Quiz Review

Here are some topics you should review:

Search Overview

- Understand the main differences between different algorithms.
- Be able to identify which algorithms run faster / slower and how they use resources.
- Know which searches are complete, informed, and systematic.
- Understand which algorithms are optimal and in what ways.

Breadth-First Searches

- Understand the difference between different measurements of cost.
- Know in detail how each of the breadth-first family of algorithms operates.
- Be able to step through code for each of the breadth-first family algorithms.

Informed Searches

- Know how to estimate the distance to the goal.
- Be ready to examine the effect of weights (heuristic and terrain).
- Understand the difference in cost between different types of steps.
- Know how to calculate the actual cost from the source (given cost).
- Know what an admissible heuristic is and how it relates to the A* algorithm.

Waypoint Graphs

- Be able to identify the required components of a waypoint.
- Know how to examine a node to find its successors, cost, and other components.
- Understand how a waypoint graph works, how to build it, and how to traverse it.
- Be able to calculate the given cost of a node and the costs between nodes.
- **Know how to calculate the given cost of a node and the costs between nodes.**
- ***Understand how to calculate the given cost of a node and the costs between nodes.***
- *(Get the picture yet?)*

Depth-First & Trial-and-Error Searches

- Be sure you can identify the different depth-first searches.
- Know the benefits and weaknesses of each of the depth-first searches.
- Know the benefits and weaknesses of trial-and-error searches.

Goal-Oriented Action Planning

- Know how goal-oriented action planning searches through the state space of actions.
- Know how GOAP constructs plans and chooses between them.

Genetic Algorithms

- Are genetic algorithms optimal / complete / exhaustive? Why or why not?
- What are the main steps in genetic algorithms?
- When are genetic algorithms useful? When are they a poor choice?
- What is a chromosome? A gene? What is fitness?
- What is crossover? What is mutation? How do they work?
- What types of selection can be used, and what are the differences?

Search Review

Categories of Searches

There are numerous path finding search algorithms and each one has its own advantages and disadvantages. The most important criteria for evaluating a search algorithm are:

- ***CPU Time*** – How much processing time does the algorithm require?
- ***Memory Use*** – How much memory does the algorithm need?
- ***Optimality*** – How "good" is the solution the algorithm finds?
- ***Completeness*** – Does the algorithm guarantee a solution if one exists?

Path finding is a search problem where the actions are steps in the path to the goal. Two major categories of searches are ***uninformed*** searches and ***informed*** searches.

Uninformed Searches

An uninformed search proceeds without any knowledge of the problem except how to generate successor states. Common uninformed searches include brute force methods and random searches. They look through the search space without consideration for which regions of the search space are more likely to contain the goal.

Informed Searches

Informed searches use problem-specific knowledge to compute priorities (cost values) to decide which regions of the search space are more likely to contain the goal. To illustrate the difference between these two categories of search algorithms, consider the problem of getting from home to a meeting location you have not visited. If you did an uninformed search, you might consider *all possible streets* when trying to find a path from home to the meeting location, or you might go in a random direction. In this context, even those streets leading away from the goal (or that might otherwise seem irrelevant) are considered equally. Uninformed search may seem foolish, but if it is systematic and the search space is finite, it will eventually find the goal. Informed searches, on the other hand, use problem specific knowledge to guide the search towards the goal. In the case of path finding, the distance from the start and/or the estimate of the distance to the goal can be used to guide the process. It is crucial to note that a route moving toward the goal does not mean that the route is the best or even part of a solution. For example, the road may be closed or might lead you onto a highway that heads the wrong way and does not have an exit for miles.

Path finding can be done at different levels of abstraction. For example, instead of planning every step from start to goal, we could consider only major roads involved and then use trial-and-error techniques to follow the high-level plan. We could also plan the high-level route and when finished plan the low-level details.

Search Algorithms

Breadth-First Search (BFS)

Breadth first is one of the fundamental methods of traversing/searching a tree or a graph. It conducts a brute-force style search, checking nodes one ply (step) at a time.

Characteristics:

- Uninformed search
- CPU intensive
- Memory intensive
- Optimal in *steps*
- Complete
- Exhaustive

Greedy Search (GS)

Greedy Search is a derivative of BFS. Instead of ordering successors by steps, it considers them by their estimated distance from the goal.

Characteristics:

- Informed Search
- Not very CPU intensive
- Not very Memory intensive
- Not Optimal
- Not Complete
- Exhaustive

Uniform Cost Search (UCS, or Dijkstra)

Uniform Cost Search is a derivative of BFS. Instead of ordering successors by steps, it considers them by their cost from the start. Lowest cost nodes are considered first.

Characteristics:

- Uninformed search
- CPU intensive
- Memory intensive
- Optimal in *cost* or *distance*
- Complete
- Exhaustive

A* Search

A* is a derivate of both UCS and GS. Instead of ordering successors by cost from the start or estimated distance from the goal, it considers them by the sum of the two. If the heuristic is admissible (not overestimating), then A* is optimal.

Characteristics:

- Informed search
- Somewhat CPU intensive
- Somewhat Memory intensive
- Optimal in *cost* or *distance* if heuristic is *admissible*.
- Complete
- Exhaustive

Depth-First Search (DFS)

Depth-first search is another fundamental method of traversing/searching a tree or a graph. It searches recursively, traveling as deeply as possible down one branch before trying a different branch.

Characteristics:

- Uninformed Search
- CPU intensive
- Memory friendly. Only requires enough memory to store the call stack.
- Not optimal – returns the first solution it finds.
- Not complete - it may go in the wrong direction and never return.
- Exhaustive

Iteratively Deepening Depth-First Search (IDDFS)

IDDFS is a derivative of depth first algorithm. It begins with some cut-off depth (usually one) and a depth-first search is performed. If a solution is not found, the depth is increased and another search is performed. This process is repeated until a solution is found or the search space is exhausted.

Characteristics:

- Uninformed
- CPU intensive
- Memory friendly.
- Optimal (if depth is iterated one step at a time) in steps.
- Complete
- Exhaustive

Genetic Algorithms

Genetic Algorithms form a class of algorithms based on the evolutionary model. They are time-limited and can be memory friendly, but they are not optimal, complete, or exhaustive because of random elements and possibility of repeat solutions and genes.

Characteristics:

- Informed
- CPU time-limited
- Memory friendly
- Not Optimal
- Not Complete
- Not Exhaustive