CS 61B
Spring 2017

Graph Algorithms

Discussion 12: April 12, 2017

1 Graph Algorithms

Traversal Visit all the nodes in the graph.

- Depth-first traversal (preorder and postorder)
- Level-order traversal

Search Given a start node, find a goal state.

- Depth-first search
- Iterative-deepening depth-first search
- Breadth-first search

***Single Pair* Shortest Path** Given a start node, find the shortest path to a goal node.

- *Uniform cost search*
- *Greedy search*
- A* search

***Single Source* Shortest Path** Given a start node, find the shortest paths to all other nodes.

- Dijkstra's algorithm

Minimum Spanning Tree A *spanning tree*, or cycle-free subset of edges connecting all the nodes, with the minimum possible total edge weight.

- Prim's algorithm
- Kruskal's algorithm

2 Minimum Spanning Trees

2.1 Give a description of how Prim's algorithm works.

2.2 Give a description of Kruskal's algorithm.

```
function GRAPH-SEARCH(start)
    seen ← an empty set
    fringe ← java.util.Queue interface
    ADD(start, fringe)
    while fringe is not empty do
        node ← REMOVE(fringe)
        if node is the goal then return node
        if node is not in seen then
            ADD(node, seen)
            for child in NEIGHBORS(node) do
                ADD(child, fringe)
    return failure
```

3 Dijkstra's Algorithm

3.1 What fringe do we use in Dijkstra's algorithm and what does it keep track of?

3.2 Assuming the runtime for `changePriority` is in $O(f(N))$ and `removeMin` is in $O(g(N))$ where N is the size of the priority queue, give the runtime of Dijkstra's algorithm on the simple graph $G = (V, E)$.

3.3 Give the runtime bound for Dijkstra's assuming the priority queue is implemented using an unordered array, ordered linked list, and a binary min-heap.

4 A* Search

4.1 What fringe do we use in A* search and what does it keep track of?

4.2 What is a *heuristic*? What is its correctness conditions?