## Homework 1 (10% of total course weight) –Search algorithms

**California State University San Bernardino, School of Computer Science and Engineering (CSE)**

**Date of Issue**: September 13, 2023, **Date of submission**: **September 29, 2023 – 11:59 pm (PST)**

**Module**: CSE 5120 Introduction to Artificial Intelligence

**Assessment brief:** This homework activity is based on our search algorithms that we covered in Week 2 (uninformed and informed search algorithms). Using a real-world problem of traveling from your start city (start – San Bernardino) to the destination (goal – Los Angeles) with the least cost incurred, you will be asked to implement various algorithms in Python to provide the solutions as outputs of your Python scripts. Your solution to each algorithm should return as output: (i) the selected path (i.e., the sequence of nodes (cities) that each algorithm will travel to reach the destination), and (ii) expanded nodes (or the visited nodes from your code).

The code for this project consists of a Python file (*homework\_1\_search.py*), which you will need to read and understand in order to complete the functions. You can download the code file from homework 1 link given on Canvas.

Your homework is based on two parts as given below:

1. Code implemented for search algorithms in given *homework\_1\_search.py* file (in specific sections as indicated in detail below)
2. A brief report on what you did for each algorithm (i.e., how you implemented with screenshots from the code editor as well as the outputs generated by your algorithms).

|  |  |
| --- | --- |
| **File Name** | **Description** |
| homework\_1\_search.py | Where all of your search algorithms will reside. |

After downloading the above file and changing to the directory where you place this file, you should be able to execute the script by running the following command.

python homework\_1\_search.py

.Which should produce a single line output for each of your algorithm since they are not implemented. Your output from the above script will first look like the following:

**Breadth First Search Result**

{'Returned solution: [], Expanded cities: []'}

**Depth First Search Result**

{'Returned solution: [], Expanded cities: []'}

**Uniform Cost Search Result**

{'Returned solution: [], Expanded cities: []'}

**A\* Search Result**

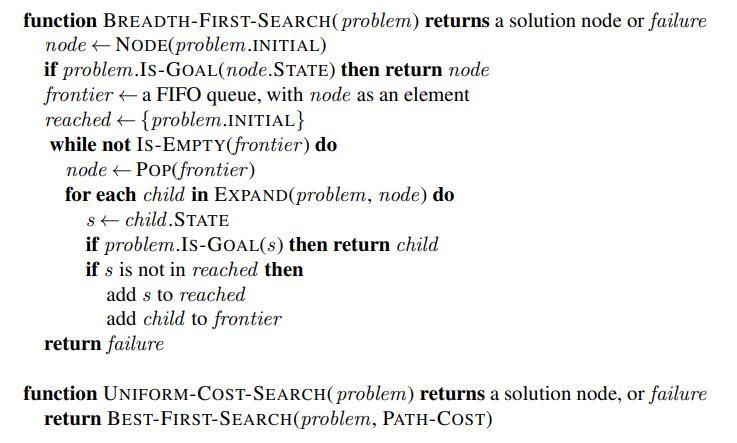
{'Returned solution: [], Expanded cities: []'}

You can use Spyder (installed through Anaconda from week 1 - 2 lecture) or other IDE for this work.

**Files to Edit and Submit:** You will need to edit and submit (homework\_1\_search.py) file to implement your algorithms. Once you have completed the homework, you are welcome to run them using the above command and observing the output for each of your algorithm before you submit your code for evaluation.

**Academic Dishonesty:** Your code will be checked against other submissions in the class for logical redundancy. If you copy someone else’s code and submit it with minor changes, they will be detected easily, so please do not try that and submit your own work only. In case of cheating, the University’s academic policies on cheating and dishonesty will strictly apply which may result from the deduction in your grade to expulsion.

**Getting Help:** If you are having difficulty in implementing the algorithms from the pseudocodes provided in this homework, contact the course staff for help. Office hours and Slack are there for your support. If you are not able to attend office hours, then please inform your instructor to arrange for additional time. The intent is to make these projects rewarding and instructional, not frustrating and demoralizing. You can either complete this homework on your own or discuss the problem and collaborate with another member of the class (or different section). Please clearly acknowledge and mention your group member in your homework report submission who you will collaborate with in this homework. Your report and program (homework\_1\_search.py file) will be separately submitted by yourself on Canvas irrespective of your collaboration with your group member. Group discussions are encouraged but copying of programs is NOT recommended. Programming based on your own skills is encouraged.



**Figure 1:** Breadth First and Uniform Cost search algorithms - pseudocode



**Figure 2:** Tree Search algorithm pseudocode

**Tasks for homework 1**

1. **Understanding the code base (not graded)**

In homework\_1\_search.py, you will find a graph data structure, which plans out a path through your problem and then executes that path step-by-step. The search algorithms for formulating a plan are not implemented: your task is to implement them.

