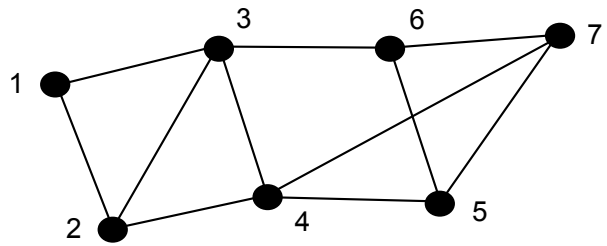
BFS/DFS - Práctica

Tecnología Digital V: Diseño de Algoritmos

Universidad Torcuato Di Tella

UNIVERSIDAD TORCUATO DI TELLA

Tipos de recorridos



- BFS(G,v): Si la lista L se implementa con una cola (queue), entonces el algoritmo recorre el grafo **a lo ancho** y se lo llama (*breadth-first search*).
  Recorre los vértices en orden de distancia creciente desde s.

- DFS(G,v): Si la lista L se implementa con una pila (stack), entonces el algoritmo recorre el grafo **en profundidad** y se lo llama (*depth-first search*).
  Encuentra primero el vértice más lejano a s.

BFS

Ejercicio 1: nodos derechos

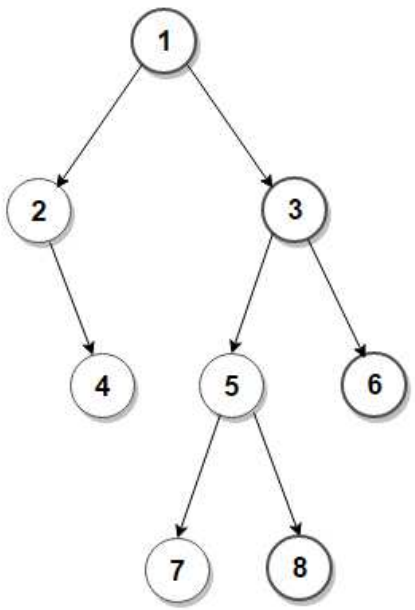
Enunciado

Dado un árbol binario escribir un algoritmo que retorne cada uno de los nodos que se encuentren a la derecha.

Ejercicio 1: nodos derechos

Enunciado

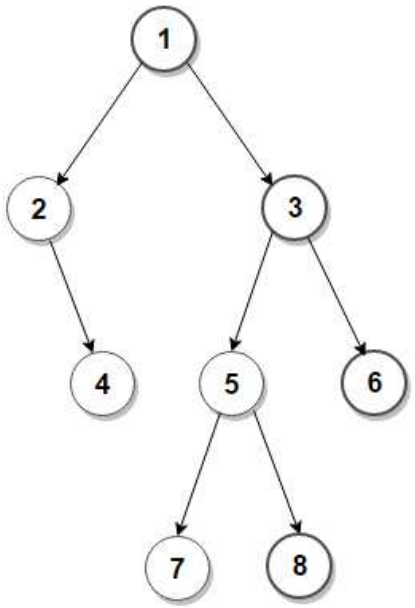
Dado un árbol binario escribir un algoritmo que retorne cada uno de los nodos que se encuentren a la derecha.



Ejercicio 1: nodos derechos

Enunciado

Dado un árbol binario escribir un algoritmo que retorne cada uno de los nodos que se encuentren a la derecha.



En este caso se debe retornar: [1, 3, 6, 8]

- Implementar con BFS.
- Ir recorriendo los nodos por niveles.
- Si el nodo actual es el último del nivel, agregarlo al resultado.

