Learning Guide Unit 3

Site:	University of the People	Printed by:	Ryohei Hayashi
Course:	CS 4408-01 Artificial Intelligence - AY2025-T3	Date:	Thursday, 30 January 2025, 3:43 PM
Book:	Learning Guide Unit 3		

Description

Learning Guide Unit 3

Table of contents

Overview
Introduction
Reading Assignment
Discussion Assignment
Learning Journal
Self-Quiz
Graded Quiz
Checklist

Overview

Unit 3: Problem Solving Through Search

Topics:

- State Spaces in search
- Uninformed search strategies
  - breadth-first
  - depth-first,
  - depth-first with iterative deepening
- Informed search strategies
  - Heuristic search
  - A* search
- Space and time efficiency of different search techniques
- Pruning the search space

Learning Objectives:

By the end of this Unit, you will be able to:

1. Analyze the difference between uninformed and informed search techniques
2. Examine how breadth-first and depth-first search algorithm work, and how depth limit can improve depth-first search.
3. Describe how A* search technique works and when to use this approach.

Tasks:

- Peer assess Unit 2 Programming Assignment
- Read the Learning Guide and Reading Assignments
- Participate in the Discussion Assignment (post, comment, and rate in the Discussion Forum)
- Complete and submit the Learning Journal
- Take the Self-Quiz
- Take the Graded Quiz

Introduction

In Unit 3, we will begin with a review of the concepts of search that were explored in CS 3304 Analysis of Algorithms. We have already established that the implementation of artificial intelligence is the agent, an autonomous decision making entity that can carry out actions within its environment.

The purpose of the agent can be thought of as a problem solver. If we consider the examples of prototypical agents offered in the textbook including autonomous delivery robots, diagnostic assistant agents, intelligent tutoring systems, or trading agents, we see in each case an entity that must solve a problem.

The delivery robot must determine how to deliver packages in the most efficient way while avoiding obstacles and navigating the environment that it operates within. The intelligent tutoring agent must solve the problem of determining the current level of knowledge of the student and formulating a plan to get them to the goal level of knowledge.

A great example of this kind of agent can be seen in the adaptive assessments offered by the Khan academy. The assessments have the goal or problem of increasing the knowledge of the student.

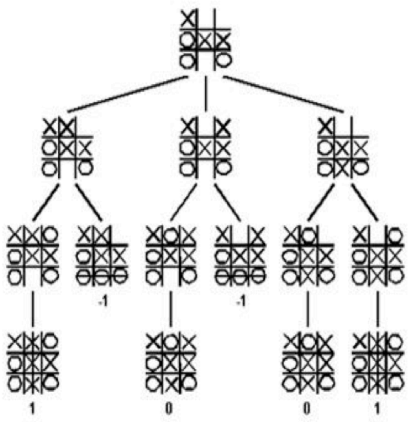
Such problem solving can be represented as a “search” for a solution. We have learned about the power of graphs within computer science and we have examined different approaches to searching graph space. We discovered that search can be optimized asymptotically through techniques such as the minimum spanning tree algorithms (Prim, Kruskal).

All of the various tree searching algorithms that we have explored in the past had in common that they were uninformed searches. The basic idea behind an uninformed search is that the graph or ‘tree’ is searched methodically without any knowledge of or regard for the location of the actual goal.

If we have some additional information beyond the set of nodes that make up the tree that would make our searching more efficient, we would refer to this as an informed search or what is typically called a Heuristic.

The idea of search as a way to solve problems may be a bit confusing so perhaps a simple example might help. Suppose that you wanted to develop a computer program that could play tic-tac-toe (how does a computer program that can play tic-tac-toe measure up when we apply the Turing test?).

For each move that the player made the program would need to determine the best counter move. The current playing board at any time in the game has 9 positions and each position can be blank, contain an x, or contain an o.   The combination of x’s, o’s, and blanks represent the state of the game.



The goal state would be one in which the computer program (let’s assume that the computer program is the ‘x’ player) can get three x’s in a row. This would be a goal of 1 or a win in the following diagram, a loss would be represented by a -1 and would occur when the player was able to get three o’s in a row and a draw would occur when all of the spots were filled and neither player was able to get three in a row.

