CIS 730 Artificial Intelligence
CIS 530 Principles of Artificial Intelligence
Fall 2007

Homework 2 of 10: Machine Problem (MP2)
Heuristic Search

Assigned: Wed 29 Aug 2007
Due: Wed 12 Sep 2007 (before midnight)

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

This homework assignment is worth a total of 20 points.
Each problem is worth 4 points for CIS 730 students and 7 points for CIS 530 students.
Upload to your KSOL drop box a .zip file named MP2-XYZ.zip, with your initials in place of XYZ.

Use C++ or Java **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 file1 file2". 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.
2
| The fifth non-comment line starts the edge-cost adjacency matrix.
| 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 is a tuple containing all heuristic values h_j(n) for a node n
| The leftmost heuristic in the example below is admissible - why?
3 3
1 2
1 2
0 1
```

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://snipurl.com/wihl).
- Turn in a README.txt file with compilation and runtime documentation for the entire assignment.

1. **(530 / 730) Breadth-first and depth-first 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: http://aima.eecs.berkeley.edu/java/aima.zip
   Save as: aima-java.zip
   Unzip to: aima-java
   Use: \aima-java\src\aima\search\uninformed
   Study the interface in:
   \aima-java\src\aima\search\nodestore (FIFO, LIFO, Priority)
   \aima-java\src\aima\search\framework (Problem, QueueSearch, NodeExpander, etc.)

   b) C++: http://www-cse.uta.edu/~holder/courses/cse5361/spr96/code.tar.gz
   Save as: aima-c++.zip
   Unzip to: aima-c++
   Use: \aima-c++\code\mydean\04.Search\C++Code\Search
   Study the interface in:
   \aima-java\src\aima\search\nodestore
   \aima-java\src\aima\search\framework

   Using the given code, extend the constructors (Problem in the Java version) to read from the specified standard input and perform Depth-Limited Search (DLS) and Breadth-First Search (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_1 -b`" and "`mp2_1 -l [limit]`" (either a class file, e.g., "`java mp2_1 -b`", or a standalone executable).

   BFS: Best path found: 0 1 3, cost 1 + 5 = 6
   DLS: Best path found for limit 2: 0 1 3, cost 1 + 5 = 6

   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). Iterative Deepening Search.** Implement Iterative-Deepening-DFS by generalizing from DLS in MP2-1.

   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]`" where `heuristic-number` selects one of the column vectors ***h**j* (where $0 \le j \le J$ and $0 \le n \le N$ as specified in the example comments above).

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

4. **(730) Iterative Deepening A/A*.** Modify your solution to MP2-2 and MP2-3 to implement IDA/IDA* as described in Chapter 4 of Russell and Norvig *2*[e].

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

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

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).

Class Participation (required)

Post any unclear points you may have on search to the class mailing list (**CIS730-L@listserv.ksu.edu**). 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 topic, and search methods such as *pathmax*, *iterative deepening*, and *bidirectional* search.

If you prefer, feel free to ask about constraint satisfaction problems (CSP) or game tree search instead.