2018학년 2학기 인공지능

중간고사

학번 :

이름 :

문항	점수
1번	
2번	
3번	
4번	
5번	
6번	
7번	
8번	
9번	
합계	

성실하고 정직하게 시험에 임하겠습니다. (서명)

※ 부정행위 적발시 F학점을 부여하오니 주의하여 주시기 바랍니다.

[문제1] 다음 질문에 답하시오 (간략히는 정말 간략히입니다.)

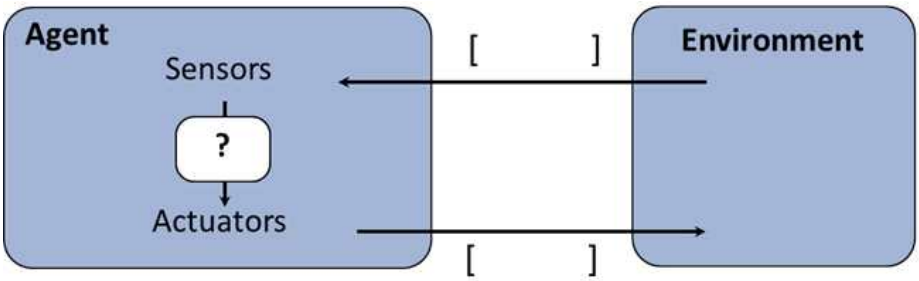
a) Rational Agent와 Reflex Agent의 차이점을 간략히 설명하시오.

b) Search Problem을 위해 정의되어야 하는 구성요소 4가지를 쓰시오.

c) Uninformed Search와 Informed Search의 차이점을 간략히 설명하시오.

d) Greedy Search와 A* Search가 있다. 두 알고리즘의 차이점을 간략히 설명하시오.

e) 그림의 빈칸 []을 채우시오. (한글, 영어 둘다 가능)



f) 다음 O/X 질문에 답하시오.

질문	O/X
(동일한 Search문제에서) DFS를 이용한 확장노드의 개수는 A* 알고리즘을 이용할 때의 확장 노드의 개수보다 항상 크거나 같다.	
휴리스틱이 모두 h(n)= 0 인 함수는 admissible한 heuristic이다.	
BFS는 UCS의 특수한 예이다.	
UCS는 A*의 특수한 예이다.	
minimax에서 Pruning(가지치기)를 하여도 최적의 솔루션을 찾을 수 있다.	

[문제2] 다음은 일반적인 Tree Search 알고리즘이다. 각 질문에 답하시오

```
function TREE-SEARCH(problem, fringe) return a solution, or failure
    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
        for child-node in EXPAND(STATE[node], problem) do
            fringe ← INSERT(child-node, fringe)
        end
    end
```

a) DFS, BFS, UCS, A* 등 거의 모든 Search 알고리즘은 위와 같이 동일한 형태를 지니지만, 서로 다르게 동작을 한다. 어느 줄(line)이 다르게 동작하게 하는지 해당 코드를 찾아 쓰시오.

b) 해당 코드(line)이 다르게 동작하기 위해서 서로 다른 자료구조와 (필요시) 서로 다른 Cost 함수를 사용하게 된다. 각 알고리즘에 맞는 내용을 빈칸에 채우시오

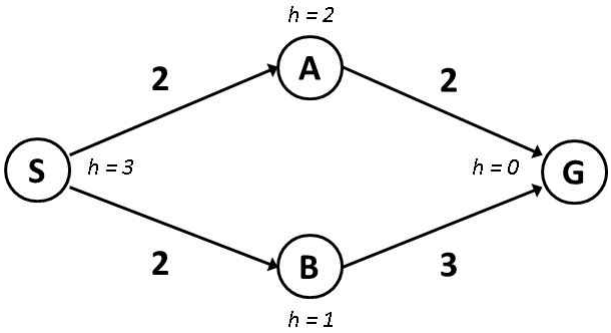
Search 알고리즘	자료구조	Cost함수 필요 여부(O/X)	Cost 함수
DFS			
BFS			
UCS			
Greedy Search			
A*			

c) 각 Tree Search함수가 Completeness와 Optimality을 만족하는지 O/X로 답하시오.