Looking at the diagram above you can see that the graph would build by mapping each move that was available to the computer program. Each of these moves would branch out to the moves available to the player and then from these the moves available to the computer program and so on until there is either a win, lose, or draw state. The program would then search the tree for the first option that resulted in a goal state of win. When this goal state is found the program will stop searching as it has found a solution to the goal. If a win state cannot be found then the program would search the tree for a draw state and if a draw state cannot be found the program would concede the game.

What we see in this example is how we can map the various options available to an agent as a tree and then search the tree to find a solution to the problem. In this example, the problem was to find a winning play in tic-tac-toe. We can apply this same kind of approach to many problems.

In our simple example, we only have a few paths or choices available, however, for other problems, there may be many paths that are a part of large complex trees. Some trees may have an infinite number of options or they may have cycles. If you recall an acyclic tree was a tree that had no cycles and a cyclic tree was one that did have cycles. A cycle, if you remember, was a directed tree (or graph) where the arcs or edges that connected the nodes of the tree circled back upon themselves. Obviously when a simple search algorithm this could cause the algorithm to get caught in a loop and never find a solution to the problem.

To address situations such as cycles or trees with an infinite number of nodes there are a number of optimization techniques that can detect cycles or 'prune' branches of the tree that are not promising paths in which to find a solution. All of these techniques can translate a simple brute force approach of exhaustive search into a search that is much more asymptotically efficient.

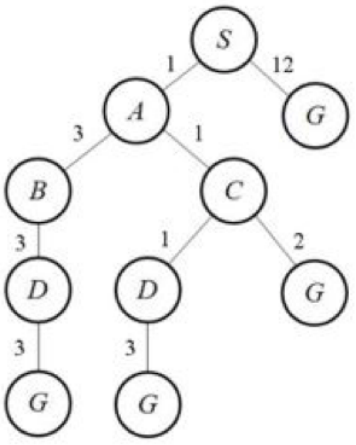
Lowest-cost-first search

The basic idea of a Heuristic search is that you use information that you have about the space (tree or graph) that you are searching to improve the efficiency of the search. A Heuristic might be employed, for example, to decide how nodes are added to a tree such that the nodes whose eventual paths may be the best locations to find the goal quickly.

In our reading, we learn about the 'frontier'. Assume that you are building the space to be searched one node at a time. The frontier represents the next node to be added and searched. In a depth-first search, nodes are added down paths and in a breadth-first search the nodes are added across all of the potential paths leading out of known nodes. One can think of the different search strategies (breadth-first, depth-first, etc.) as the approach used to add nodes to the frontier. The frontier itself can be thought of as a queue or stack where nodes are placed and then processed.

In the lowest-cost-first search, nodes are placed on the frontier based upon their cost. The cost is basically some value that is assigned to each path in the graph. These costs may be many things. In the classic Traveling salesman problem, a salesperson is trying to determine the most efficient route that would minimize the distance that he needs to travel between cities in his territory. In this case, the 'cost' was distance in miles or kilometers between the cities. If we consider the delivery robot example, the cost might be the time it takes to make various deliveries from the current location. In any case, the cost is some resource that the agent is trying to use in the most efficient manner possible.

Consider the following graph. If the goal were represented by nodes G and the current state was node S, then the least cost search would identify the cost of each path and the frontier would be implemented as a priority queue such that the path with the lowest cost would appear first.



The following are all of the paths that can be searched from the preceding graph with their costs to the right.

S,A,C,G	4
S,A,C,D,G	6
S,A,B,D,G	10
S,G	12

The path proceeding from node S through A, C, and finally G is the first path as it has the lowest cost. As learned in CS 3304 Analysis of Algorithms, we might use other names to describe this process such as "minimum-spanning-tree". If you recall both Prim's and Kruskal's algorithms were designed to identify the minimum spanning tree or to be able to identify the paths within the tree (graph) that had the shortest length in terms of cost.

Our textbook describes a simple lowest path approach that begins with a start node (S in our example above) and adds the paths into a priority queue that has the path with the lowest cost first. Which, after the first step, would be as follows:

{A1,G12} Where we have a path containing {S}
Of course, the A1 path is lowest cost and will be chosen (taken off of the frontier) which results in two more paths B and C being added.

{C1,B3,G12} Where we have a path containing {S,A}
The lowest cost option is now C so this is selected resulting in D and G being added as follows.

{D1,G2,B3,G12} Where we have a path containing {S,A,C}
In this case, the D would be selected and resulting in the following:

{G2,G3,B3,G12} Where we have a path containing {S,A,C,D}
Of course, this is a bit confusing as we have several G nodes representing the goal, but it should be clear that the best option would be to backtrack up the path to select the node G with a cost of 2, which would result in the following path.