Ejercicio 1: implementación

```
void printRightView(Node* root) {
    if (root == nullptr)
        return;
    list<Node*> queue;
    queue.push_back(root);
    Node* curr = nullptr;
    while (!queue.empty())
        int size = queue.size();
        int i = 0;
        while (i++ < size)
            curr = queue.front();
            queue.pop_front();
            if (i == size)
                cout << curr->key << " ";
            if (curr->left)
                queue.push_back(curr->left);
            if (curr->right)
                queue.push_back(curr->right);
}
```

Ejercicio 2: visitar todos los nodos desde el nodo X

Enunciado

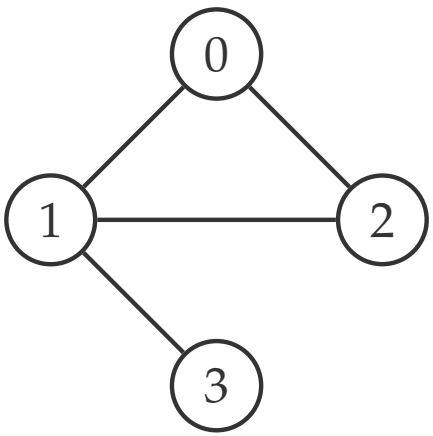
Dado un grafo G con N nodos enumerados de 0 a $N-1$ representado con un vector de vectores donde arr[i] representa todos los nodos que son vecinos de i-esimo nodo. Retornar si se pueden visitar todos los nodos desde X.

Ejercicio 2: visitar todos los nodos desde el nodo X

Ejemplo 1

Input: arr = [[1, 2], [0, 3, 2], [0, 1], [1]], $N = 4$, $X = 0$

El agoritmo devuelve true.



Ejemplo 2

Input: arr = [[1, 2], [0, 3, 2], [0, 1], [1]], $N = 5$, $X = 4$

El algoritmo devuelve false

Explicación: no existe ningún nodo conectado al nodo 4.

Ejercicio 2: visitar todos los nodos desde el nodo X

Algoritmo

La idea es usar BFS desde el nodo X y contar todos los nodos que se visitan en la ruta. Finalmente, se verifica si el número de nodos que se visitan es igual al número dado de nodos N o no.

- El arreglo funciona como una lista de adjacencia del grafo.
- Usar una cola para los nodos visitados.
- Tener un contandor de la cant. de nodos visitados durante el BFS.
- Mientra la cola no se encuentre vacia, sacar el último nodo y aumentar el contador en 1.
- Chequear si los hijos del nodo actual son visitados, si no, introducirlos en la cola y marcarlos como visitados.
- Por último, verificar si el contador llegó a N.

Ejercicio 2: implementación

```
bool canVisitAllNodes(arr[][], int X, int n) {
    queue<int> q;
    vector<int> visited(n, false);
    q.push(X);
    visited[X] = true;
    int count = 0;

    while (q.size() > 0) {
        int size = q.size();
        for (int i = 0; i < size; i++) {
            auto curr = q.front();
            q.pop();
            count++;
            for (auto j : arr[curr]) {
                if (visited[j] == false) {
                    q.push(j);
                    visited[j] = true;

    return count == n;
}
```

DFS

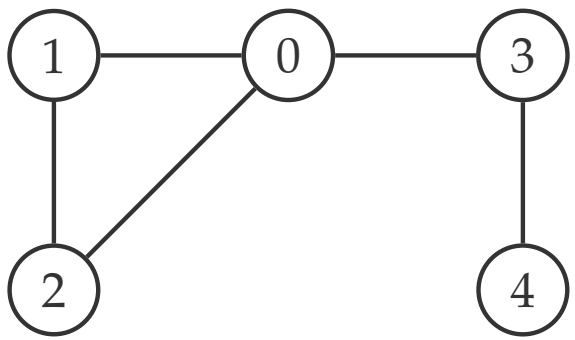
Ejercicio 1: cantidad de puentes

Puentes

Una arista en un grafo no dirigido conexo es un puente si al removerlo el grafo pasa a ser no conexo. Para grafos no conexos, un puente es una arista que al removerla incrementa la cantidad de componentes conexas.

Enunciado

Dado un grafo G no dirigido, encontrar los puentes.



En este caso se debe retornar: $[(0,3);(0,4)]$

- Implementar con DFS.
- Eliminar las aristas una por una para ver si se forma un grafo disconexo.