* Completeness 만족: Start State에서 Goal State로 가는 길이 있을 경우, 이를 찾을 수 있음
* Optimality 만족: Start State에서 Goal State로 가는 길이 있을 경우, 최적의 길을 찾을 수 있음

Search 함수	Completeness (O/X)	Optimality (O/X)
DFS (Depth First Search)		
BFS (Breath First Search)		
UCS (Uniform Cost Search)		
Greedy Search		
A* (A-star Search Algorithm)		

[문제3] 각 Search 알고리즘을 이용하여 S(Start)에서 G(Goal)로 가는 길을 찾을 경우, 각 알고리즘에 대해 다음 질문에 대해 답하시오. (h는 휴리스틱의 값이다)



a) BFS (동일한 경우, 알파벳 순)

큐에 들어가는 순서	
큐에서 나오는 순서	
최종 PATH는?	

b) UCS (동일한 경우, 알파벳 순)

우선순위 큐에 들어가는 순서 (우선순위 값도 표시)	
우선순위 큐에서 나오는 순서 (우선순위 값도 표시)	
최종 PATH는?	

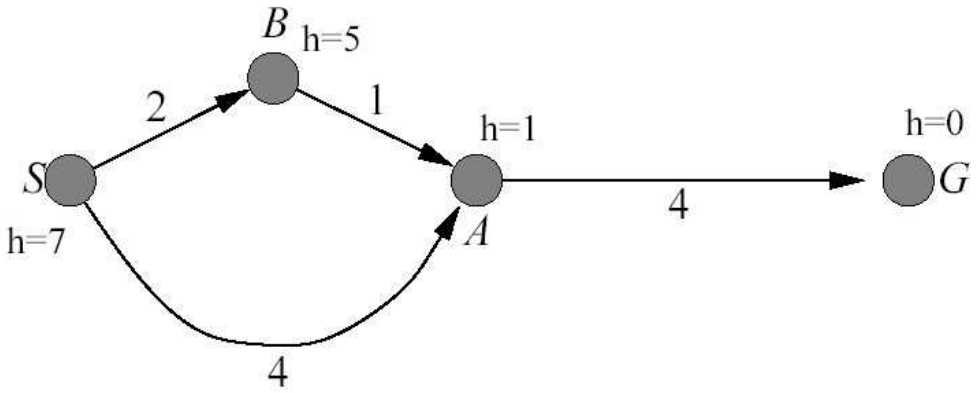
c) Greedy (동일한 경우, 알파벳 순)

우선순위 큐에 들어가는 순서 (우선순위 값도 표시)	
우선순위 큐에서 나오는 순서 (우선순위 값도 표시)	
최종 PATH는?	

d) A* (동일한 경우, 알파벳 순)

우선순위 큐에 들어가는 순서 (우선순위 값도 표시)	
우선순위 큐에서 나오는 순서 (우선순위 값도 표시)	
최종 PATH는?	

[문제4] 인공지능 초보생 유나는 자신만의 휴리스틱을 만들어 다음 그래프에 적용한 후에, A* 알고리즘(Graph 기반의 Search알고리즘 적용, Tree기반 Search 알고리즘 아님)을 통해 S로부터 G까지의 최소 거리가 되는 PATH를 찾으려 하고 있다.



a) A* 알고리즘을 수행하고 그 결과를 쓰시오.

우선순위 큐에 들어가는 순서 (우선순위 값도 표시)	
우선순위 큐에서 나오는 순서 (우선순위 값도 표시)	
최종 PATH는?	

b) 해당 Heuristic은 Admissible한가? 맞다면 이유를 설명하고, 아니면 Admissible하게 고치시오

c) 해당 Heuristic은 Consistency한가? 맞다면 이유를 설명하고, 아니면 Consistency하게 고치시오

[문제5] 다음은 minimax, expectimax 함수의 일부다. 코드를 완성하시오. (4군데, v = ...)

```
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v =
    return v
```

