In the previous assignment, you built a basic agent that navigated around a grid based on reflexes. One version of this agent could remember where it had been, but neither version could plan ahead. In this assignment, you will build an agent that is capable of using AI search techniques in order to plan ahead.

There are three parts to this assignment. Part one requires you to implement search techniques using the simulator you built in the previous assignment, and part two requires you to implement them in a different domain. You may choose to integrate the second part into your project (see the project description for details). Part three requires you to compare the two implementations.

Part 1: The GridWorld Robot Domain

In this assignment you will implement breadth-first search and $A^*$ using search agents in the simulator that you built for the first programming assignment. You will compare these algorithms for solving problems in the robot domain and the other. You will also compare the performance of your basic agent (from the previous assignment) to the performance of the search algorithms in the robot domain.

The “GridWorld” robot domain will be similar to what you implemented for the previous assignment. In this assignment, GridWorld will have the following characteristics:

- The robot’s world will be a $25 \times 25$ grid (i.e., a larger version of the $5 \times 5$ grid that we have used in class)
- The robot can move only to adjacent squares by going up, down, left or right. The robot cannot move diagonally. It cannot move off the grid or into a location where an obstacle resides. The cost of each move is 1.
- There are obstacles in the world. The position of these obstacles will remain constant while the robot is planning a path between its initial and goal locations. However, the obstacles should be in different locations for different runs.

Part 2: Another Domain

Choose a domain in which to implement searches to compare with your GridWorld implementations. You can either choose to integrate this into your project or to pick another domain.

Integration with Your Project

If you choose to do a search assignment related to your project, you need to see me before starting in order to define what it is that you should be doing.

In most cases, you will do a breadth-first search and $A^*$. However, in some cases, these searches will not be the best for your project. In these cases, you will pick two searches that are reasonable for your project and compare them against the same searches implemented in GridWorld; in these cases, you will not have to implement breadth-first search or $A^*$ in GridWorld. If it turns out that you do different searches, then these are the requirements:

- Any searches done must be of comparable difficulty and address the same issues as if you had done breadth-first search and $A^*$.
- You will need to try two alternative forms of search and compare the results.
• At least one of your searches must be a heuristic search; you will need to do tests in both
domains with two different heuristics and compare them. (In some cases, your search won’t
be applicable to the robot world. We’ll straighten this out when you talk to me.)

Not Integrating with Your Project

If you choose not to integrate this assignment with your project, you will choose the second test domain
from traditional AI search problems:
• missionaries and cannibals
• towers of Hanoi
• cryptarithmetic
• traveling salesman
• 8-puzzle
• 8 queens
• water jug problem
• monkeys and bananas
• blocks world problems

You should pick a problem that will help you to understand more about breadth-first search and A*.
You will be asked to justify your selection in the write-up for this program. You will also need to vary the
problem for test runs, so you should choose a domain where the initial and goal states can be easily varied
to create different problems.

Part III: Implementation and Comparison Runs

To compare the searches in two domains, you will:

1. Write breadth-first search and A* so that they are domain-independent.
   (a) Write heuristic functions for each domain to evaluate the successors in A*.
   (b) Isolate domain-dependent aspects of the algorithms (such as finding successors
       for nodes) in separate functions. To change domains, only these functions will
       need to be changed.
   (c) Keep track of the number of nodes created during the search.

2. Write a function to generate the test problems. The function should choose random initial and
goal states. It should also change the world, when the world affects the problem. For example,
for the robot domain, the initial state, goal state, and number and location of the obstacles
should all be randomly-generated.

3. Run both breadth-first search and A* on 20 randomly-generated problems from each domain.
   Note: you’ll probably want to automate this process using what is often called a “test harness.”
   Basically, you’ll want to write a Lisp program that loops for twenty times, each time generating
   a sample problem, running your program on the sample, and collecting the data (e.g., writing
   it to a file). Data to collect should be the number of nodes generated and the total run time
   for each problem, either CPU time (best) or clock time (not so good).

4. Run A* again on the previously-generated problems. This time change the heuristic function
   for each domain. (So, if you’re using a test harness, you’ll want to run BFS, A*, and the A*
   with the new heuristic each time through the loop.)

In addition, you will generate 20 random robot worlds in which a goal is located in a corner, with obstacles
located randomly (just not on the goal location, of course). Run your basic agent program, breadth-first
search, and A* on each world, collecting data on the number of nodes generated (and run time, if desired).

To get full credit for the programming portion of this assignment, the program (searches and simulation)
must run, must be well-documented, and must be well-structured.
Write-up

Hand in a write-up which includes:

- A discussion of your test domains:
  - What operators are available?
  - What is different in different test runs?
  - What heuristic functions were used?
  - Why did you choose the second domain?
- An example run of each kind of agent (breadth-first, A*).
- A discussion of the results:
  - Compare the number of nodes created for breadth-first search and for A*. You will want to look at both the average number of nodes created for all 20 runs and the number created for each individual run. You will need to do statistical analysis of the results; I’d suggest something like a paired t-test, which Excel should be able to do for you. You should report your results along with a the confidence you have in them, expressed as a p-value. For example, “BFS produced more nodes than A* (p<.05),” which means, roughly, that you’re 95% certain of the conclusion. Do this for run times, too. How can you explain the results?
  - Compare the number of nodes created and the run times for A* when different heuristics are used. You will want to look at both the average number of nodes created for all 20 runs, and you may need to look at the number created for each individual run. Compare run times as well. Again, use statistical analysis to bound the confidence in your conclusion. How can you explain the results?
  - Are there other variables that you would like to compare? If so, what are they and how would you run tests to compare them?
- What have you learned, or demonstrated, in these experiments that might be useful when implementing AI systems in the real world.

In addition, include a separate section comparing the basic agent to the breadth-first search and A* runs in the robot world.

A good organization for your write-up is something like:
1. Introduction/Background – what are you trying to do? Etc.
2. Methods – include here any discussion of your statistical techniques, program, etc., that will be useful. You must tell me the statistical basis for your conclusions.
3. Results – describe the data obtained.
4. Discussion – conclusions, etc.

That is, the general format for a scientific paper.

If, after consulting with me, you did not implement breadth-first search and A*, then your write-up will be similar to the following and will include:

- A description of your search techniques and why they were chosen.
- A discussion of your test domains:
  - What operators are available?
  - What is different in different test runs?
  - What heuristic functions were used?
- A discussion of the results, as above.
- What have you learned, or demonstrated, in these experiments that might be useful when implementing AI systems in the real world.
- How this ties to your project. Be specific and thoughtful in your answer. Will this be useful in the actual project? I.e., do you expect to include it in the final version of your project?

---

1 If it can’t, and you don’t have access to any other statistical software, then see me – we have a copy of a Lisp-based statistics package locally.
2 Really that there is less than 1 chance in 20 that the results you obtained are due to chance.
To turn in:

Turn in the following in *electronic* form:

- Your well-documented code
- Write-up as described above; your analysis should be thoughtful and thorough in your write-up. Your write-up should be in PDF, if at all possible. If not, then Word (.doc, .rtf) format or even plain text is okay.

Grading:

Your grade will be divided evenly between your program and your write-up. Both sections will be graded on form as well as content. Your program should be well-documented and well-structured. Your write-up should be well-written and should not contain spelling, grammar, or typographical errors.