CIS 730 Artificial Intelligence
CIS 530 Principles of Artificial Intelligence
Fall 2013

Homework 2 of 8: Machine Problem (MP2)
Heuristic Search

Assigned: Sat 07 Sep 2013
Due: ~~Mon 16 Sep 2013~~ Wed 18 Sep 2013 (before midnight)

The purpose of this assignment is to exercise your basic understanding of uninformed and heuristic searches through implementation.

Each problem is worth 20% for CIS 730 students and 30% for CIS 530 students.
Upload to your KSOL drop box a .zip file named MP2.zip.

Use C#, C++, Java, or Python **only** to solve the first three problems. Specify which you are using in a README.txt file and name the programs accordingly: each problem should have a source code file mp2_i.[LANGUAGE]. Put the compile line in your README.txt file. The file names should be given on the command line, e.g., "`mp2_i < inputfile`". The file format consists of a list of attributes for the relation, each beginning with @, then zero or more rows.

Data format

Input will be taken from standard input in the following format:

```
| The vertical bar denotes comments.  Your program should ignore all
| input on a line containing the '|' symbol.
|
| The first non-comment line contains N, the number of nodes in the graph.
4
| The second non-comment line contains the unique start node.
| NB: The first node is 0, but this is not necessarily the start node.
0
| The third non-comment line contains a single goal node.
3
| The fourth non-comment line specifies J, the number of heuristics given.
| (You may ignore this unless you are doing the extra credit problem.)
1
| The fifth non-comment line starts the edge-cost adjacency matrix (* denotes
| infinite edge cost, and UNDIRECTED edges are symmetric across the diagonal).
| The example below denotes the adjacency list:
| 0 → (1) 1 → 2 (2) → NULL
| 1 → 3 (5) → NULL
| 2 → 3 (1) → NULL
| 3 → NULL
* 1 2 *
* * * 5
* * * 1
* * * *
| After the adjacency matrix, the heuristic evaluation vectors are given,
| where each row containing the heuristic value h(n) for a node n
3
6
1
0
```

The example above is not admissible (and therefore not monotonic/consistent). Why not? Does it matter?

Notes

- See the note posted in the class mailing list for guidelines on good coding style.
- **Java is recommended for this assignment.** You may use C++, but the instructor has not tested the provided code (and specifically does not recommend *AISearch*, the C++ package posted in the AI repository at http://bit.ly/neMY5D).
- Turn in a README.txt file with compilation and runtime documentation for the entire assignment.

1. **(530 / 730) ID-DFS and Bidirectional Search: using the AIMA Java or C++ code.** Download **one** of the two code archives from the *Artificial Intelligence: A Modern Approach* site:
   http://aima.cs.berkeley.edu/code.html

   a) Java: https://code.google.com/p/aima-java/
      Download: http://bit.ly/aima-code-java
      Read first: https://code.google.com/p/aima-java/wiki/GettingStarted
      Use: http://bit.ly/aima-code-java-search-uninformed
      Study the interface in:
      i. http://bit.ly/aima-code-java-search-graph
         General framework
      ii. http://bit.ly/aima-code-java-search-uninformed
         DFS, BFS, DLS, Uniform Cost, ID-DFS, Bidirectional

   b) Python: https://code.google.com/p/aima-python/
      Download: http://bit.ly/aima-code-python
      Read first: https://code.google.com/p/aima-python/wiki/ReadMe
      Study the interface in:
      i. http://bit.ly/aima-code-python-queue
         Basic priority queue data structure
      ii. http://bit.ly/aima-code-python-search-graph
         General framework
         BFS, DLS, Uniform Cost, ID-DFS, Bidirectional

   c) C# (Kris Noesgaard): http://www.cedarlearning.org/aima/readme_notes.txt
      Download: http://www.cedarlearning.org/aima/
      Save as: aima-cs.zip
      Unzip to: aima-cs
      Study the interface in:
      \aima-cs\src\aima\search\nodestore
      \aima-cs\src\aima\search\framework
      \aima-cs\src\aima\search\uninformed

   d) C++ (Larry Holder): http://bit.ly/aima-code-cpp-home
      Download: http://bit.ly/aima-code-cpp
      Save as: aima-cpp.zip
      Unzip to: aima-cpp
      Use: \aima-c++\code\mydean\04.Search\C++Code\Search
      Study the interface in:
      \aima-cpp\src\aima\search\nodestore
      \aima-cpp\src\aima\search\framework
      \aima-cpp\src\aima\search\uninformed

Using the given code, extend the constructors (Problem in the Java version) to read from the specified standard input and perform Iterative-Deepening Depth First Search (ID-DFS) and Bidirectional Search.

