Graph Traversals

CS 277
Brad Solomon

April 19, 2023
Exam 3 Signups Available

April 24 — April 27

Very limited window for makeup exams (since end of semester is near)

Covers content from week 10 — 14
Learning Objectives

Re-introduce breadth first search traversal in a graph

Explore use cases for BFS traversal

Introduce depth first search traversal in a graph
Graph Traversals

There is no clear order in a graph (even less than a tree!)

How can we systematically go through a complex graph in the fewest steps?

Tree traversals won’t work — let’s compare:

- Rooted
- Acyclic
- Clear base cases (‘doneness’)

- Arbitrary starting point
- Can have cycles
- Must track visited nodes directly
Simple BFS Traversal

1) Create a queue and a visit list
2) Initialize both to contain our start
3) While queue not empty:
   Remove front vertex of queue
   Check if each edge has been seen before
   Add unvisited edges to queue (and list)
Simple BFS Traversal

1) Create a queue and a visit list
2) Initialize both to contain our start
3) While queue not empty:
   - Remove front vertex of queue
   - Check if each edge has been seen before
   - Add unvisited edges to queue (and list)

What is my runtime?
Simple BFS Traversal

What is the shortest distance from \( A \) to \( H \)?

What is the shortest path from \( A \) to \( H \)?

What is the shortest path from \( A \) to \( F \)?

What is the shortest distance from \( A \) to \( F \)?
What data structure is this?
A **minimum spanning tree** is a tree formed by a subset of graph edges such that all vertices are connected with the smallest total possible edge weight.

On an unweighted, undirected graph this MST can be built by tracking **discovery** edges during a BFS traversal.

We call the remaining edges **cross** edges. What can I say about a graph with at least one **cross** edge?
If we modify our BFS traversal algorithm, we can track both distances and discovery edges!
Traversal: BFS

Replace ‘visited’ list with a **distance** and a **previous**.

When we add to queue, record **previous**.

When we process vertex from queue, record **distance**.
Traversal: BFS

Replace ‘visited’ list with a **distance** and a **previous**.

When we add to queue, record **previous**.

When we process vertex from queue, record **distance**.
Traversal: DFS
Traversal: DFS

1) Create a stack and a visit list
2) Initialize both to contain our start
3) While stack not empty:
   Remove top item from stack (temporarily)
   If first time seeing top vertex, process it
   If one or more adjacent edges unvisited, \textbf{add the item back to stack before…}
   Add \textbf{one} unseen adjacent vertex to stack
Traversal: DFS

Does distance have meaning here?

Do our edge labels have meaning here?

---

Discovery Edge

Back Edge
Traversal: DFS
DFS vs BFS

**DFS:**

**Pros:**

**Cons:**

**BFS:**

**Pros:**

**Cons:**
"The Muddy City" by CS Unplugged, Creative Commons BY-NC-SA 4.0