**Graph Algorithms Reference Sheet**

**breadth-first-search()**
- make a queue of nodes.
- enqueue the start node.
- color the start node yellow.

**while (the queue is not empty) {**
  - dequeue a node from the queue.
  - color that node green.

  **for (each neighboring node) {**
    - if (that node is gray) {
      - color the node yellow.
      - enqueue it.
    }
  }

} **}

**dijkstra's-algorithm()**
- make a priority queue of nodes.
- enqueue the start node at distance $0$.
- color the start node yellow.

**while (the queue is not empty) {**
  - dequeue a node from the queue.
  - if (that node isn't green) {
    - color that node green.
  }

  **for (each neighboring node) {**
    - if (that node is not green) {
      - color the node yellow.
      - enqueue it at the new distance.
    }
  }

} **}

**aStarSearch()**
- make a priority queue of nodes.
- enqueue the start node at distance $0$.
- color the start node yellow.

**while (the queue is not empty) {**
  - dequeue a node from the queue.
  - if (that node isn't green) {
    - color that node green.
  }

  **for (each neighboring node) {**
    - if (that node is not green) {
      - color the node yellow.
      - enqueue it at the new distance plus the heuristic.
    }
  }

} **}

**kruskals-algorithm()**
- remove all edges from the graph.
- put each node into its own cluster.

**for (each edge, in increasing order of cost) {**
  - if (the edge’s endpoints are in different clusters) {
    - add that edge back to the graph.
    - merge those two clusters.
  }

} **
- return the edges added back.**