CMPT 280

Topic 21: Graph Traversals

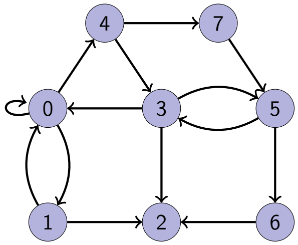
Mark G. Eramian

University of Saskatchewan

References

- Textbook, Chapter 21

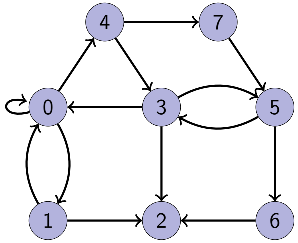
Exercise 1



```
// An algorithm for breadth first
// traversal of a graph
Algoirthm bft(s)
s is the start node in the graph

q = new Queue()

For each vertex v of V
    reached(v) = false

reached(s) = true
q.insert(s)

while not q.isEmpty() do
    w = q.item()     // get top node on stack
    q.deleteItem()   // pop the stack

    // perform the "visit" operation on w

    For each v adjacent to w do
        if not reached(v)
            reached(v) = true
            q.insert(v)
```

Find the breadth-first traversal of the graph if we assume an adjacency list representation where nodes happen to be in numerical order.

Exercise 2



```
// An algorithm for depth-first
// traversal of a graph.
Algorithm dft(s)
s is the start node.

// V is the set of nodes in the graph
For each vertex v in V
    reached(v) = false

dftHelper(s);

// Recursive helper method for algorithm dft()
Algoirthm dftHelper(v)
v is a graph node

reached(v) = true

// perform the visit operation for v

For each node u adjacent to v
    if not reached(u)
        dftHelper(u)
```

Find the depth-first traversal of the graph if we assume an adjacency list representation where nodes happen to be in numerical order.

Exercise 3

```
// An algorithm for breadth first
// traversal of a graph
Algoirthm bft(s)
s is the start node in the graph

q = new Queue()

For each vertex v of V
    reached(v) = false

reached(s) = true
q.insert(s)

while not q.isEmpty() do
    w = q.item()    // get top node on stack
    q.deleteItem()  // pop the stack

    // perform the "visit" operation on w

    For each v adjacent to w do
        if not reached(v)
            reached(v) = true
            q.insert(v)
```

Assuming adjacency list representation, what is the worst-case time complexity of BFT? What if we assume adjacency matrix instead?

Exercise 4

```
// An algorithm for depth-first
// traversal of a graph.
Algorithm dft(s)
s is the start node.

// V is the set of nodes in the graph
For each vertex v in V
    reached(v) = false

dftHelper(s);

// Recursive helper method for algorithm dft()
Algoirthm dftHelper(v)
v is a graph node

reached(v) = true

// perform the visit operation for v

For each node u adjacent to v
    if not reached(u)
        dftHelper(u)
```

Assuming adjacency list representation, what is the worst-case time complexity of DFT? What if we assume adjacency matrix instead?

Exercise 5

- In many applications, we associate additional data with a node besides its index.
- A specific type of graph traversal is a search for a particular node with a particular property, i.e. *breadth-first search* and *depth-first search*.
- A breadth-first or depth-first search is just a breadth-first or depth-first traversal, respectively, where we stop as soon as we find the node we are looking for and return it.
- How can we modify the breadth- and depth-first traversals to do this?

Exercise 6

- How can we further modify the depth-first search to return, instead of the sought node, the **entire path** from the start node to the sought node?
- What is the time complexity of our resulting algorithm?

Next Class

- Next class reading: Chapter 22: Shortest Path Algorithms for Graphs