Artificial Intelligence
Take home Quiz 01

NAME: Hoang Do
DATE: 2/14/2019

This homework quiz is meant to help you prepare for the mid-term exams. The mid-term will cover concepts from Agents, Environments, Search (both informed and un-informed), Adversarial Search and Constraint Satisfaction Problems..

1. Prove each of the following statements, or give a counterexample (6 points)
   a. Breadth-first search is a special case of uniform-cost search.

When all step costs are equal, g(n) is a multiple of depth n. BFS and UCS will behave the same.

   b. Depth-first search is a special case of best-first tree search.

Depth-first search is Best-first tree search with $f(n) = -\text{depth}(n)$

   c. Uniform-cost search is a special case of A* search.

A* search with $h(n) = 0$ is Uniform-cost search.

2. Decide whether the following statements are true or false. If true, explain why. If false, give a contradicting example. Recall that B is the average branching factor and L is the length of the shortest path from start to goal. (8 points)

   a) Bi-directional BFS is always faster than BFS when B >= 2 and L >= 4.

False, If we search on a tree from leaf to root, BFS is fast since branching factor is 1, while Bi-directional BFS take time to reverse while tracing a lot of branches

   b) A* search always expands fewer nodes than DFS does.

False: DFS expands fewer nodes than A* search with admissible heuristic.

Not always true, A* can be shorter, most of time

c) For any search space, there is always an admissible and consistent A* heuristic.

True, when $h(s) = 0$

d) IDA* does not need a priority queue as in A*, but can use the program stack in a recursive implementation as in DFS.

True, because the inner loop in IDA* is DFS, not A*

3. Tree search algorithm

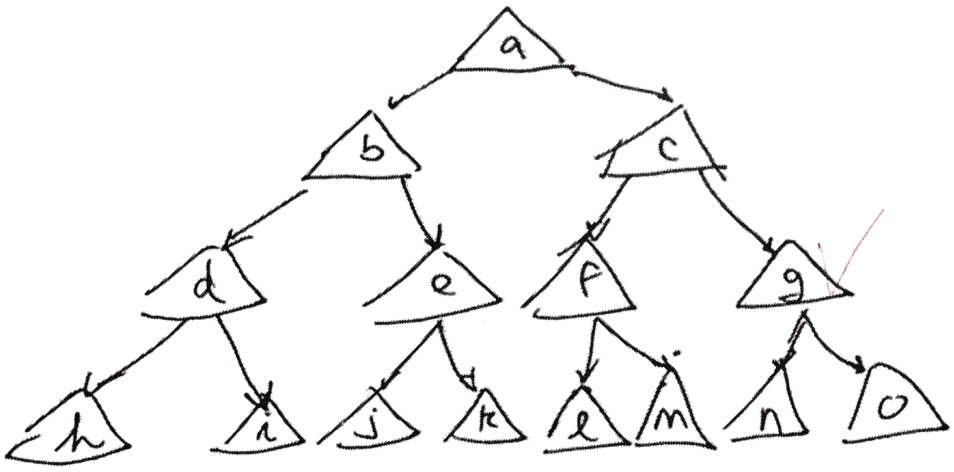
Tree search algorithm is applicable to searches where there is no worry about re-visiting prior states. It is simpler than graph-search because there is no need to maintain the closed set (of previously visited states). Use tree search to solve the following problems

**NOTES**

"The intuitive idea behind generic search algorithm, given a graph, a set of start nodes, and a set of goal nodes, is to incrementally explore paths from the start nodes. This is done by maintaining a **frontier** (or **fringe**) of paths from the start node that have been explored. The frontier contains all of the paths that could form initial segments of paths from a start node to a goal node. Initially, the frontier contains trivial paths containing no arcs from the start nodes. As the search proceeds, the frontier expands into the unexplored nodes until a goal node is encountered. To expand the frontier, the searcher selects and removes a path from the frontier, extends the path with each arc leaving the last node, and adds these new paths to the frontier. A search strategy defines which element of the frontier is selected at each step."

*Refer to the reference books (Artificial Intelligence: Foundations of Computation Agents - Chapter 3 (It was in our reading materials))*

A. Breadth-first search using FIFO queue




0
1
2
3
b=2
d=3

Frontier (fill from L–R):

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ~~a~~ | ~~b~~ | ~~c~~ | ~~d~~ | ~~e~~ | ~~f~~ | ~~g~~ | h | i | j | k | l | m | n | o |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | |

Using the tree search algorithm, perform the search for **goal state *o*** starting from initial state *a*.

Expand new state/action pairs from left to right. Draw nodes in the Fringe as they are expanded. When they get removed for goal-testing, cross them out. (4 points)

**Draw nodes as a circle** with a letter inside. This is to visually distinguish them from states (triangles).

**Treat the Fringe as a FIFO queue. This should produce breadth-first search.**

a. What goal node is returned by the algorithm? (2 points)

o

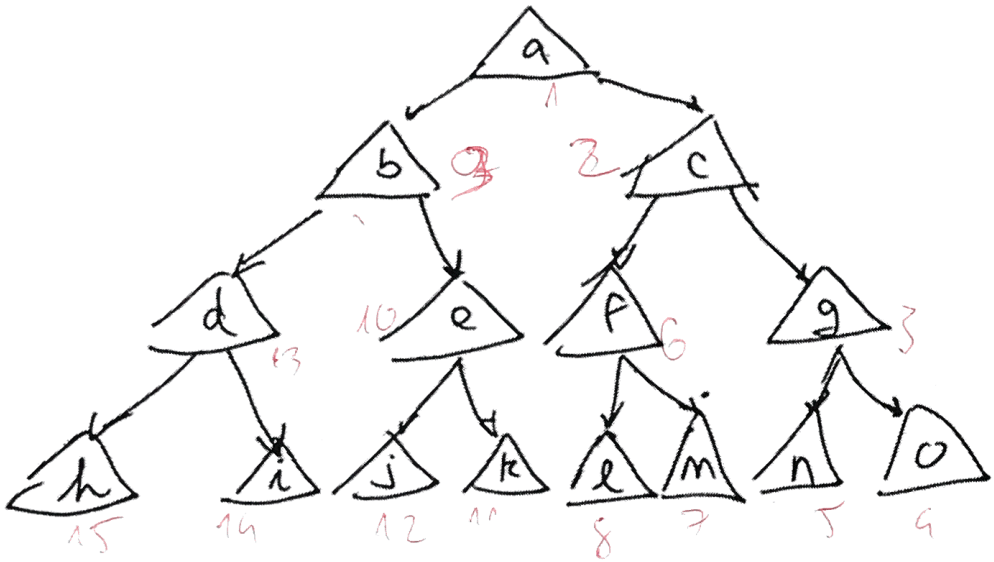
b. How many states were expanded in total? (2 points)