Your program must print out the actual path and total cost in the following format, when run with command line "`mp2_1 -b`" and "`mp2_1 -l [limit]`" (either a class file, e.g., "`java mp2_1 -b`", or a standalone executable).

ID-DFS: Best path found: 0 1 3, cost 1 + 5 = 6
Bidirectional BFS: Same. (When will the behavior be different? Can you come up with an example?)

Turn in all modified files and add specific compilation instructions given the AIMA code. In Java, give the exact command lines for compilation, including class paths, and specify target `mp2_1.class`; in C++, provide a Makefile with target `mp2_1`.

2. **(530 / 730). Testing Iterative Deepening and Bidirectional Search.** Run Iterative-Deepening-DFS and Bidirectional Search, which you implemented in MP2-1. Compare your results to DLS and BFS. You **may** use the provided code, but cite your sources in comments and your README file.

   Your program must print out the actual path and total cost in the following format, when run with command line "`mp2_2 `".

   ID-DFS: [Show each level being expanded]
   Best path found: 0 1 3, cost 1 + 5 = 6

3. **(530 / 730). A/A* search.** Implement A/A* search using Best-First-Search applied with functions for *g* and *h* (these need not be downward function arguments, but you should pass in some selector constant for the specified heuristic).

   Your program must print out the actual path and total cost in the following format, when run with command line "`mp2_3 [heuristic-number]`". (Your program may ignore the heuristic number if you are not completing the extra credit exercise, but it should accept the input.)

   A/A*: Best path found: 0 2 3, cost 2 + 1 = 3

4. **(730) Simplified Memory-Bounded A/A*.** Modify your solution to MP2-2 and MP2-3 to implement SMA/SMA* as described in Chapter 4 of Russell and Norvig *2e*.

   Your program must print out the actual path and total cost in the following format, when run with command line "`mp2_4`".

   SMA/SMA*: [Show each level being expanded]
   Best path found: 0 2 3, cost 2 + 1 = 3

   In comments, explain the difference between SMA and IDA. Give an example of a graph where the two algorithms behave differently.

5. **(730 only) BFS / Branch and Bound.** Implement Branch and Bound and Breadth-First Search as special cases of your solution to MP2-3.

   Your program should be invoked using "`mp2_5 -bnb`" (what heuristic value should be used here?) or "`mp2_5 -bfs`" (which suppresses the edge cost).

6. **(530 only: 10%) Analysis.** Problem 4.7, p. 135. Prove that if a heuristic is consistent, it must be admissible. Give an example of an admissible heuristic that is not consistent.

Class Participation (required)

Post your responses to the class message board "Discussion and Questions" under the thread for Homework 2:

Please post any unclear points you may have on search to the class mailing list. That is, ask questions about any search-related topic for which your understanding is unclear. This includes uninformed (blind) search, informed (heuristic) search, analysis and proofs on either type of sear, search methods such as *iterative deepening*, and *bidirectional* search.

Also, consider the problem of navigating through a simple bitmap world such as:
http://www.thelimeydragon.com/map.php?map=et
How exactly would you transform this problem into a representation that matches the input to MP2?

If you prefer, feel free to ask about local and global search, games and game tree search, constraint satisfaction problems (CSP), or the term project topics covered in Lectures 1 - 3.

Extra Credit (20%)

Modify your search to handle `heuristic-number`, which selects one of the column vectors $\boldsymbol{h}j$ (where $0 \leq j \leq J$ and $0 \leq n \leq N$ as specified in the example comments above).