First, test that the homework\_1\_search.py is working correctly by running the following command.

python homework\_1\_search.py

The command above prints empty solutions for each algorithm. Now you will need to implement different search algorithms to plan your routes and reach your destination. Remember that a search node must contain not only a state but also the information necessary to reconstruct the path (plan) which gets to that state from the start state.

**Important note:** All of your search functions need to return (i) a list of actions (sequence of cities) that will lead you from the start to the goal, and (ii) a list of expanded nodes. You can ignore the order of the expanded nodes in which they are displayed in your output.

**Important note:** Make sure to **use** the Stack, Queue and PriorityQueue data structures recommended to you in homework\_1\_search.py! These data structure implementations have particular properties that are required for compatibility with the accuracy of your algorithms.

**Hint:** The algorithms we covered so far are quite similar. DFS, BFS, UCS, and A\* algorithms differ only in the details of how the fringe (or *frontier*) is managed. So, concentrate on getting DFS right and the rest should be relatively straightforward. Indeed, one possible implementation requires only a single generic search method which is configured with an algorithm-specific queuing strategy. (Your implementation need not be of this form to receive full credit.)

1. **Depth First Search (2%)**

Implement the depth-first search (DFS) algorithm in the depthFirstSearch function in homework\_1\_search.py.

**Evaluation**: Run the following command to test your solution: python homework\_1\_search.py. Printing correct results in the form of the returned path and expanded nodes will account for **2%**. Failure to print the correct result will deduct your grade.

1. **Breadth First Search (2%)**

Implement the breadth-first search (BFS) algorithm in the breadthFirstSearch function in homework\_1\_search.py.

**Evaluation**: Run the following command to test your solution: python homework\_1\_search.py. Printing correct results in the form of the returned path and expanded nodes will account for **2%**. Failure to print the correct result will deduct your grade.

### **Uniform Cost Search (3%)**

BFS tries to minimize the number of actions taken, but not necessarily the least-cost path. By varying the cost function (weights data structure), you can explore different paths. By changing the cost function (weights), we can find different paths. For example, we can charge more for going through cities that are heavy in traffic or cover longer distance and vice versa.

Implement the uniform-cost search (UCS) algorithm in the uniformCostSearch function in homework\_1\_search.py.

**Evaluation**: Run the following command to test your solution: python homework\_1\_search.py. Printing correct results in the form of the returned path and expanded nodes will account for **3%**. Failure to print the correct result will deduct your grade.

1. **A\* Search (3%)**

Implement the A\* search algorithm in the AStar function in homework\_1\_search.py. A\* takes a heuristic function as an argument.

You need to test your A\* implementation on the problem of finding a path from San Bernardino to Los Angeles. Your solution might return the same path as that of one or more of the above algorithms which is OK.

**Evaluation**: Run the following command to test your solution: python homework\_1\_search.py. Printing correct results in the form of the returned path and expanded nodes will account for **3%**. Failure to print the correct result will deduct your grade.

**Homework 1** (10%)

CSE 5120 (Section 01) – Introduction to Artificial Intelligence – Fall 2023

*Submitted to*

Department of Computer Science and Engineering  
California State University, San Bernardino, California

*by*

Student name (CSUSB ID)

(Your collaborator in this homework (if any))

Date: *Month Day, Year*

*Email:*

* *Your email*
* *Your collaborator’s email (if you collaborated with any)*

**Report**

Brief description of your work here acknowledging your collaboration with your class fellow and the capacity at which he/she collaborated with you, followed by the algorithms you implemented.

1. **Depth First Search**

Your brief explanation (e.g., does DFS expand the shallowest or deepest unexpanded node? did you use Stack, Queue, or PriorityQueue in your code?) with screenshots of your code Evaluation (results from homework\_1\_search.py)

1. **Breadth First Search**

Your brief explanation (e.g., does BFS expand the shallowest or deepest unexpanded node? did you use Stack, Queue, or PriorityQueue in your code?) with screenshots of your code Evaluation (results from homework\_1\_search.py)

1. **Uniform Cost Search**

Your brief explanation (e.g., does UCS expand the cheapest or closest node to the goal state? What function did you use to expand the cheapest or closest node in this algorithm and at which line?) with screenshots of your code Evaluation (results from homework\_1\_search.py)

1. **A\* Search**

Your brief explanation (e.g., does A\* use g(n) or h(n)? Where in the code are you retrieving the cost of an unexpanded node to plan and which function did you implement/use to get g(n), h(n), f(n) etc?) with screenshots of your code Evaluation (results from homework\_1\_search.py)