7 states 14 $b^{d+1} - 2$

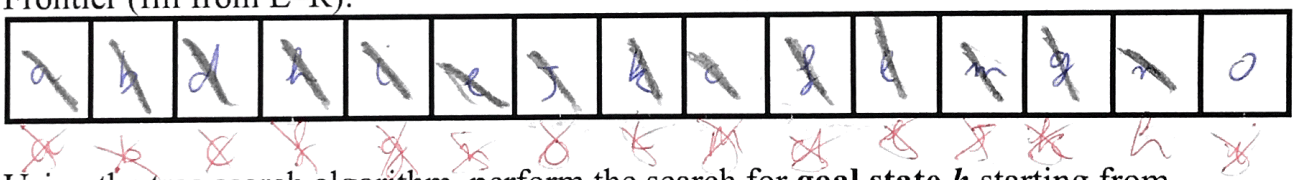
c. At most, how many nodes were actively in the fringe at one time during the algorithm? (2 points)

8 nodes 8 nodes

B) Depth-first search using LIFO stack




Frontier (fill from L–R):



Using the tree search algorithm, perform the search for **goal state *h*** starting from initial state *a*.

Expand new state/action pairs from left to right. Draw nodes in the Fringe as they are expanded. When they get removed for goal-testing, cross them out. (4 points)

**Draw nodes as a circle** with a letter inside. This is to visually distinguish them from states (triangles).

**Treat the Fringe as a LIFO stack. This should produce depth-first search.**

When done:

a. What goal node is returned by the algorithm? (2 points)

h

b. How many states were expanded in total? (2 points)

13 states

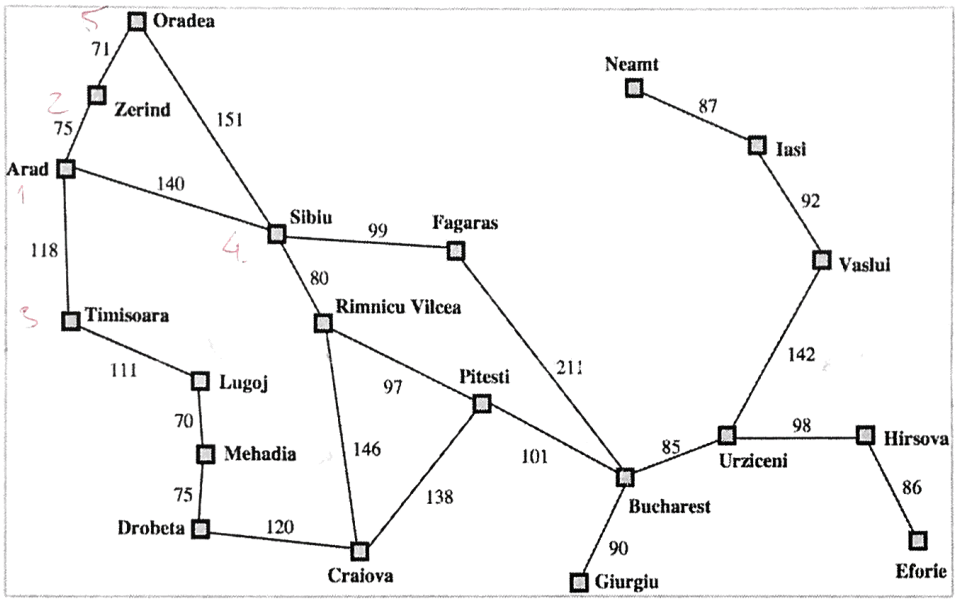
14

c. At most, how many nodes were actively in the fringe at one time during the algorithm? (2 points)

1 node  4 nodes

4. Uniform cost search

Uniform cost search (UCS) is a breath-first search where the node to be expanded is the one with the lowest accumulated path cost in the frontier. Let's use it to search for the best path from Arad to Bucharest. "Best path" is defined




as shortest distance.

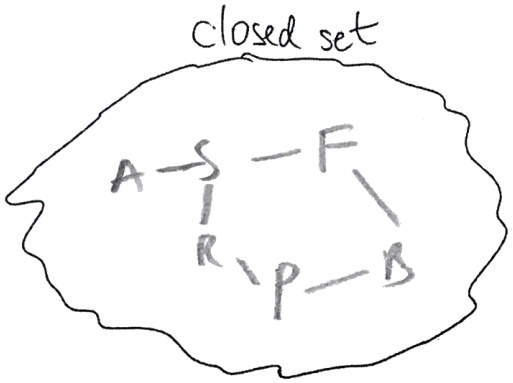
We will use **Graph Search**

```
function GRAPH-SEARCH(problem, fringe) return a solution, or failure
    closed ← an empty set
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST(problem, STATE[node]) then return node
        if STATE[node] is not in closed then
            add STATE[node] to closed
            for child-node in EXPAND(STATE[node], problem) do
                fringe ← INSERT(child-node, fringe)
            end
    end
```

A Z T X ~~O~~ ~~I~~ V R F ~~N~~ P C B D

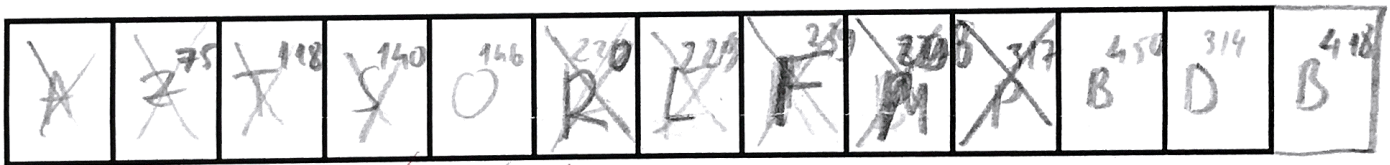
0 75 118 140 146 ~~220~~ 220 239 299 317 375 ~~450~~ 374

**Uniform Cost Search using priority queue**




A 0
T 118
S 140
F 146

Frontier (fill from L–R):




A 0 Z 75 S 140 T 118 O 146 L 209 F 239 R 220

Start state is Arad.
Goal state is Bucharest.