{S,A,C,G}

You should recognize three things. First, this approach could be referred to as a 'greedy' algorithm because it is always selecting the best option that is presented at the moment. Second, this process produces the path to the goal that has the least cost and finally the process can be asymptotically inefficient as you may be required to search many paths and backtrack before finding the best goal.   The question is whether this backtracking can be minimized and that is the role of best-first.

Best-first Search

The best first search uses a Heuristic to estimate (guess) at the cost of a particular path. The basic idea of a heuristic is that you use information that is readily available about the nodes to compute a cost.   One approach to a heuristic (that is used in best-first search) is to estimate the cost of the path.   Although there are many ways to implement heuristics, the following is a simple example.

Let's assume that all of the nodes on our graph are locations and the path defined the route that has to be taken to get to the location. For simplicity let's assume that the nodes are cities and the path is the route that must be taken to get to the city from the starting node (starting city).

Assume that your nodes were cities including San Diego, Santa Barbara, San Bernardino, Los Angeles, Pasadena, Sacramento, San Jose, and San Francisco. The highways are the paths and the distances on the highways are the costs. Now if you assume that your start node is San Diego and your goal is to get to Sacramento, there are a number of different routes to get there through a number of different nodes (cities).

Of course, if you haven't traveled these routes before you might not know what the best route to take is.   Now let's assume that you have the GPS coordinate for each of these cities. It would be very simple to compute the straight line distance between any two cities. For example, assume that you are in San Diego and your goal is to get to Sacramento. You could take highway 15 to San Bernardino, or highway 5 to Los Angeles, or highway 101 to Santa Barbara but you don't know which route is best to reach your goal. The best-first search could employ a simple heuristic. Assume that we could easily determine the straight line (as the bird flies) distance between your goal, Sacramento, and each of the three options San Bernardino, Los Angeles, and Santa Barbara, and then select the route with the shortest distance to Sacramento.

We would be informing our choice of a path with a heuristic. Of course, the heuristic could result in a path that was actually much longer because the highways leading to Sacramento from Santa Barbara might take longer or might have many more stops so this might not be the best option.

OREGON
NEVADA
MEXICO
MOJAVE DESERT
SIERRA NEVADA
COAST RANGES
MILES
0 25 50 75 100 125 150
Albers equal area projection
Sacramento
San Francisco
San Jose
Fresno
Bakersfield
Los Angeles
San Bernardino
Pasadena
Santa Barbara
San Diego
POPULATED PLACES
1,000,000 and over ● Los Angeles
500,000 – 999,999 ● San Francisco
100,000 – 499,999 • Berkeley
25,000 – 99,999 • Hollister
24,999 and less • Ukiah
State capital ★ Sacramento
Urban areas
TRANSPORTATION
Interstate; limited access highway
Other principal highway
Railroad
Ferry
PHYSICAL FEATURES
Streams: perennial; intermittent
Lakes: perennial; intermittent
Highest elevation in state (feet) +14494
Other elevations (feet) +10457

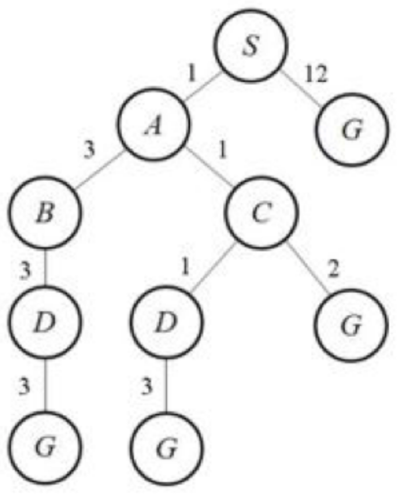
A* Search

In the lowest-cost-first search we see an approach that can guarantee the selection of the most optimal solution, if it finds a solution, however, the search is potentially asymptotically expensive. In the best-first approach, we see an example of how we can employ a heuristic to use readily available information to inform our search, which has the potential of efficiently finding a solution. Both approaches are not optimal and both have significant shortcomings.

What A* Search does is incorporate both searches into a single search. It does this by defining a cost function (let's call it ƒ) that combines the lowest cost first (we will refer to this as cost(p)) of a node with the heuristic (we will refer to this as h(p)) to yield the heuristic function:

$f(p) = \text{cost}(p) + h(p)$

Bringing back our graph example from before.



