Problem 1
(a)
State space description of the problem:
Any possible word that meets the required input length and the given hints .

What information should individual states contain?
Each letter of the current input word. And hints of the current word (Which letter is correct and in the right spot. Which letter is correct but in the wrong spot. Which letter is totally wrong.)

What are the valid actions that an agent can take in each state?
Guessing the new word using all given hints.

(b)
Fully observable:
The agent can sense which letter is correct or not and whether the right letters are in the right spot.

Single agent:
This is a single-player game.

Deterministic:
We can predict the next state given the hints and the current input word.

Sequential:
The current input word will depend on past input word and hints.

Static:
The environment will not change when the player is thinking.

Discrete:
The valid input words are finite.

Problem 2

(a)

$\exists p\ [p, A; 1-p, C\ ] \sim B$

(b)

No

U(L1) = p*U(A) + (1-p)*U(C)

U(L2) = q*U(B) + (p-pq)*U(A) + (1-q)(1-p)*U(C)

U(L1) – U(L2) > 0

pq*U(A) – q*U(B) + q(1-p)*U(c) >0

p*U(A) + (1-p)*U(C) > U(B)

so $\exists p\ [p, A; 1-p, C\ ] \succ B$, such that L1 $\succ$ L2 regardless of q.

For example, if $\exists p\ [p, A; 1-p, C\ ] \succ B$ and q $=$ 1, L1 $\succ$ L2 also holds.

(c)

U(L2) = q*U(B) + (p-pq)*U(A) + (1-q)(1-p)*U(C)

Solve following two equations:

q*U(B) = (p-pq)*U(A)

q*U(B) = (1-q)(1-p)*U(C)

Getting the result:

$\mathrm{p} = \frac{\mathrm{U(C)}}{\mathrm{U(A)+U(C)}}$

$\mathrm{q} = \frac{\mathrm{U(A)\times U(C)}}{\mathrm{U(A)\times(U(B)+U(C))+U(B)\times U(C)}}$

problem 3 ~~#~~

(a) $E[U(X_i)] = 100 \times \frac{1}{50} + (-10) \times \frac{1}{50} + 1 \times \frac{48}{50} = 2.76$

(b) choosing goat $\rightarrow$ switch to FAANG

$$P_1 = \frac{48}{50} \times \frac{1}{44}$$

choosing Tuition $\rightarrow$ switch to FAANG

$$P_2 = \frac{1}{50} \times \frac{1}{44}$$

choosing goat $\rightarrow$ switch to ~~FAANG~~ Tuition

$$P_3 = \frac{48}{50} \times \frac{1}{44}$$

choosing FAANG $\rightarrow$ switch to Tuition

$$P_4 = \frac{1}{50} \times \frac{1}{44}$$

choosing goat $\rightarrow$ switch to goat

$$P_5 = \frac{48}{50} \times \frac{42}{44}$$

choosing FAANG $\rightarrow$ switch to goat

$$P_6 = \frac{1}{50} \times \frac{42}{44}$$

choosing Tuition $\rightarrow$ switch to goat

$$P_7 = \frac{1}{50} \times \frac{42}{44}$$

Expected Utility of switching to another door.

$$100 \cdot (P_1 + P_2) + (-10)(P_3 + P_4) + 1 \cdot (P_5 + P_6 + P_7) = 2.959$$

(c) Because $2.959 > 2.76$, It's better to switch.

$VPI = 2.959 - 2.76 = 0.199$

(d) choosing FAANG

$P_8 = \frac{1}{50}$

choosing goat ~~car~~ or Tuition → switch to FAANG

$P_9 = \frac{49}{50} \cdot \frac{1}{44}$

Expected Utility : $100 (P_8 + P_9) = 4.227$

Problem 4
(a)
DFS:
S -> a -> b -> d -> G1
Solution:
S -> b -> d -> G1

BFS:
S -> a -> b -> c -> d -> G1
Solution:
S -> c -> G1

UCS:
S -> c -> b -> d -> G2
S -> b ->d -> G2

(b)
For what range of values is h an admissible heuristic?
[0,2]