Draw nodes in the Fringe as they are expanded. When they get removed for goal-testing, cross them out and put them in the closed set as states.

Draw nodes as a circle with a letter inside, and include the total path cost. (4 points)

Treat the Fringe as a priority queue. When performing "Remove-Front," take the node with the smallest accumulated cost.

a. What path is returned by the algorithm? (2 points)

Arad – Sibiu – Rimnicu Vilcea – Pitesti – Bucharest

b. What is its path cost (total distance)? (2 points)

140 + 80 + 97 + 101 = 418

c. What did you observe that was interesting about the algorithm? (2 points)

It may cost a lot of time to find the goal but it make sure to give shortest path.

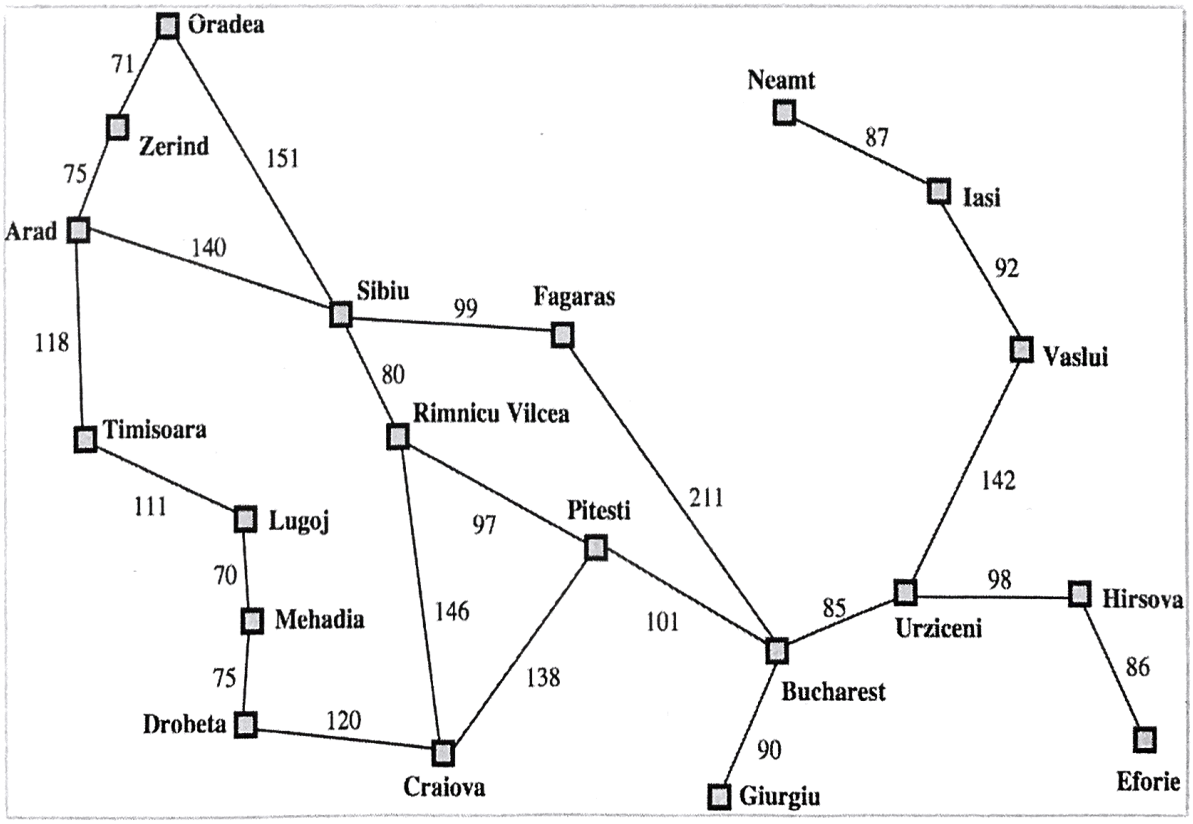
5. A* Search

A* search (pronounced "a star") is a breath-first search where the node to be expanded is the one with the best (lowest) value of (accumulated path cost in the frontier) + (heuristic distance to the goal). In other words, the priority function *f* for a node *n* is:

$$f(n) = g(h) + h(n)$$

The heuristic function we will use is straight line distance (*hsld*), which is guaranteed to be admissible for Euclidean spaces.

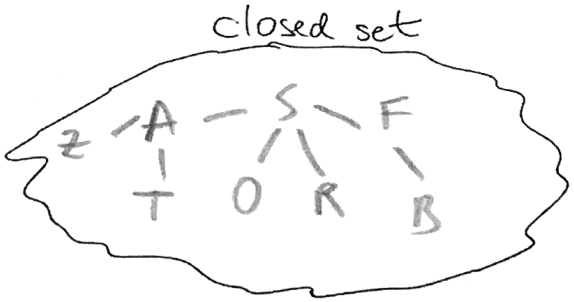
The actual path cost may be equal to or greater than this heuristic value, but it cannot be less than it.



| | | | |
|---|---|---|---|
| **Arad** | 366 | **Mehadia** | 241 |
| **Bucharest** | 0 | **Neamt** | 234 |
| **Craiova** | 160 | **Oradea** | 380 |
| **Drobeta** | 242 | **Pitesti** | 100 |
| **Eforie** | 161 | **Rimnicu Vilcea** | 193 |
| **Fagaras** | 176 | **Sibiu** | 253 |
| **Giurgiu** | 77 | **Timisoara** | 329 |
| **Hirsova** | 151 | **Urziceni** | 80 |
| **Iasi** | 226 | **Vaslui** | 199 |
| **Lugoj** | 244 | **Zerind** | 374 |

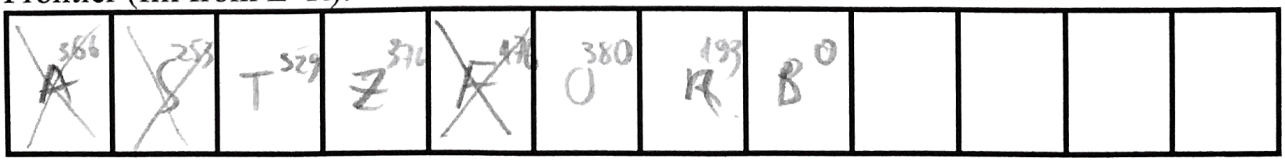
Perform A* search using the priority function $f(n) = g(h) + h(n)$. Start state is Arad. Goal state is Bucharest. Draw nodes in the Fringe as they are expanded. Include their $f$ value at the point of insertion. (4 points)

When nodes get removed for goal-testing, cross them out and put them in the closed set as states.