⇔

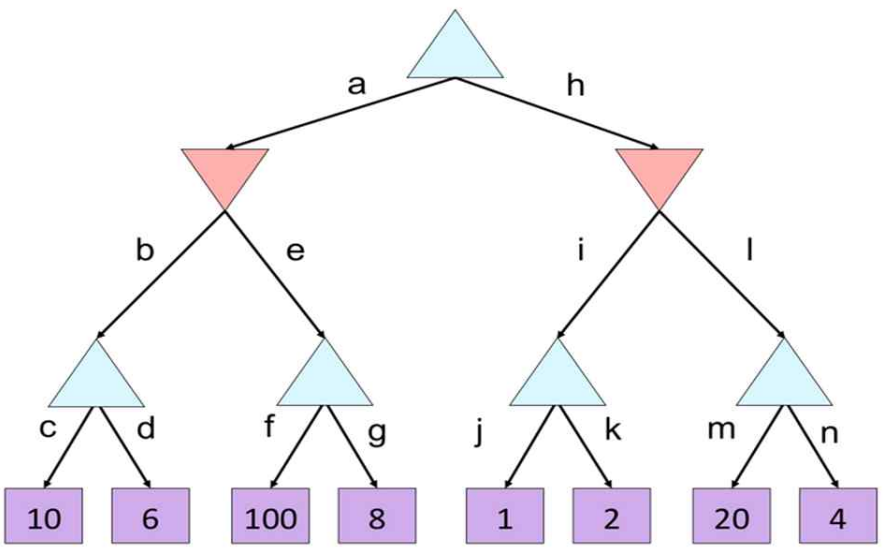
```
def min-value(state):
    initialize v = +∞
    for each successor of state:
        v =
    return v
```

```
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v =
    return v
```

⇔

```
def exp-value(state):
    initialize v = 0
    for each successor of state:
        p = probability(successor)
        v =
    return v
```

[문제6] 다음은 minimax 알고리즘에서 pruning을 적용하였을 때의 아래그림에서 Evaluation이 필요없는 연결선에 X로 표시하시요. (파란색 일반삼각형이 MAX이며, 적색 역삼각형이 MIN이다. Evaluation은 왼쪽에서부터 오른쪽으로 이루어진다.)

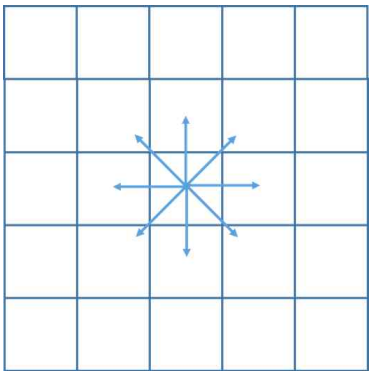


[문제 7] Agent는 아래 그림과 같이 8방향(동,서,남,북,동남,남서,서북,북동)으로 움직임이 가능하다. 하나의 블락에서 다음 블락으로 움직일 경우 그 비용은 두 점사이의 직선거리인 Euclidean Distance이다.
이와 같은 환경에서 우리는 A* 알고리즘을 사용하기 위해 Goal까지의 거리를 계산하는 휴리스틱으로 현재 위치와 Goal까지의 격자거리인 Manhattan Distance를 이용하고자 한다.