For what range of values of h does A∗ return a suboptimal solution?
h>3

For what range of values of h does A∗ expand fewer nodes than UCS?
h>3

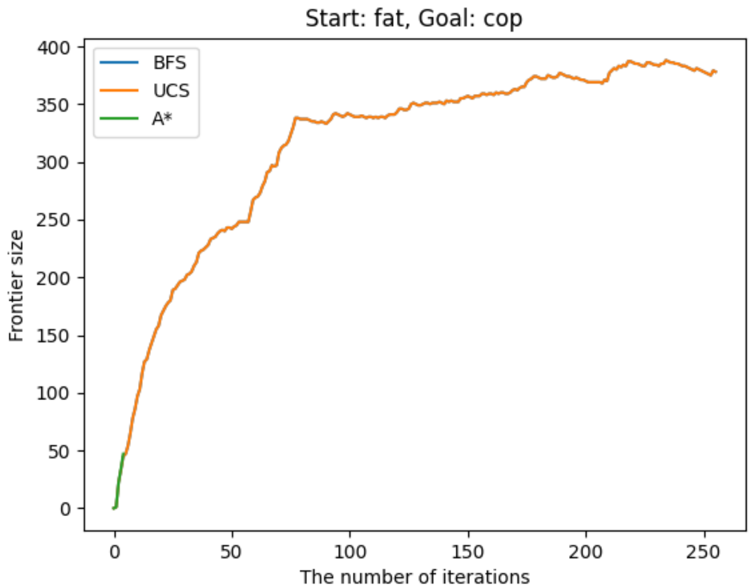
```python
"""
5.1: Best-first search
"""
def best_first_search(start, goal, f):
    """
    Inputs: Start state, goal state, priority function
    Returns node containing goal or None if no goal found, total nodes expanded,
    frontier size per iteration
    """
    node = {'state':start, 'parent':None, 'cost':0}
    frontier = []
    reached = {}
    nodes_expanded = 0
    frontier_size = [len(frontier)]


    # COMPLETE THIS FUNCTION
    heappush(frontier, tuple([f(node, goal)]+[ch for ch in node['state']]+[node]))
    reached[node['state']] = node['cost']

    while frontier:
        frontier_size.append(len(frontier))
        node = heappop(frontier)[-1]
        nodes_expanded += 1
        if node['state'] == goal:
            return node, nodes_expanded, frontier_size
        words = successors(node['state'])
        for idx, word in words:
            if word not in reached or node['cost']+1 < reached[word]:
                reached[word] = node['cost']+1
                new_node = {'state':word,
                            'parent':node,
                            'cost':node['cost']+1}
                heappush(frontier, tuple([f(new_node, goal)]+
                                         [x for x in new_node['state']]+
                                         [new_node]))

    return None, nodes_expanded, frontier_size
```

```python
"""
5.2: Priority functions
"""
def f_dfs(node, goal=None):
    # IMPLEMENT THIS FUNCTION
    return -node['cost']

def f_bfs(node, goal=None):
    # IMPLEMENT THIS FUNCTION
    return node['cost']

def f_ucs(node, goal=None):
    # IMPLEMENT THIS FUNCTION
    return node['cost']

def f_astar(node, goal):
    # IMPLEMENT THIS FUNCTION
    h = 0
    goal_list = [ch for ch in goal]
    word_list = [ch for ch in node['state']]
    for i in range(len(goal)):
        if goal_list[i]!=word_list[i]:
            h += 1

    return node['cost'] + h
```