Treat the Fringe as a priority queue. When performing "Remove-Front," take the node with the smallest $f=g+h$ cost.

Frontier (fill from L–R):




a. What path is returned by the algorithm? (2 points)

Arad — Sibiu — Fagaras — Bucharest

b. What is its path cost (total distance)? (2 points)

140 + 99 + 211 = 450

c. What did you observe that was interesting about the algorithm, compared to UCS? (2 points)

A* search. have expanded less nodes than Uniform cost search but itt takes longer distant than UCS

6. More A*

a) Suppose we have two admissible heuristic functions, *h1* and *h2*, where for all search states s, *h1(s)* < *h2(s)*. In other words:

$$\forall(s): h1(s) < h2(s)$$

What will be the impact on the search process? Specifically:

i) Optimality: Will the search return the best solution when using *h1*? What about when using *h2*? (3 points)

No, if $h_2(n) > h_1(n)$ for all n, then $h_2$ dominates $h_1$ and is better for search.

both admissible

ii) Efficiency: Will more nodes be expanded when using *h1* or *h2*? The same? (3 points)

- A* search using $H_2$ will not expand more nodes than A* using $H_1$.
- Heuristics do not affect the length of the path found, A* will eventually find the optimal path for an admissible heuristic

↳ 1 less efficienty because could more nodes to expanded

b) Suppose we use a heuristic *h3* that is *not* admissible (i.e., it produces values that are larger than the actual best path to the goal). What will be the implications of this if only one solution exists, and there is only one path to it. Consider this in terms of:

i) Optimality: Will *h3* find the solution? Why or why not? (2 points)

$h_3$ will find the solution but it would take more time before reach ~~to the~~ solution path.

Yes,

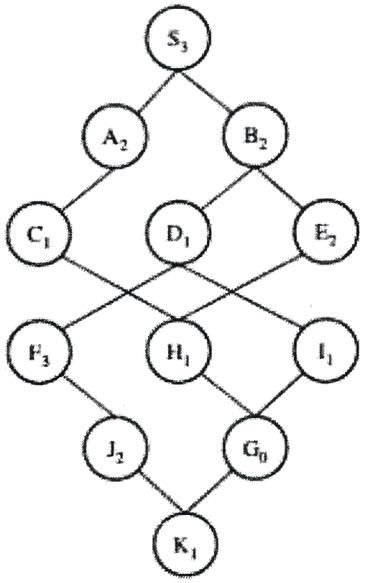
ii) Efficiency: Will *h3* cause more, fewer, or the same number of nodes to be expanded vs. an admissible heuristic? Why? (3 points)

A non-admissible maybe closer to reality than admissible one or it may not. It depends on the path to the goal. Since there is only one solution exists and only one path to the goal; $h_3$ cause more nodes to expanded compare with an admissible heuristic

Bad, because it will expand a lot of nodes in many direction.

7. Consider the problem of moving a knight on a 3x4 board, with start and goal states l labeled as S and G in figure 3. The search space can be translated into the following graph. The letter in each node is its name and the subscript digit is the heuristic value. All transitions have cost 1.

| | | | |
|---|---|---|---|
| $S_3$ | $H_1$ | $D_1$ | $K_1$ |
| $I_1$ | $J_2$ | $A_2$ | $E_2$ |
| $C_1$ | $B_2$ | $G_0$ | $F_3$ |




Make the following assumptions:

- All the algorithms do not generate paths with loops.
- Nodes are selected in alphabetical order when the algorithm finds a tie.
- A node is visited when that node is at the front of the search queue.

Write the sequence of nodes in the order visited by the specified methods.

Note: You may find it useful to draw the search tree corresponding to the graph above.

(a) Depth-first search. (4 points)

S A C H E B D F J K G.

(b) Breadth-first search. (4 points)

S A B C D E H F I G

(c) A* search. In addition to the sequence of nodes visited, you are also to attach the estimated total cost f(s)=g(s)+h(s) for each node visited and return the final path found and its length. (4points)

S (0+3=0) / A (1+2=3) / B (1+2=3) / C (2+1=3) /

D (2+1=3) / E (2+2=4) / H (3+1=4) / G (4+0=4)

=) Final path: S A C H G

# Artificial Intelligence
# Take Home Quiz 2

**Important note**

**If something is not clear, make a reasonable assumption, state it and work on the assignment.**

NB.

a) To get partial credit, all your formulas and calculations should be shown.

**NAME:** Hoang Do

**Student ID:** 01521888

| QUESTION | TOPIC | POINTS | SCORE |
|---|---|---|---|
| 1 | Adversarial Search | 25 | |
| 2 | Genetic Algorithms | 25 | |
| | | | |
| | TOTAL | 50 | |

