Programming Assignment 01 (PA1)

CMSC 471: Introduction to Artificial Intelligence
University of Maryland, Baltimore County
Spring 2025
Instructor: KMA Solaiman

Maximum Points: 100 **Date: Feb 10, 2025**

The following table gives the overall point breakdown for this assignment.

Question	1	2
Points	60	40

Honor Code

You are to complete this assignment on your own: that is, the code and writeup you submit must be entirely your own. However, you may discuss the assignment at a high level with other students or on the discussion board. Note at the top of your assignment who you discussed this with or what resources you used (beyond course staff, any course materials, or blackboard discussions).

Language and External Resources

Your code must compile, and you will use Python. If necessary, we must be able to compile it as well. You may use and reference code from `https://github.com/aimacode`, though acknowledge if you do so. Failure to do so, or the use of other external resources without prior written permission from the instructor, will be considered an academic integrity violation and result, in a minimum, in a 0 on this assignment.

What To Turn In

You must turn in two(+) items on Blackboard:

1. A writeup in PDF format that answers the questions. The filename should be `hw01.pdf`.
2. `dc.py` with your solution, any environment files (if applicable), and execution instructions necessary to replicate your output.

Provide any necessary analyses and discussion of your results. Students may make multiple submissions, but please note that **only the last submission will be considered.**

The DOGCAT Game

Before you start on the problem definition, take a look at the 'what to do' section below. You are already familiar with 8-puzzle game and water-jug problem. We have provided sample codes to implement those as search problems. Familiarize yourself with them first.

1. Coding Problem [60]

DOGCAT is a simple word game. You are given two words with the same number of letters (we will be using three- and four-letter words) in each, e.g., DOG and CAT. Your goal is to transform the first word into the second by replacing one letter at a time with any other letter as long as the result is a proper English word. For example, we could change DOG to FOG but not to GOG (not a proper English word) or to GOD (two letters were changed). Thus, one way of getting from DOG to CAT might be:

DOG => COG => COT => CAT

There are many ways to make most translations. Here is another way:

DOG => HOG => HOT => HAT => CAT

Some examples are very hard to do. Try changing WHY to ASK.

The game can be generalized to words of other lengths, and we can also make the problem more interesting by defining other properties of the transformation we want to optimize. For this assignment, we'll work with words of three or four characters.

We will define three cost measures:

1. The simplest cost measure we will call **steps**. It is just the number of steps in the sequence from the initial word to the goal word. The cost of DOG => COG => COT => CAT is just three.

2. The second one, **scrabble**, associates a cost with using a new letter as a replacement based on the scrabble game.

The cost of replacing any letter of a word by a A, E, I, O, U, L, N, S, T or R is one.
The cost of using a D or G is two.
The cost of replacing a letter with a B, C, M, or P is three.
The cost of using a F, H, V, W, or Y as a replacement is four.
Using a K costs five.

Using a J or a X costs six.
The cost of replacing a letter with a Z or Q is ten.
Thus, the transformation DOG => COG => COT => CAT costs 3 + 1 +1 or five.

3. The third cost measure is based on how common the words after the initial one is. The idea is that going from CAM to a very rare word. (e.g., CWM) should cost more than going to a common word like CAT. We've provided a list of all of the "legal" English three and four-letter words along with a measure of the relative frequency of each. The cost of the sequence DOG => JOG => JOT => COT => CAT is computed as 1+R for each step where R is the measure of how rare the word introduced by the step is. When you unzip words34.txt.gz file, you will see entries including the following:

    jog 10.052741
    jot 11.177437
    cot 10.071238
    cat 6.886906
    so, the cost is 4 + the sum of the scores, or 42.188322.

What to do

The provided zip folder contains the starter file `dc.py` that you will need to complete and the AIMA Python required files `search.py` and `utils.py` files. Specifically study the AIMA `search.py` file. You should look at the Problem class and its methods, in particular, and the search algorithms.
Write a generic version of a DC class that solves instances of the dog_cat problem given an initial and final word using the aima code for astar_search. **We've created a stub file for dc.py, which is the only file you need to complete.** To complete this stub, you must finish the DC class by completing the init, actions, result, goal_test, h, and repr methods. You will probably need to write some auxiliary functions as well. **Your heuristic h function must be admissible**, that is, always returning an estimate that is less than or equal to the true value. It should not be the trivial estimate of 0, of course. Note that both your path_cost and h methods will have to be sensitive to the problem's cost parameter. We've provided you with a dictionary of all of the legal English three and four-letter words along with their rarity measures. See this datafile words34.txt.gz and the code in the dc.py stub that loads it:

```
import gzip
...
dict_file = "words34.txt.gz"
dictionary = {}
for line in gzip.open(dict_file):
word, n = line.strip().split('\t')
dictionary[word] = float(n)
```

There is also p8.py and wj.py, with corresponding notebook codes, but they do not need to be changed. They are provided to you **for practice and as an example of solving the 8-puzzle and the water-jug problem.**

Testing your code

Once you've written your `dc.py` file you can test it with `dcsolve.py` and `dctest.py`. The `dcsolve.py` script takes an initial and goal word and an optional cost measure and shows the solution found by our `dc.py` problem along with the cost and time taken.

```
python dcsolve.py dog cat
    dc(dog,cat,steps) cost: 3.00; time:0.001; solution: dog cog cot cat; deltas:[0.0, 0.0, 0.0]
python dcsolve.py why ask
    dc(why,ask,steps) cost: 5.00; time:0.004; solution: why wha aha ahs ass ask; deltas:[-2, -1, -1, 0.0, 0.0]
python dcsolve.py test bush
    dc(test,bush,steps) cost: 3.00; time:0.001; solution: test best bust bush; deltas:[0.0, 0.0, 0.0]
python dcsolve.py love hate
    dc(love,hate,steps) cost: 3.00; time:0.001; solution: love hove have hate; deltas:[0.0, 0.0, 0.0]
python dcsolve.py bear duck steps
    dc(bear,duck,steps) cost: 4.00; time:0.002; solution: bear beak beck buck duck; deltas:[0.0, 0.0, 0.0, 0.0]
python dcsolve.py bear duck scrabble
    dc(bear,duck,scrabble) cost: 11.00; time:0.002; solution: bear beak beck buck duck; deltas:[0.0, 0.0, 0.0, 0.0]
python dcsolve.py bear duck frequency
    dc(bear,duck,frequency) cost: 38.24; time:1.025; solution: bear beak beck buck duck; deltas:[-27.011, -15.546, -
7.034, 0.0]
```

The dctest.py script takes no arguments and runs a number of examples using each of the three cost measures. **Note that your program may not produce exactly the same answers, since the A* algorithm with an admissible heuristic finds one of the shortest solutions to a problem -- there may be several that are equally short.** But the number of steps should remain same.

2. Written Questions [5 *8 = 40]

Answer the following questions in a separate PDF file named 'hw01.pdf':

(1) Describe in words the heuristic you used for the steps cost and explain why it is admissible.

(2) Describe in words the heuristic you used for the scrabble cost and explain why it is admissible.

(3) Describe in words the heuristic you used for the frequency cost and explain why it is admissible.

(4) Given an initial word W1 and goal word W2, if the shortest path from W1 to W2 has N steps, will the shortest path from W2 to W1 also have N steps? Explain why or why not.

(5) Using the steps cost, what is the longest possible path from a three-letter word to a four-letter word? You can answer based on specific examples or based on generalization from multiple different words.