The heuristic function computes an estimated distance from every node to the goal. Lets' assume that the following are the heuristic values for each node (Again the particular heuristic function that will be used will vary from one problem to the next and will be defined by the kind of information that is readily available)

h(a) = 6
h(b) = 4.5
h(c) = 3
h(d) = 3
h(g) = 1.5

We also know the cost of each path. From S, the cost of the path, if we select node A, would be:

cost(A) = 1
h(A) = 6
ƒ(p) = cost(p) + h(p) = 7

From S, the cost of the path, if we select node G, would be:

cost(G) = 12
h(G) = 1.5
ƒ(p) = cost(p) + h(p) = 13.5

In this case, our A* would inform us to select node A and add B and C to the frontier. The search would continue in this way until the goal is found. As you can see the A* search factors in both the learning of the heuristic with the efficiency of the best first algorithm.

References

Kruska, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society. 7*(1), 48-50.

Prim, R. C. (1957). Shortest connection networks and some generalizations. *Bell System Technical Journal, 36*(6), 1389-1401.

Reading Assignment

Poole, D. L., & Mackworth, A. K. (2017). *Artificial Intelligence: Foundations of computational agents*. Cambridge University Press. https://artint.info/2e/html/ArtInt2e.html

Read the following chapters:

- Chapter 3 – Searching for Solutions

Video Resources

Taipala, D. (2014, September 22). *CS4408 Artificial Intelligence unit 3 lecture 1* [Video]. YouTube.

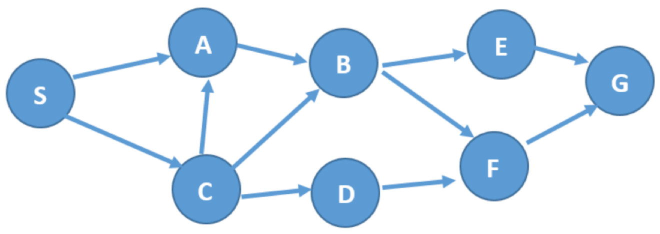
Taipala, D. (2014, September 22). *CS4408 Artificial Intelligence unit 3 lecture 2* [Video]. YouTube.

Discussion Assignment

In this unit, you have learned about Depth-first search (DFS), Breadth-first search (BFS) Consider the following directed graph and perform DFS and BFS where S is the starting node and G is the goal node. For both search strategies, you must show the search tree diagram and the final path to reach the goal node (G).

Then, describe in your own words, which search strategy is a better choice for this given graph problem. Your comparison should mention about time and space/memory complexity.

Finally, in the field of AI, where the optimal solution is highly expected, what are the factors you should consider before choosing the right search algorithm?



You must show your work step by step following the A* algorithm. Feel free to upload any hand diagram, table, stack, or queue you have created for this solution.

Your Discussion should be at least 250 words in length, but not more than 750 words. Use APA citations and references for the textbook and any other sources used.

Learning Journal

Assignment instructions

This assignment will assess your proficiency with depth-first search, breadth-first search, the A* algorithm, and your comprehension of how depth limit could improve depth-first search.

You can revise the following resources on search techniques:

- Poole, D. L., & Mackworth, A. K. (2017). Artificial Intelligence: Foundations of computational agents. Cambridge University Press. https://artint.info/2e/html/ArtInt2e.html
- Taipala, D. (2014, September 22). CS4408 Artificial Intelligence unit 3 lecture 1 [Video]. YouTube. https://cdnapisec.kaltura.com/index.php/extwidget/preview/partner_id/1934481/uiconf_id/45150521/entry_id/1_074wgkzt/embed/dyna
- Taipala, D. (2014, September 22). CS4408 Artificial Intelligence unit 3 lecture 2 [Video]. YouTube. https://cdnapisec.kaltura.com/index.php/extwidget/preview/partner_id/1934481/uiconf_id/45150521/entry_id/1_96qaahzc/embed/dyna

There are two parts to this assignment, and you must submit both.

Part 1

You are an employee of a tour and travel company. One of the clients is planning for a world tour and seeks your help for determining the best route for his travel. Create a report to give to your client and include answers to both questions A and B.