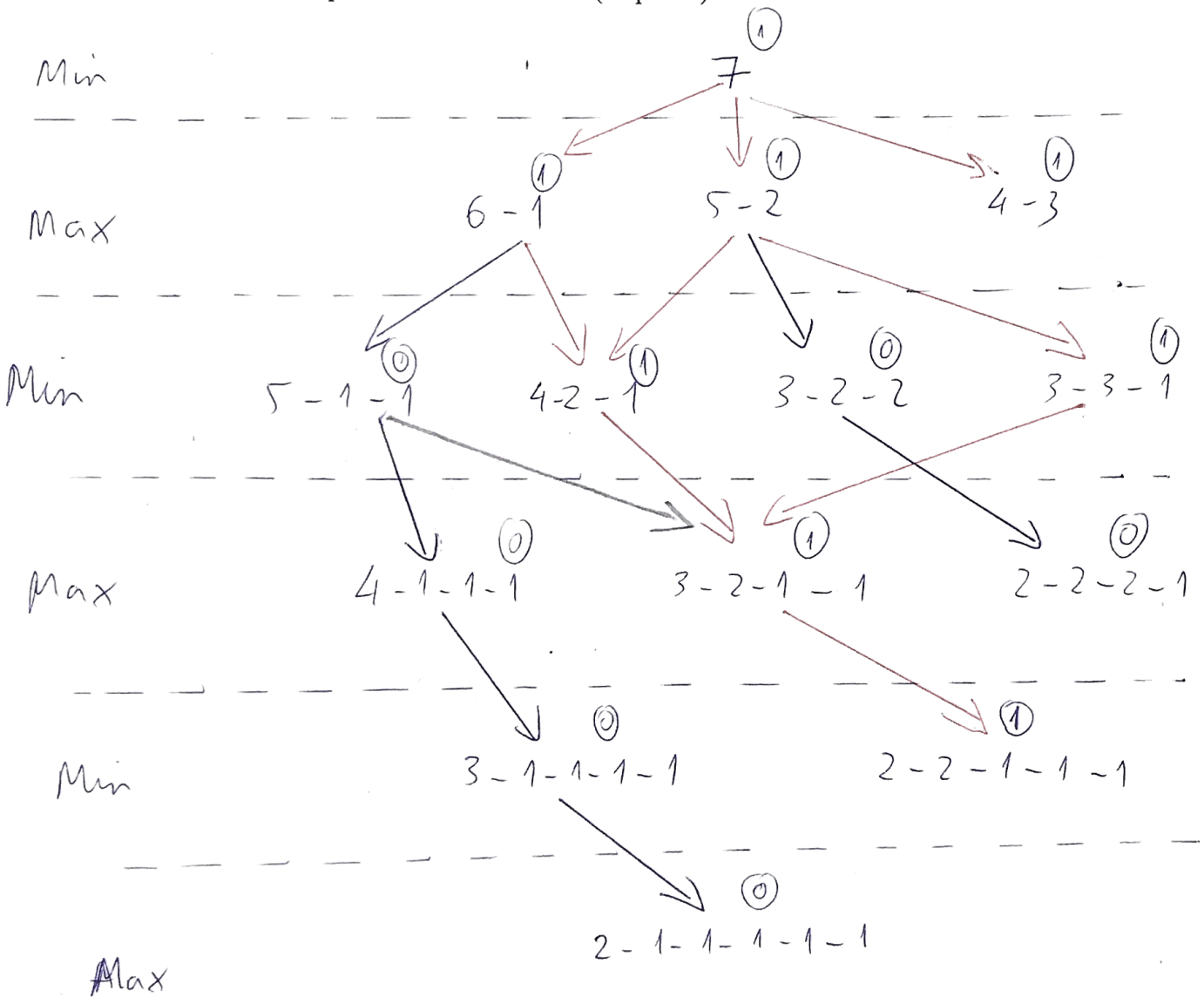
# Artificial Intelligence
# Take Home Quiz 2

**Question 1: Adversarial Search**

(a) Nim is a two-player game. The rules are as follows.

The game starts with a single stack of 7 tokens. At each move a player selects one stack and divides it into two non-empty, non-equal stacks. A player who is unable to move loses the game.

Draw the complete search tree for nim (10 points).



# Artificial Intelligence
# Take Home Quiz 2

**Question 2: Genetic Algorithms**

a) Given the following parents, $P_1$ and $P_2$, and the template T

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **$P_1$** | A | B | C | D | E | F | G | H | I | J |
| **$P_2$** | E | F | J | H | B | C | I | A | D | G |
| **T** | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |

Show how the following crossover operators work

- uniform crossover
- order-based crossover

with regards to genetic algorithms **(8 )**

Uniform crossover

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $C_1$ | A | F | C | D | B | C | I | H | D | J |
| $C_2$ | E | B | J | H | E | F | G | A | I | G |

order-base crossover.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $C_1$ | A | E | C | D | F | B | I | H | G | J |
| $C_2$ | E | B | J | H | C | D | F | A | I | G |

Use this problem description for parts b to e.
Assume we have the following function

$$f(x) = x^3 - 60 * x^2 + 900 * x + 100$$

where x is constrained to 0..31. We wish to maximize f(x) (the optimal is x=10)
Using a binary representation we can represent x using five binary digits.

b) Given the following four chromosomes give the values for x and f(x). **(2)**

| **Chromosome** | **Binary String** |
|---|---|
| $P_1$ | 11100 |
| $P_2$ | 01111 |
| $P_3$ | 10111 |
| $P_4$ | 00100 |

| Chromosome | Binary String | x | f(x) |
|---|---|---|---|
| $P_1$ | 11100 | 28 | 212 |
| $P_2$ | 01111 | 15 | 3475 |
| $P_3$ | 10111 | 23 | 1227 |
| $P_4$ | 00100 | 4 | 2804 |

# Artificial Intelligence
## Take Home Quiz 2

(b) Assume two players, min and max, play nim (as described above). Min plays first.

If a terminal state in the search tree developed above is a win for min, a utility function of zero is assigned to that state. A utility function of 1 is assigned to a state if max wins the game.

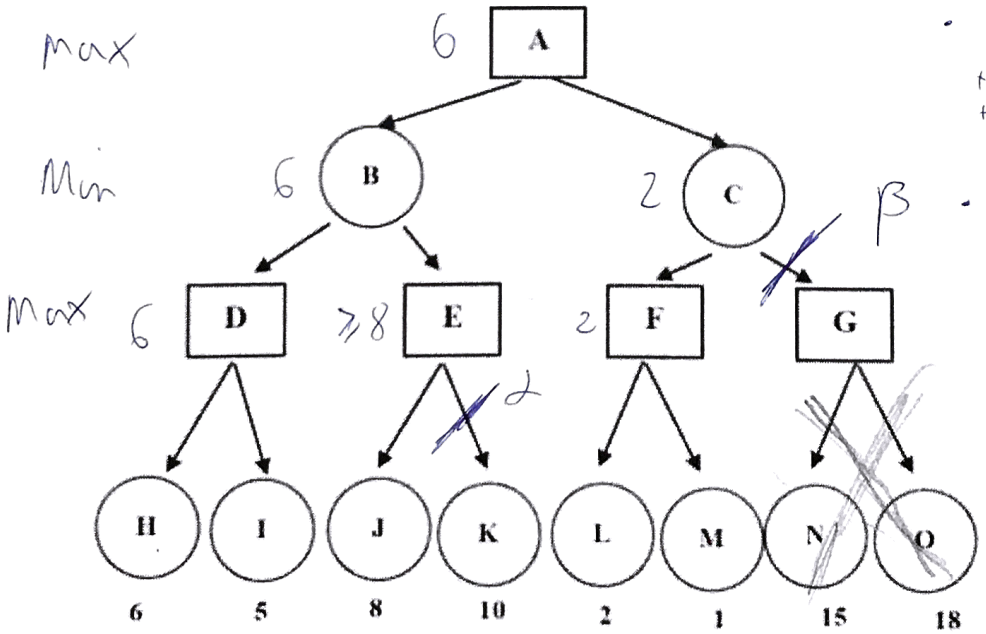
Apply the minimax algorithm to the search tree to assign utility functions to all states in the search tree. (3)