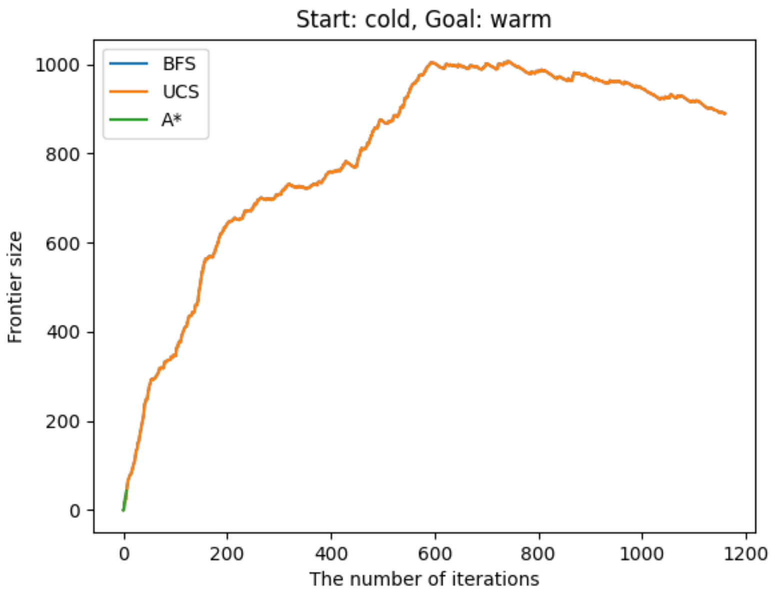
Problem 5

5.3

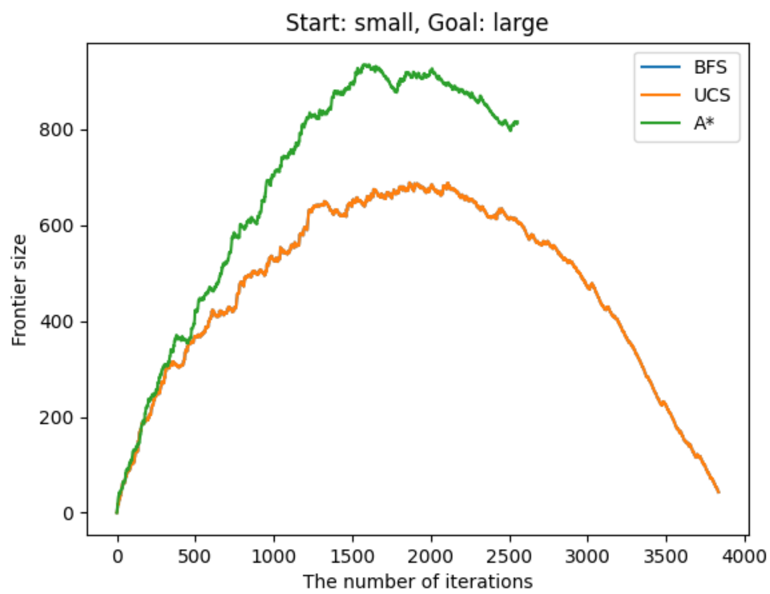
```
/Users/david/opt/anaconda3/envs/ai/bin/python /Users/david/Documents/4701/hw1/hw1.py
Start: fat
Goal: cop

The solution path of DFS algorithm is:
['fat', 'bat', 'baa', 'boa', 'bob', 'bib', 'bid', 'aid', 'add', 'ado', 'ago', 'age', 'ace
The DFS algorithm execution time is: 176.51083517074585
The length of the solution of DFS algorithm is: 30
The number of nodes expanded of DFS algorithm is: 29634

The solution path of BFS algorithm is:
['fat', 'cat', 'cap', 'cop']
The BFS algorithm execution time is: 1.4766089916229248
The length of the solution of BFS algorithm is: 4
The number of nodes expanded of BFS algorithm is: 255

The solution path of UCS algorithm is:
['fat', 'cat', 'cap', 'cop']
The UCS algorithm execution time is: 1.4781670570373535
The length of the solution of UCS algorithm is: 4
The number of nodes expanded of UCS algorithm is: 255

The solution path of A* algorithm is:
['fat', 'cat', 'cap', 'cop']
The A* algorithm execution time is: 0.017102718353271484
The length of the solution of A* algorithm is: 4
The number of nodes expanded of A* algorithm is: 4

Process finished with exit code 0
```




```
Start: cold
Goal: warm

The solution path of BFS algorithm is:
['cold', 'cord', 'card', 'ward', 'warm']
The BFS algorithm execution time is: 8.452422142028809
The length of the solution of BFS algorithm is: 5
The number of nodes expanded of BFS algorithm is: 1160

The solution path of UCS algorithm is:
['cold', 'cord', 'card', 'ward', 'warm']
The UCS algorithm execution time is: 7.338251113891602
The length of the solution of UCS algorithm is: 5
The number of nodes expanded of UCS algorithm is: 1160

The solution path of A* algorithm is:
['cold', 'cord', 'card', 'ward', 'warm']
The A* algorithm execution time is: 0.031935930252075195
The length of the solution of A* algorithm is: 5
The number of nodes expanded of A* algorithm is: 6

Process finished with exit code 0
```




```
Start: small
Goal: large

The solution path of BFS algorithm is:
['small', 'shall', 'shale', 'shade', 'shads', 'sheds', 'seeds', 'sends', 'sands', 'sangs', 'tangs', 'tango', 'mango', 'mange', 'marge', 'large']
The BFS algorithm execution time is: 27.122417211532593
The length of the solution of BFS algorithm is: 16
The number of nodes expanded of BFS algorithm is: 3834

The solution path of UCS algorithm is:
['small', 'shall', 'shale', 'shade', 'shads', 'sheds', 'seeds', 'sends', 'sands', 'sangs', 'tangs', 'tango', 'mango', 'mange', 'marge', 'large']
The UCS algorithm execution time is: 28.635447025299072
The length of the solution of UCS algorithm is: 16
The number of nodes expanded of UCS algorithm is: 3834

The solution path of A* algorithm is:
['small', 'shall', 'shale', 'shade', 'shads', 'sheds', 'seeds', 'sends', 'sands', 'sandy', 'randy', 'rangy', 'range', 'mange', 'marge', 'large']
The A* algorithm execution time is: 17.657507181167603
The length of the solution of A* algorithm is: 16
The number of nodes expanded of A* algorithm is: 2553

Process finished with exit code 0
```



(1)
Which algorithms are optimal?
A* algorithm is optimal because it has fewer iterations to find the goal among four algorithms.

Why do we generally want to avoid DFS for this problem?
Because the tree is very deep in this problem, so it takes many iterations to get to the bottom of one branch of the tree. But the goal node is not very deep, so the DFS is less efficient compared to other algorithms

(2)
Which algorithms have roughly the same performance regardless of puzzle?
The BFS algorithm and UCS algorithm have the same performance. Because all edges have the same cost 1, both algorithms generate the same sorting order in the frontier. Therefore the frontier will pop the same item at each iteration.

(3)
Explain the trend of the frontier size.
The frontier size grows up very fast at the beginning and gradually slow down. Then it reaches a maximum point and start to go down. It is kind of like a concave quadratic function.

What does it mean for a search to finish when the frontier size is still increasing, near its maximum, or decreasing?

The size increasing means the goal being at the position of very shadow level of the tree, and the size decreasing means the goal being at the very bottom level of the tree. While the maximum point means the goal being at the position between above two situation.