Finish up Problem from last class

Version 1: One robot start + goal in maze

Vertex for each grid square
Edge if no wall between two grid squares (undirected, unweighted)

Run BFS from start (source = start) and report best path to goal
Version 2: 2 robots each with start + goal – can never be adjacent or in same position.

**source**

\( (\text{robot 1}, \text{start}, \text{start}) \)

**vertex**: state of world as given by ordered pair of robot location

**goal**

\( (\text{robot 1}, \text{goal}, \text{goal}) \)

**edge**: edge from \( S_1 \) – \( S_2 \) when a single move by each robot goes between \( S_1 \) + \( S_2 \)
Greedy Tree Builder

- Change semantics associated with tag/priority for each vertex

- Change tag/priority given to source/seed

- decide if min or max priority vertex is the best one to pick next
<table>
<thead>
<tr>
<th>Tag semantics for V</th>
<th>Shortest path</th>
<th>MST</th>
<th>Max bottleneck</th>
</tr>
</thead>
<tbody>
<tr>
<td>length of shortest path found so far from s to v</td>
<td>min edge to connect v to partial T</td>
<td>value of max bottleneck path so far from s to v</td>
<td></td>
</tr>
<tr>
<td>tag for source/seed</td>
<td>0</td>
<td>0</td>
<td>$\infty$</td>
</tr>
<tr>
<td>Min or max</td>
<td>min</td>
<td>min</td>
<td>max</td>
</tr>
</tbody>
</table>
Greedy Tree Builder

Initially \( s \) is placed in \( T \). Then the following steps are repeated until all discovered vertices have been placed in \( T \).

1. Select the vertex \( u \in Q \) with the highest priority over all vertices in \( Q \). (For each algorithm, a proof that this greedy choice is part of an optimal solution is required to prove that the final solution is optimal.)

2. Remove \( u \) from \( Q \), which implicitly places \( u \) in \( T \). Since the cost for each vertex \( v \in Q \) represents its best connection to some vertex in \( T \), the addition of \( u \) to \( T \) provides a new possible connection for each vertex \( v \notin T \).

3. Consider all outgoing edges \( e = (u, v) \) from \( u \).
   
   (a) If \( v \in U \), then \( v \) is placed into \( Q \) after setting the edge from its parent to \( e \) and initializing \( v_{\text{cost}} \) to the cost associated for parent edge \( e \).

   (b) If \( v \in Q \), the cost associated with \( v \), for parent edge \( e \), is computed. If this cost \( c \) is better than \( v \)'s current cost, then the cost for \( v \) is set to \( c \) and its parent edge is set to \( e \).
Function to Consider an Edge

```java
void consider(E e, double parentCost, TaggedPriorityQueue<Double, V> pq) {
    double newCost = getCost(e, parentCost);
    if (newCost < loc.get().getTag()) {
        edgeFromParent = e;
        cost = newCost;
        pq.updateTag(cost, loc);
    }
}
```

Shortest path: parent Cost + e.weight
MST: e.weight
max bottleneck: min(parent Cost, e.weight)
void greedyTreeBuilder(tree, seedCost, comp)

Create a TaggedPriorityQueue<Double, V> pq that uses comp
add source/seed as root of tree with cost seedCost
source.loc = pq.putTracked(seedCost, source)

While (!pq.isEmpty())
    V u = pq.extractMax().getElement();

    for each outgoing edge e leaving u
        if (e.weight < 0) throw new NegativeWeightEdgeException
        V v = other endpoint of e (other than u)

    vData is an object holding data associated
    with v. For this algorithm
    if (vData.loc. in Collection(v))
        vData.consider(v, e, vData.getCost(e u's cost))
    else
        add v to tree with parent e + cost
        vData.loc = pq.putTracked(vData, cost, v)
Prim's Minimum Spanning Tree
Algorithm

Algorithm that results with greedy tree builder
Time Complexity of greedy tree builder

\[ O\left(n \cdot T_1(n) + n T_E(n) + m T_U(n)\right) \]

- Insertion cost in a priority queue or \( n \) elements
- Extract Max time
- Update cost (raise priority)

- \( O(1) \)
- \( O(\log n) \)
- \( O(1) \)
Knapsack holds 100 lbs

80, 50, 40
Prove correctness of Prim's MST

Step 1 Prove first edge selected is part of some MST

Use proof by contradiction
Claimed MST $T$ adding $e$ to $T$ must creat a cycle then remove other edge in cycle with $S$.
know part of optimal
new vertex

graph contraction