Proof in question a.

(c) If both min and max play a perfect game, who will win? Explain your answer. (3)

Since Min goes first, if both Min and Max playe a perfect game, Max is guaranteed to win if it follow the red arrow. Min can not change the path to blue arrow where Min can win the game.

(d) Given the following search tree, apply the alpha-beta pruning algorithm to it and show the search tree that would be built by this algorithm. Make sure that you show where the alpha and beta cuts are applied and which parts of the search tree are pruned as a result. (9)




- α cut.
  + E has value at least 6.
  + E > D and B = min(D, E)
- β cut.
  + F = 2.
  + C = min(F, G) and
    B > 6, A = max(B, C)

# Artificial Intelligence
## Take Home Quiz 2

c) If $P_3$ and $P_2$ are chosen as parents and we apply one point crossover show the resulting children, $C_1$ and $C_2$. Use a crossover point of 1 (where 0 is to the very left of the chromosome)
Do the same using $P_4$ and $P_2$ with a crossover point of 2 and create $C_3$ and $C_4$ **(6)**

$C_1$ : 11111
$C_2$ : 00111
$C_3$ : 00111
$C_4$ : 01100

d) Calculate the value of x and f(x) for $C_1..C_4$. **(2)**

| Chromosome | Binary string | x | f(x) |
|---|---|---|---|
| $C_1$ | 11111 | 31 | 131 |
| $C_2$ | 00111 | 7 | 3803 |
| $C_3$ | 00114 | 7 | 3803 |
| $C_4$ | 01100 | 12 | 3988 |

e) Assume the initial population was x={17, 21, 4 and 28}. Using one-point crossover, what is the probability of finding the optimal solution? Explain your reasons. **(7)**

17 : 10001
21 : 10101
4 : 00100
28 : 11100

optimal solution: x = 10 => x: 01010.

probability of finding optimal solution is zero because we need a "1" in position 2,4 (counting from the left of binary string) but 17, 21, 4, 28 provide binary string without "1" in position 2. => We can't find the optimal solution

Hoang Do



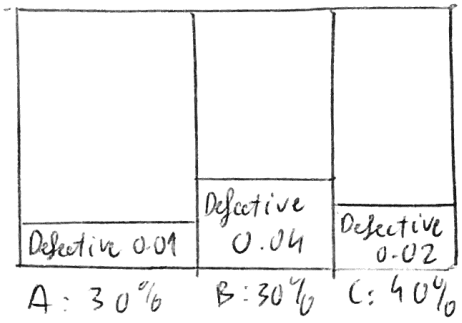
**Problem 1 [15].** Widgets are manufactured in three factories: A B and C. The proportion of defective widgets from each factory are as follows:

Factory A: .01
Factory B: .04
Factory C: .02

Factories A and B produce 30% of the widgets apiece, and the remaining 40% come from Factory C.

What is the likelihood that a given widget is defective?

Solve graphically *and* with Bayes'. Put your numeric answer in the box and show your work below.




A: Widgets from factory A
B: Widgets from factory B
C: Widgets from factory C.
E: Widget is defective.

$$\Rightarrow P(E) = P(A).P(E|A) + P(B).P(E|B) + P(C).P(E|C)$$
$$= \frac{3}{10}.(0.01) + \frac{3}{10}.(0.04) + \frac{4}{10}.(0.02).$$
$$= 0.023$$

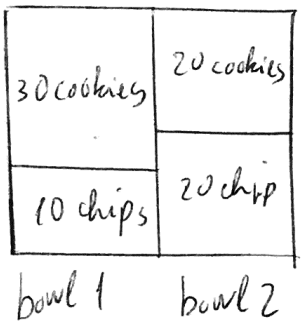
*Question adapted from study.com.*

Hoang Do



**Problem 2 [20].** Suppose there are two full bowls of cookies. Bowl #1 has 10 chocolate chip and 30 plain cookies, while bowl #2 has 20 of each. Our friend Stacy picks a bowl at random, and then picks a cookie at random. We may assume there is no reason to believe Stacy treats one bowl differently from another, likewise for the cookies. The cookie turns out to be a plain one. How probable is it that Stacy picked it out of Bowl #1?

Solve graphically *and* with Bayes'. Put your numeric answer in the box and show your work below.




Hypothese: A: Stacy picked from bowl 1.
B: Stacy picked from bowl 2.
$\Rightarrow P(A) = P(B) = \frac{1}{2}$.

E: Stacy picked a plain cookie.

* Bayes's theorem

$$P(A|E) = \frac{P(E|A) \cdot P(A)}{P(E)} = \frac{P(E|A) \cdot P(A)}{P(A) \cdot P(E|A) + P(B) \cdot P(E|B)}$$

$$= \frac{\frac{30}{40} \cdot \frac{1}{2}}{\frac{1}{2} \cdot \frac{30}{40} + \frac{1}{2} \cdot \frac{20}{40}} = 0.6$$

2. (10 points) Given the Bayes net shown in the Figure below; A, B, C, D, and E are all Boolean variables. P(A="T") is simply denoted as P(A).

