Assignment 1: Search Methods

Deadline

Submission: 5pm, Friday 20 April, 2018 (week 6).

This assignment is worth 10% of your final mark. It is an individual assignment; no group work.

Late submissions policy

No late submissions are allowed.

Programming languages

Your implementation can be written in Python, Java, C, C++ or MATLAB. The assignment will be tested on the University machines, so your code must be compatible with the language version installed on those machines.

Submission

Your assignment must be completed individually using the submission tool PASTA (http://comp3308.it.usyd.edu.au). In order to connect to the website, you'll need to be connected to the university VPN. You can read this page to find out how to connect to the VPN. PASTA will allow you to make as many submissions as you wish, and each submission will provide you with feedback on each of the components of the assignment. You last submission before the assignment deadline will be marked, and the mark displayed on PASTA will be the final mark for your assignment.

1. The 3-digit puzzle

In this assignment, you will implement a number of search algorithms to solve the 3-digit puzzle.

Given are two 3-digit numbers called S (start) and G (goal) and also a set of 3-digit numbers called *forbidden*. To solve the puzzle, we want to get from S to G in the smallest number of moves. A move is a transformation of one number into another number by adding or subtracting 1 to one of its digits. For example, a move can take you from 123 to 124 by adding 1 to the last digit or from 953 to 853 by subtracting 1 from the first digit. Moves must satisfy the following constraints:

1. You cannot add to the digit 9 or subtract from the digit 0;
2. You cannot make a move that transforms the current number into one of the forbidden numbers;
3. You cannot change the same digit twice in two successive moves.

Note that since the numbers have 3 digits, at the beginning there are at most 6 possible moves from S. After the first move, the branching factor is at most 4, due to the constraints on the moves and especially due to constraint 3.

For the purpose of this assignment numbers starting with 0, e.g. 018, are considered 3-digit numbers.

2. Tasks

1. Write a program to find a solution of the puzzle using the following 6 search strategies: BFS, DFS, IDS, Greedy, A* and Hill-climbing. Use the Manhattan heuristic as a heuristic for A* and Greedy and also as the evaluation function in Hill-climbing.
   The Manhattan heuristic for a move between two numbers A and B is the sum of the absolute differences of the corresponding digits of these numbers, e.g. $h(123, 492) = |1 - 4| + |2 - 9| + |3 - 2| = 11$.
2. Avoid cycles. When selecting a node from the fringe for expansion, if it hasn't been expanded yet, expand it, otherwise discard it. *Hint: It is not just the three digits that you need to consider when determining if you have expanded a node before. Two nodes are the same if a) they have the same 3 digits; and b) they have the same 'child' nodes.*
3. Generate the children in the following order:
   a. 1 is subtracted from the first digit
   b. 1 is added to the first digit
   c. 1 is subtracted from the second digit
   d. 1 is added to the second digit
   e. 1 is subtracted from the third digit
   f. 1 is added to the third digit
   Example: the order of the children of node 678 coming from parent 668 is 578, 778, 677, 679. Note that there are no children for the second digit as it already has been changed in the previous move (constraint 3).
4. For the heuristic search strategies: if there are nodes with the same priority for expansion, expand the last added node.
5. Set a limit of 1000 expanded nodes maximum, and when it is reached, stop the search and print a message that the limit has been reached (see section "Input and Output" for the exact message required).

3. Input and Output

As your program will be automatically tested, it is important that you adhere to these strict rules for program input and output.

Input

Your program should be called ThreeDigits, and will be run from the command line with the following arguments:

1. A single letter representing the algorithm to search with, out of B for BFS, D for DFS, I for IDS, G for Greedy, A for A*, H for Hill-climbing.
2. A filename of a file to open for the search details. This file will contain the following:

```
start-state
goal-state
forbidden1,forbidden2,forbidden3,…,forbiddenN (optional)
```

For example, the file `puzzle.txt` might contain the following:

```
345
555
355,445,554
```

This file means that the search algorithm will start at state 345, and the goal is state 555. The search may not pass through any of 355, 445 or 554. Remember that the last line may not be present; i.e. there are no forbidden values.

The following examples show how the program would be run for each of the submission languages, assuming we want to run the A* search algorithm, and the input is in a file called `sample.txt`.

Python (version 2.7.9):

```
python ThreeDigits.py A sample.txt
```

Java (version 1.8):

```
javac ThreeDigits.java
java ThreeDigits A sample.txt
```


C (gcc version 4.4.7):


```
gcc -lm -w -std=c99 -o ThreeDigits ThreeDigits.c *.c
./ThreeDigits A sample.txt
```

C++ (gcc version 4.4.7):


```
g++ -o ThreeDigits ThreeDigits.cpp
./ThreeDigits A sample.txt
```

MATLAB:

```
matlab -nodesktop -nosplash -nojvm -nodisplay -r
"try;ThreeDigits('A','sample.txt');catch me;disp(me.message);end;quit"
```

Output

You program will output **two lines only**. The first line will contain the solution path found from the start node to the goal node (inclusive), in the form of states separated by commas. If no path can be found with the given parameters, then the first line should be “No solution found.”.

The second line should be the order of nodes expanded during the search process, in the form of states separated by commas. If no path was found, this should still print the list of expanded nodes. Remember that this list should never exceed 1000 states (point 5 in the Tasks section).

Examples

`sample.txt:`

```
320
110
```

Note that the outputs of BFS and IDS here are quite long, and so the second line has wrapped. These outputs are still only two lines.

Search Method	Output
BFS	320,220,210,110 320,220,420,310,330,321,210,230,221,410,430,421,210,410, 311,230,430,331,221,421,311,331,110
DFS	320,220,210,110 320,220,210,110
IDS	320,220,210,110 320,320,220,420,310,330,321,320,220,210,230,221,420,410, 430,421,310,210,410,311,330,230,430,331,321,221,421,311, 331,320,220,210,110
Greedy	320,310,210,211,111,110 320,310,210,211,111,110
Hill-Climbing	No solution found. 320,310,210
A*	320,220,210,110 320,310,210,220,210,110

4. Submission Details

This assignment is to be submitted electronically via the PASTA submission system.

Your submission files should be zipped together in a single .zip file and include a main program called ThreeDigits. Valid extensions are .java, .py, .c, .cpp, .cc, and .m. If your program contains only a single file, then you can just submit the file without zipping it.

Upload your submission on PASTA under **Assignment 1**.