[참고] 점(x1, y1)에서 목적지(x*, y*)까지의 거리 계산

Euclidean Distance = $\sqrt{(x1 - x^*)^2 + (y1 - y^*)^2}$

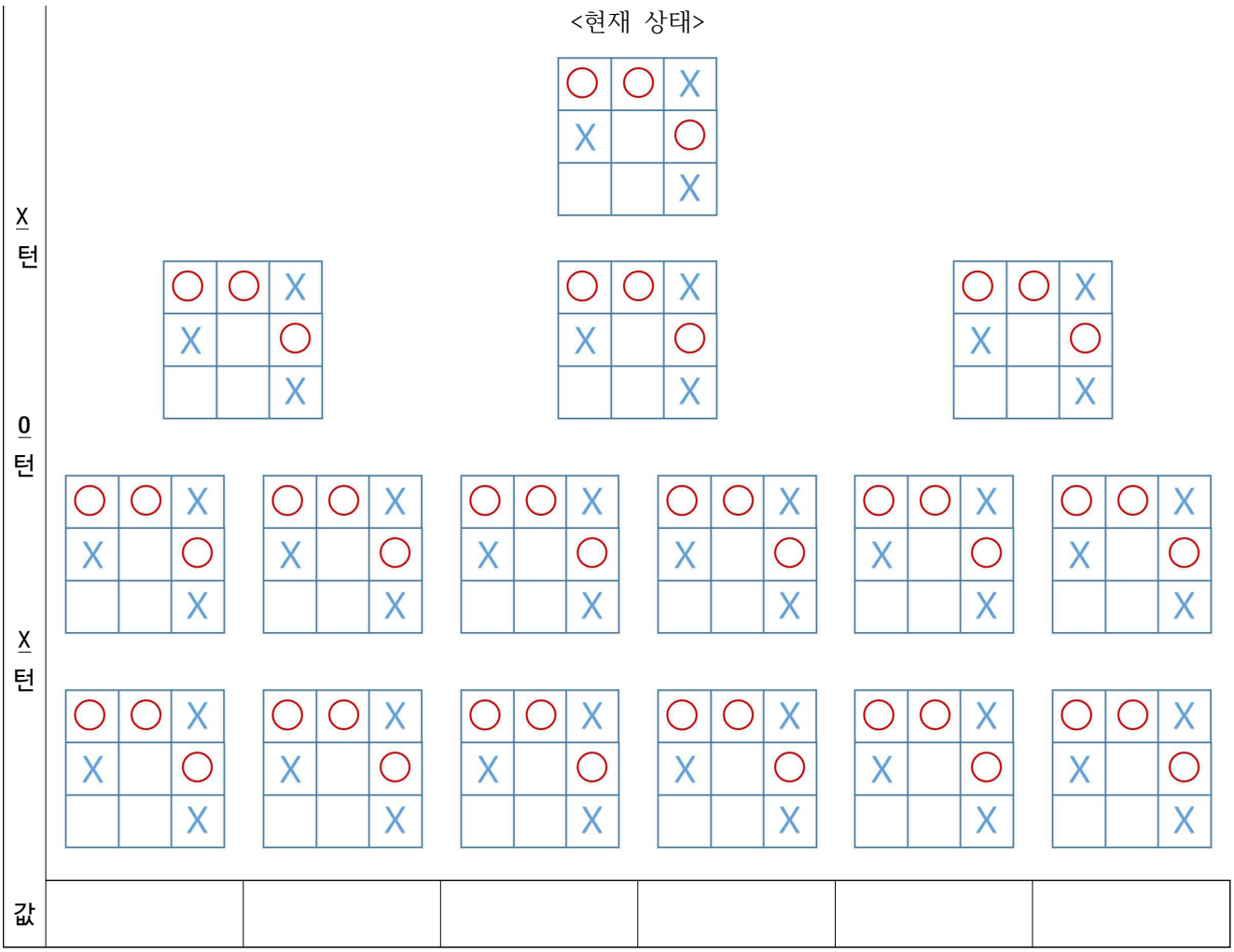
Manhattan Distance = $|x1 - x^*| + |y1 - y^*|$



위에 설명한 Manhattan Distance 휴리스틱이 Admissible한지 증명하시오.

[문제 8] Tic-Tac-Toe의 한 상태이다. 다음에 둘 차례는 X 이다. (즉, X의 관점에서 생각)

a) 아래 그림은 각 턴의 변화를 나타낸 그래프이다. 빈 칸을 채우시오



b) 모든 칸이 다 채워지게 되면 승부가 판가름이 나고, 그에 따라서 해당 terminal node의 값 (value)을 결정할 수 있게 된다. X의 관점에서 이기면 1점, 비기면 0점, 지면 –1점이 된다. (* 3개의 X가 직선/대각선을 이루면 이기는 게임) 위의 그림에서 값의 빈칸을 채우시오

c) Minimax 알고리즘을 돌려 중간노드의 값을 계산하시오. (각 블록의 왼쪽 위에 값을 쓰시오)

d) Alpha-Beta Pruning적용 시 O 턴의 레벨에서 총 6개 중 몇 개가 Evaluation이 필요없는지 쓰시오.

만약, 값의 범위 정보가 없어서 노드 값의 범위를 [-∞, +∞]로 가정한 경우,

- 맨 왼쪽부터 오른쪽 방향으로 평가하였을 경우 : 개
- 맨 오른쪽부터 왼쪽 방향으로 평가하였을 경우 : 개

만약, 값의 범위 정보가 있어서 노드 값의 범위가 –1, 0, 1 로 단 3가지로 가정한 경우,

- 맨 왼쪽부터 오른쪽 방향으로 평가하였을 경우 : 개
- 맨 오른쪽부터 왼쪽 방향으로 평가하였을 경우 : 개

[문제9] 아래는 [문제2]에서 주어진 Tree Search를 위한 알고리즘이다. Graph상에서 Search를 위해서는 중복된 노드 방문을 막아야 하는데, 아래 코드는 그러하지 못하다. 해당 코드를 변형하여 Graph Search에서 중복 노드 방문을 방지하도록 코드를 구현하시오.
(아래 기호보다 python코드가 용이하면, 과제에서 작성하였던 python코드를 적으셔도 답으로 인정됩니다.)

```
function TREE-SEARCH(problem, fringe) return a solution, or failure
    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
        for child-node in EXPAND(STATE[node], problem) do
            fringe ← INSERT(child-node, fringe)
        end
    end
```