CSC 520-001/601 # Introduction to Artificial Intelligence **Fall 2015**

[Back to Moodle] [Back to Course at a Glance]

Assignment 2

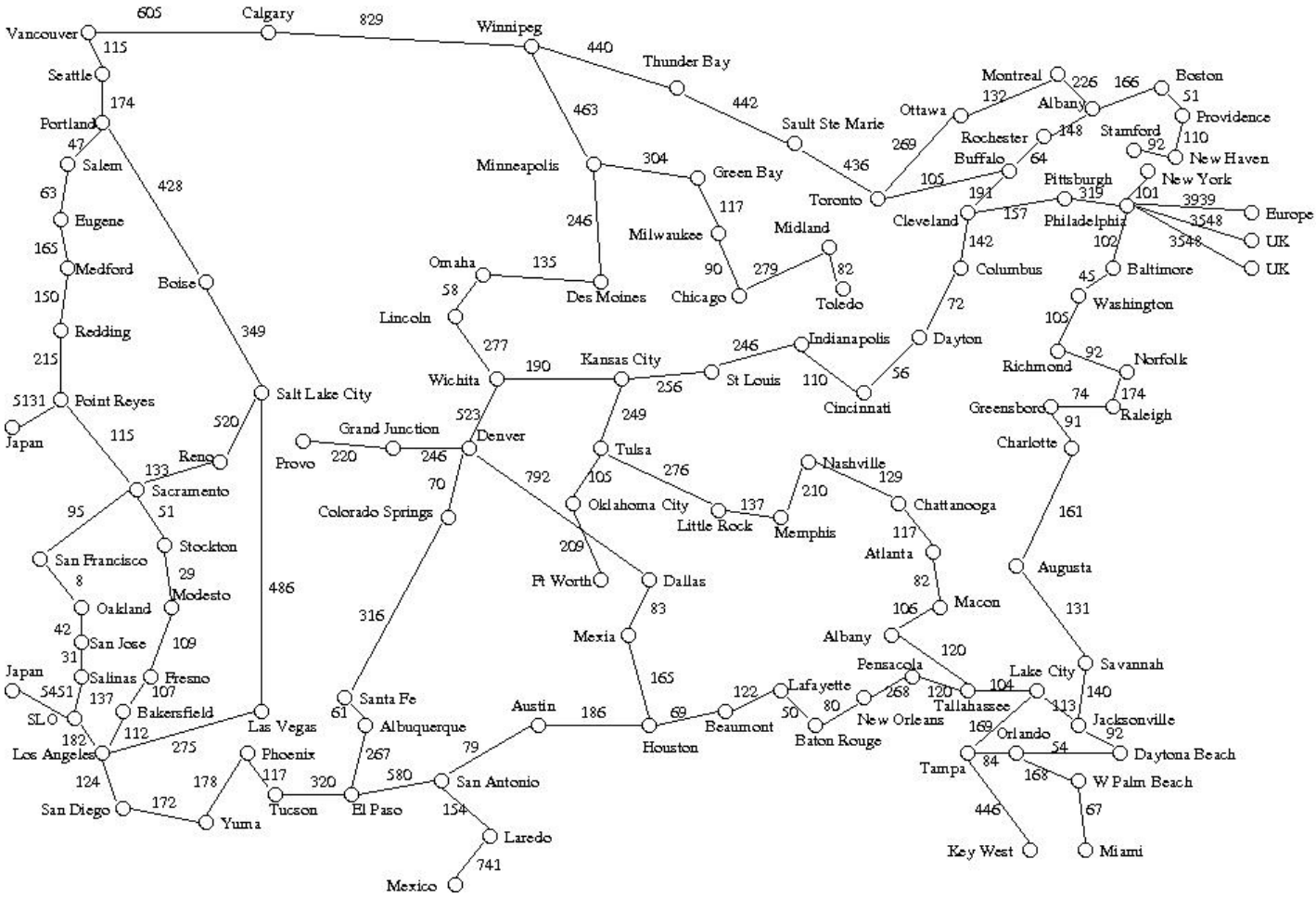
Due: before 2345 T 29 SEP

The Assignment

1. [50 points]

   This question concerns route-finding, with comparison of several search algorithms. This time, we're in the U.S. Here's jpg of the map below.

   The solution consists of the series of cities a network packet must pass through, each city connected to one or more others by network links of the indicated length. There are no other network links.



   In a language of your choice (Java or C++), implement the A* search algorithm. Your code should keep track of nodes expanded and should be able to compute the number of such nodes. Then run your algorithm on the telecom link map.

   - Name the source code file with the main function `SearchUSA`, with the file extension appropriate to the language.
   - The inputs should will be given through the command line, similar to Assignment 1.
     - In java:

       % java SearchUSA searchtype srccityname destcityname

     - In C++:

```
% mv ./a.out SearchUSA
% SearchUSA searchtype srccityname destcityname
```

- The searchtype should be either `astar`, `greedy`, or `uniform`.
- To save a bit of typing, the network is implemented as Prolog procedures in usroads.pl. This is a Prolog source file, and this assignment does not use Prolog. The file is provided solely as a convenience; you will have to modify it for use with your code. This time we will need the distances, and the longitude/latitude of the cities. (The percent sign (%) is a Prolog comment char, from there to end-of-line.)
- The spelling of `srccityname` and `destcityname` must be the as given in usroads.pl. Do NOT change the names from lower case to upper case.
- A node is said to be **expanded** when it is taken off a data structure and its successors generated.
- Use as a heuristic the straight line distance between cities. The straight-line distance between cities is computed using decimal degrees of latitude and longitude, which also given in the file. There is a complication in computing straight line distance from longitude and latitude that arises because the earth is roughly a sphere, not a cylinder. heuristic.pl is another Prolog file with a header comment indicating how the heuristic should work.
- As in Assignment 1, test your code on remote linux machines. Your code must compile without errors (warnings are okay) and must not print statements other than the ones stated above.

