Graphs and Network Flows
ISE 411

Lecture 6

Dr. Ted Ralphs

References for Today's Lecture

- Required reading
  - Section 18.8
- References
  - AMO Sections 3.4
  - CLRS Chapter 22

General Graph Search

- Depth-first search is so called because the node selected in each step is a neighbor of the node that is farthest from the root (in the tree).
- This is convenient because it allows a simple recursive implementation.
- Could we search the graph in a different "order"?

General Search Algorithm

This is what a more general search algorithm might look like.

Input: Graph $G = (N, A)$ and source node $s \in N$

$Q \leftarrow \{s\}$
while $Q \neq \emptyset$ **do**
    remove the "next" node v from Q
    mark v
    process v
    for $w \in A(v)$ **do**
        process (v, w)
        if w is not marked **then**
            $Q \leftarrow Q \cup \{w\}$
        end if
    end for
end while

(Figure 9.1 from Papadimitriou and Steiglitz)

Search Algorithm

- The search proceeds differently depending on which element v is selected from Q on line 3 in each iteration.
- Q must be ordered in some way by storing it in an appropriate data structure.
  - If Q is a *queue*, elements are inserted at one end and removed from the other and we get FIFO ordering.
  - If Q is a *stack*, elements are inserted and deleted from the the same end and we get LIFO ordering.
- The efficiency of the algorithm can be affected by
  - the data structure used to maintain Q,
  - what additional steps are required in processing v (line 5),
  - what additional steps are required in processing (v, w) (line 7).

General Graph Search Algorithms

```
def search(self, root, q = Stack()):
    if isinstance(q, Queue):
        addToQ = q.enqueue
        removeFromQ = q.dequeue
    elif isinstance(q, Stack):
        addToQ = q.push
        removeFromQ = q.pop
    visited = {}
    visited [root] = True
    addToQ(root)
    while not q.isEmpty():
        current = removeFromQ()
        for n in current.get_neighbors():
            if not n in visited:
                visited[n] = True
                addToQ(n)
```

Complexity of Search Algorithms

1. Initialization.
2. Maintaining the set Q.
   - What is the maximum number of additions and removals?
   - How many operations are required for each?
3. Searching the adjacency lists.
   - How many times do we touch each element of each list?
   - How much work do we do each time we touch an element?
4. Processing a node.

Algorithm Search

- The algorithm is a template for a whole class of algorithms.
  - If Q is a *stack* (LIFO), we are doing something like depth-first search, as before (but not precisely...).
  - If Q is a *queue* (FIFO), we are doing *breadth-first search*.
  - In other cases, we will want to maintain Q as a priority queue.
- What problem does breadth-first search of a graph solve?