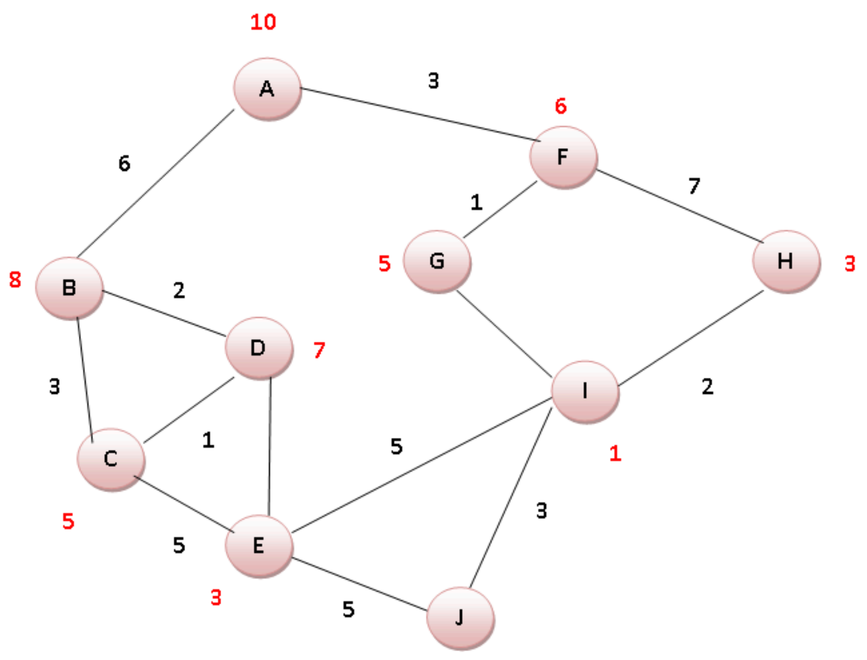
A. Create a list of various factors you will use for selecting the most efficient searching techniques for determining the best path for your client. Be sure to justify your selection by providing a detailed explanation. (ULO 3.1)

B. Think about a situation where another technique different from the one chosen by you in part A above, works more efficiently. Discuss situation and the proposed search technique by covering the characteristics that make this technique it more suitable for the situation. (ULO 3.1)

Part 2

Using A* algorithm, find the most cost-effective path to reach from start state "A" to final state "J" for the graph below. Be sure to provide calculations for all nodes traversed in the path. (ULO 3.3)

- The numbers written on edges represent the distance between the nodes.
- The numbers written on nodes represent the heuristic value.



Submission Instructions

- Submit both Part 1 and Part 2 in one document.
- Your report must be within 500 to 1000 words, not including references. Make sure your submission is double-spaced, Times New Roman, 12-point font, and 1" margins.
- Edit for spelling and grammar errors.

- Use sources to support your arguments. Use high-quality, credible, relevant sources to develop ideas that are appropriate for the discipline and genre of the writing.
- Use APA citations and references to support your work. For assistance with APA formatting, view the Learning Resource Center: Academic Writing.
- Importantly, this is not a research paper. A good essay will make original arguments using tables, diagrams, and sensible analysis. It will not just string together quotes from the "possible readings" (discussed below) or other sources.
- Your submission should be clearly written, concise, well organized, and free of spelling and grammar errors. The grading will be based on both the quality of your analysis and the quality of your writing.

This assignment will be assessed by your instructor using the rubric available on the assignment page located on the course homepage.

Self-Quiz

The Self-Quiz gives you an opportunity to self-assess your knowledge of what you have learned so far.

The results of the Self-Quiz do not count towards your final grade, but the quiz is an important part of the University's learning process and it is expected that you will take it to ensure understanding of the materials presented. Reviewing and analyzing your results will help you perform better on future Graded Quizzes and the Final Exam.

Please access the Self-Quiz on the main course homepage; it will be listed inside the Unit**.**

Graded Quiz

The Graded Quiz will test your knowledge of all the materials learned thus far. The results of the quiz will count towards your final grade.

Please access the Graded Quiz on the main course homepage; it will be listed inside the Unit. After you click on it, the quiz’s introduction will inform you of any time or attempt limits in place.

Good luck!

Checklist

Peer assess Unit 2 Development Assignment

Read the Learning Guide and Reading Assignments

Participate in the Discussion Assignment (post, comment, and rate in the Discussion Forum)

Complete an submit the Learning Journal assignment

Take the Self-Quiz

Take the Graded Quiz