Now perform some experiments.

The experiments

1. [10 points] Experiment with executing your implementation of A* to find various paths, until you understand the meaning of the output. Are there any pairs of cities (A,B) for which the algorithm finds a different path from B to A than from A to B? Are there any pairs of cities (A,B) for which the algorithm expands a different total number of nodes from B to A than from A to B?

   Output in such cases should consist of the following:

   - A comma separated list of expanded nodes (the closed list);
   - The number of nodes expanded;
   - A comma-separated list of nodes in the solution path;
   - The number of nodes in the path;
   - The total distance from A to B in the solution path.

2. [10 points] Change your code so as to implement **greedy** search, as discussed in the web notes.
3. [10 points] Do enough exploration to find at least one path that is longer using greedy search than that found using A*, or to satisfy yourself that there are no such paths. Find at least one path that is found by expanding more nodes than the comparable path using A*, or satisfy yourself that there are no such paths. If there is such a path, list the nodes in the path and the total distance.

   Output in such cases should consist of the following:

   - A comma separated list of expanded nodes (the closed list);
   - The number of nodes expanded;
   - A comma-separated list of nodes in the solution path;
   - The number of nodes in the path;
   - The total distance from A to B in the solution path.

4. [10 points] Change your code so as to implement **uniform cost** search, as discussed in the web notes.
5. [10 points] Do enough exploration to find at least one path that is longer using uniform cost than that found using A*, or to satisfy yourself that there are no such paths. Find at least one path that is found by expanding more nodes than the comparable path using A*, or satisfy yourself that there are no such paths. If there is such a path, list the nodes in the path and the total distance.

   Output in such cases should consist of the following:

   - A comma separated list of expanded nodes (the closed list);
   - The number of nodes expanded;
   - A comma-separated list of nodes in the solution path;
   - The number of nodes in the path;
   - The total distance from A to B in the solution path.

6. As part of your answer, compare the solution paths and explain what happened, especially any weird behavior you might detect.

---

2. [20 points] Tic-tac-toe (also known as Noughts and Crosses) is a two-person, zero-sum game, in which player X and player O alternate placing their symbols in one of the blank spaces in a 3-by-3 grid that looks like this:

```
  |   |
-- -- -- -- --
  |   |
-- -- -- -- --
  |   |
```

The first player to place three of his symbols in a row -- horizontally, vertically, or diagonally -- wins.

1. [10 points] Beginning from the position

```
   |  O |
-- -- -- -- --
X |    |  O
-- -- -- -- --
   |  X |  O
```

   with X's turn to move, construct by hand the game-tree for the rest of the game. Assume the search horizon is the end of the game on all branches.

   Hint: You can save a lot of work by taking advantage of symmetry. For example,

```
   |  X |        |    |        |    |           |    |
-- -- -- -- --  -- -- -- -- --  -- -- -- -- --       -- -- -- -- --
   |    |    , X |    |    ,   |    |    , and  |    |  X
-- -- -- -- --  -- -- -- -- --  -- -- -- -- --       -- -- -- -- --
   |    |        |    |        |  X |           |    |
```

   are all the same position because of symmetry.
2. [5 points] Suppose the static evaluation function scores a win for O as +1, a draw as 0, and a loss for O as -1. At each level of your sketch of the game tree, indicate the value of each node based on its children. Indicate the best next move for X.
3. [5 points] Now use α-β pruning. At each level of your sketch of the game tree, indicate the value of the nodes and indicate any nodes that would be pruned. Explicitly indicate the final value of the α-β interval is for each node. Indicate the best next move for X.

---

3. [12 points] Use Propositional Logic to determine whether or not the following set of requirements is logically consistent. In other words, represent the following sentences in Propositional Logic, convert to Conjunctive Normal Form, and run Resolution until a contradiction is derived, or else show a model of all the expressions showing that no contradiction exists.

   > The system is in multiuser state if and only if it is operating normally. If the system is operating normally, the kernel is functioning. Either the kernel is not functioning or the system is in interrupt mode. If the system is not in multiuser state, then it is in interrupt mode. The system is not in interrupt mode.

   Use the following lexicon:

   `Propositional symbols:`

   - m -- The system is in multiuser state.
   - n -- The system is operating normally.
   - k -- The kernel is functioning.
   - i -- The system is in interrupt mode.

4. [18 points] Consider the following English sentences.

   - All cats are mammals.
   - The head of any cat is the head of a mammal.

   1. [6 points] Convert the sentences into first-order predicate logic. Be extremely careful about quantification; because not everything in the universe is a mammal, or a cat, or a head, you will need both kinds of quantification. Use the following lexicon:

```
Predicates: cat(X) -- X is a cat.
            mammal(X) -- X is a mammal.
            headOf(H,X) -- H is the head of X.
```

   2. [4 points] Convert the logic statements into CNF. HINT: With this lexicon, you will need two Skolem constants if you're doing this correctly.
   3. [6 points] Using resolution and the 4-part heuristic presented in class, prove using FOPL resolution that the second of the original sentences follows from the first. Number your clauses, and indicate explicitly step-by-step what resolves together, under what substitution.
   4. [2 points] Show a shorter proof that doesn't use the heuristic, if you can find one.

Deliverables

Submit by the above deadline a file containing:

1. All necessary sourcecode **files**,
2. A **readme** file (format: pdf) with Clear indication how to run your program including queries and giving any other pertinent information;
3. Your answers to any questions not requiring a program, including those asked as part of the search question.

Top of Page

Comments or suggestions are always welcome.
Dr. Dennis Bahler /