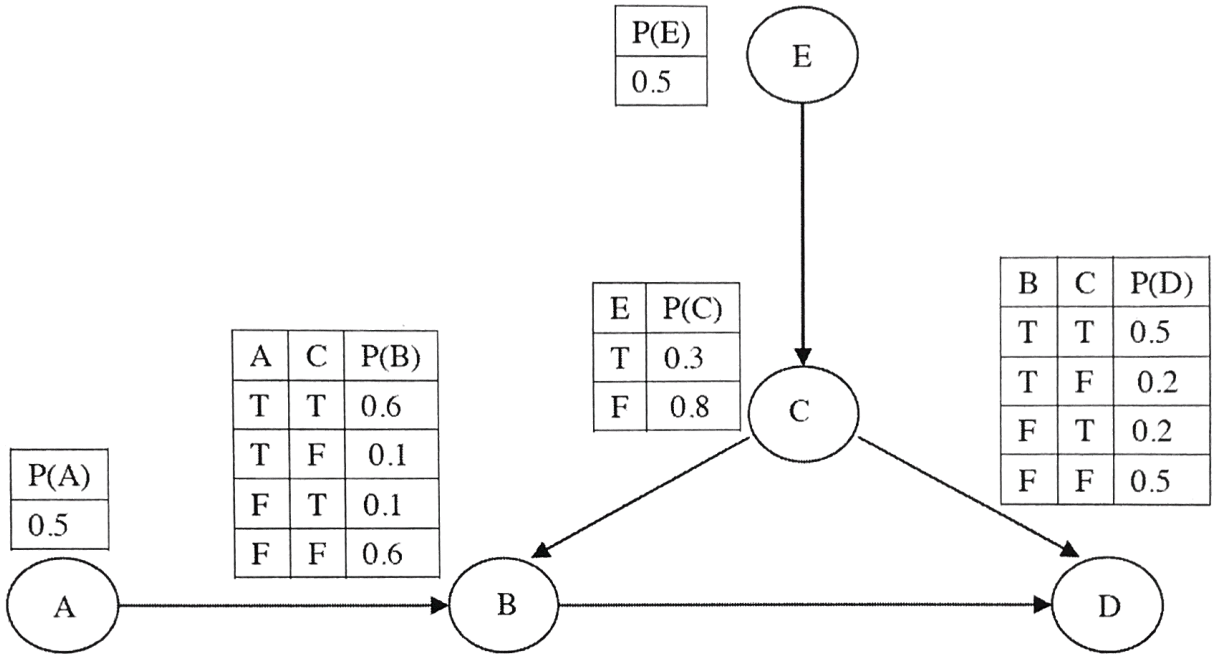


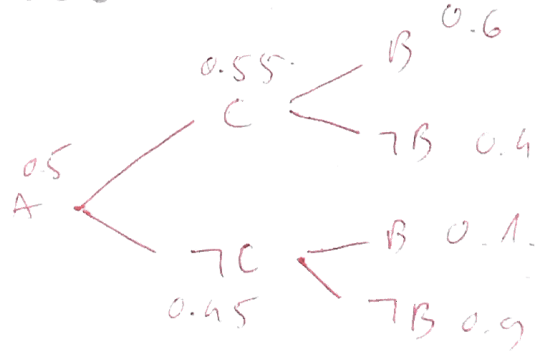

Fig 2: Bayes Net
Note: In the *following, the notations:* A ⊥ B *means A is independent of B;* A ⊥ B | C *means A is conditionally independent of B given C.*

a. Please judge if the following independence assumptions are correct or not:
   i. (1 point) B ⊥ E | C True
   ii. (1 point) A ⊥ D False
   iii. (2 point) A ⊥ D | B False
   iv. (2 point) A ⊥ D | B, C False. True

b. (2 point) Compute the value of P(C) 0.55 $P(C) = P(C|E) \cdot P(E) + P(C|\neg E) \cdot P(\neg$

c. (2 point) Compute the value of P(B|A) 0.375

$P(B|A) = \frac{P(B,A)}{P(A)}$



# COMP4200 / COMP 5430 AI: Homework V
# Markov Decision Processes

UMASS - LOWELL

Name: Hoang Do

Student ID: 01521888

| Question | Points |
|---|---|
| 1 | 13 |
| 2 | 12 |
| 3 | 10 |
| Total | |

11/14/0643

**Instructions:**

1. This examination contains 6 pages, including this page.
2. Write your answers in this booklet. If you must write on the back page, please indicate **very** clearly on the front of the page that you have written on the back of the page.
3. You **may** use any resources, including lecture notes, books, other students or other engineers, but you should provide a reference.
4. You may use a calculator. You may not share a calculator with anyone.

## Question 1: MDP: Fight or Run

[12 pts] A boy is being chased around the school yard by bullies and must choose whether to Fight or Run.

a. There are three states:

- Ok (O), where he is fine for the moment.
- Danger (D), where the bullies are right on his heels.
- Caught (C), where the bullies catch up with him and administer noogies.

b. He begins in state O 75

c. He begins in state D 25

The graph of the MDP is given here:

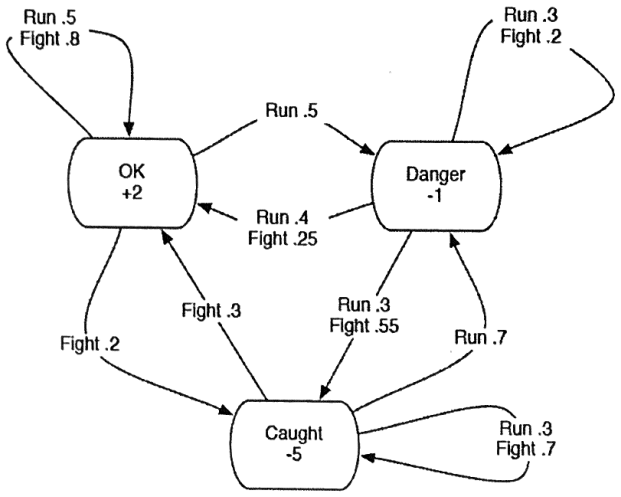



Figure 1: MDP: implausible robot

(a) (6 pts) Fill out the table with the results of value iteration at t =2 with a discount factor $\gamma = 0.9$. Note; to calculate values at t =1, I used initial values as 0.

| t | $J^t(O)$ | $J^t(D)$ | $J^t(C)$ |
|---|---|---|---|
| 1 | 2 | $-1$ | $-5$ |
| 2 | 2.54 | $-1.9$ | $-6.98$ |

$J^2(O) = \max(2 + 0.9(0.5 \times 2 + 0.5(-1)), 2 + 0.9(0.8 \times 2 + 0.2 \cdot (-5)))$
$= \max(2.45, 2.54) = 2.54$

$J^2(D) = \max(-1 + 0.9(0.4 \times 2 + 0.3(-1) + 0.3(-5)), -1 + 0.9(0.25 \times 2 + 2 \cdot (-1) + 0.75(-5)))$
$= \max(-1.9, -3.2050) = -1.9$

$J^2(C) = \max(-5 + 0.9(0.7 \cdot (-1) + 0.3 \cdot (-5)), -5 + 0.9(2 \cdot 0.3 + 0.7 \cdot (-5)))$
$= \max(-6.98, -7.61) = -6.98$

## Question 2: MDPs: Implausible robot

[13 pts] You are a wildly implausible robot who wanders among the four areas depicted below. You hate rain and get a reward of -30 on any move that starts in the deck and -40 on any move that starts in the Garden. You like parties, and you are indifferent to kitchens

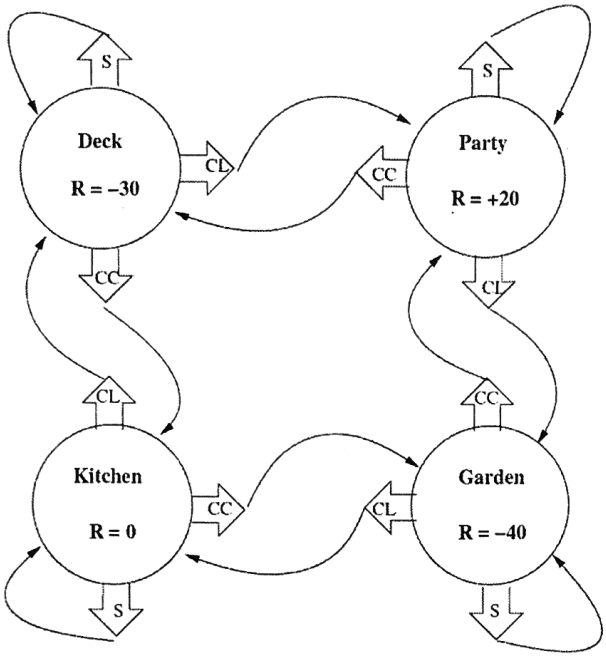



Figure 2: MDP: implausible robot

Actions: All states have three actions: Clockwise (CL), Counter-Clockwise (CC), Stay (S). Clockwise and Counter-Clockwise move you through a door into another room, and Stay keeps you in the same location. All transitions have are deterministic (probability 1.0).

(a) (3 pts) How many distinct policies are there for this MDP?

$3^4 = 81$

(b) (5 pts) Let $J^*$( Room)= expected discounted sum of future rewards assuming you start in "Room" and subsequently act optimally. Assuming a discount factor $\gamma$=0.5 , give the $J^*$ values for each room.

- the optimal policy: $\begin{cases} \pi(\text{Deck}) = CL \\ \pi(\text{Kitchen}) = S \\ \pi(\text{Garden}) = CC \\ \pi(\text{Party}) = S \end{cases}$ $\Rightarrow \begin{cases} J^*(\text{Deck}) = -30 + J^*(\text{Party})(0.5 \\ J^*(\text{Kitchen}) = 0 + J^*(\text{Kitchen})(0.5 \\ J^*(\text{Garden}) = -40 + J^*(\text{Party})(0.5 \\ J^*(\text{Party}) = 20 + J^*(\text{Party})(0.5 \end{cases}$

$\Rightarrow \begin{cases} J^*(\text{Deck}) = -10 \\ J^*(\text{Kitchen}) = 0 \\ J^*(\text{Garden}) = -20 \\ J^*(\text{Party}) = 40 \end{cases}$

(b) (6 points) At t = 2 with $\gamma = 0.9$, what policy would you select? Is it necessarily true that this is the optimal policy? At t = 3 what policy would you select? Is it necessarily true that this is the optimal policy??

From (a) :
$$\begin{cases} O \Rightarrow \text{Fight} \\ D \Rightarrow \text{Run} \\ C \Rightarrow \text{Run} \end{cases}$$

- the value iteration is not converged and is not guaranteed to find the optimal policy. So this policy is not necessarily optimal.

## Question 3: MDP: Solving Bellman Equations

[10 pts]

(a) (5 points) Suppose you have a robot trying to reach a goal and avoid cliffs in a small grid world. It can only move North, South, East, or West, but occasionally fails to move in the intended direction. If you were to model this using an MDP and were trying to solve it optimally, should you use value iteration or policy iteration? Justify your answer.

In this case, we should use value iteration because this problem have many states with only four actions. So value iteration is better than policy iteration.

(b) (5 points) Now suppose that the robot can teleport to any grid cell but the teleportation causes it to land in neighboring grid cells near the target with some probability. Of you were to model this using an MDP and were trying to solve it optimally should you use value iteration or policy iteration? Justify your answer.

In this case, we have another action named "teleport" that can transfer the robot to any grid cell but it can land in neighbors grid cell. That teleport actions can caused more actions for the robot to reach the goal state. So we use policy iteration in this case for the better outcome.

(c) (7 pts) The optimal policy when the discount factor, $\gamma$ , is small but non-zero (e.g. $\gamma = 0.1$) different from the optimal policy when is large e.g. $\gamma = 0.9$). If we began with $\gamma = 0.1$ , and then gradually increased , what would be the threshold value of above which the optimal policy would change?

- If the discount factor increase, the policy changes from S to CL in the Kitchen. this happens at the value of taking action S equal to CL.

$$\begin{cases} J^S(\text{Kitchen}) = 0 + \gamma J^S(\text{Kitchen}) = 0 \\ J^{CL}(\text{Kitchen}) = 0 + \gamma J^*(\text{Peck}) \end{cases}$$

- We have

$$J^*(\text{Peck}) = -30 + \gamma J^*(\text{Party})$$

$$J^*(\text{Party}) = 20 + \gamma J^*(\text{Party}).$$

$$\Rightarrow -30 + \gamma\left(\frac{20}{1-\gamma}\right) = 0$$

$$\Rightarrow 20\gamma = 30(1-\gamma)$$

$$\Rightarrow \gamma = \frac{3}{5} = 0.6$$