CS410: Artificial Intelligence 2020 Fall

Homework 1: Search Algorithms

Chunyu Xue 518021910698

Problem 1:

(1. a) (1) The representation of the state space is:

Agent states s_a: 23 × 2 = 46

Ghost states s_g: 7

Key states s_k: 2

Therefore, the state space could be represented as:

$$S_k = (s_a, s_g, s_k)$$

(2) The size of state space is:

$$size = s_a \times s_g \times s_k = 46 \times 7 \times 2$$

(3) Agent states (s_a) mean the combination of agent positions (totally 23 positions) and agent directions (left or right). Ghost states (s_g) mean the positions of the ghost (totally 7 positions). Key states mean whether the key is reached by the agent (0 for not reached, 1 for already reached).

(1. b)

Algorithm 1 getSuccessor(state)

```
successors ← empty list
for action ∈ {left, left_up, left_down, right, right_up, right_down} do
    if action ∈ {left,left_up,left_down} and state.facing_direction == right then
        continue
    end if
    if action ∈ {right,right_up,right_down} and state.facing_direction == left then
        continue
    end if
    if action ∈ {turn_right} and state.facing_direction == right then
        continue
    end if
    if action ∈ {turn_left} and state.facing_direction == left then
        continue
    end if
    next_state ← state.apply_action(action)
    if next_state not out of bound and next_state not in successors and next_state in state space then
        successors.append(next_state)
    end if
end for
return successors
```

(1. c)

The Goal test of this problem is that the agent (originally in cell (1, 3)) reaches the key in cell (3, 0) without crushing on the skull walking around on the last line.

The goal state is:

$$S_g = (s'_a, s_g, 1)$$

In which s'_a means that the agent arrives cell (3, 0), and the direction is either left or right.

Problem 2:

(2. a) (1) The order of nodes expanded:

C - A - B - D - E - F - G - H

(2) Number of nodes expanded: 8

(3) Path returned:

C - E - H

(4) Length of path: 2

(5) Cost of path: 4

(6)

State s	A	B	C	D	E	F	G	H
$b(s)$	1	1	0	1	1	1	2	2
length from C to s	1	1	0	1	1	1	2	2
cost from C to s	1	3	0	1	1	3	2	4

(2. b) (1) Order of nodes expanded:

C - A - B - D - E - F - G - H

(2) Number of nodes expanded: 8

(3) Path returned: C - A - B - D - E - F - G - H

(4) Length of path: 7

(5) Cost of path: 11

(6)

State s	A	B	C	D	E	F	G	H
length from C to s	1	2	0	3	4	5	6	7
cost from C to s	1	2	0	3	6	7	10	11

(2. c) (1) Order of nodes expanded:

C - A - D - E - B - F - G - H

(2) Number of nodes expanded: 8

(3) Path returned: C - D - F - H

(4) Length of path: 3

(5) Cost of path: 3

(6)

State s	A	B	C	D	E	F	G	H
$c(s)$	1	2	0	1	1	2	2	3

(2. d) (1) Order of nodes expanded:

C - A - D - E - B - F - G - H

(2) Number of nodes expanded: 8

(3) Path returned:

C - D - F - H

(4) Length of path: 3

(5) Cost of path: 3

(6)

State s	A	B	C	D	E	F	G	H
$f(s)$	5	5	3	3	3	3	3	3

Problem 3:

In general, DFS obtains the worst path (path length is 7, cost is 11), next is BFS (path length is 2, cost is 4), UCS and A* are both optimal (path length is 3, cost is 3).

BFS finds the shortest path. UCS and A* algorithms find the optimal path.

Problem 4:

(4. a)

State s	A	B	C	D	E	F	G	H
Heuristic $h(s)$	3	3	2	2	1	1	0	0

Explanation:

a) Admissible: For any state s, we all have h(s) ≤ actual cost from s to H. For example, the cost from A to H is 4, which is larger than h(A) = 3.

b) Consistent: for any node pair <u, v>, we all have h(u) - h(v) ≤ actual cost between u and v. For example, node pair <A, B> has a cost of 1, which is no less than h(A) - h(B) = 0.

Therefore, this type of heuristics satisfies both admissible and consistent.

(4. b)

State s	A	B	C	D	E	F	G	H
Heuristic $h(s)$	0	3	2	2	1	1	0	0

Explanation:

a) Admissible: We can prove admissible property as (4. a) shown.

b) Not consistent: Since there exists node pair <A, C> who has a cost of 1, which is less than h(C) - h(A) = 2, the consistent property is not valid.

(4. c)

State s	A	B	C	D	E	F	G	H
Heuristic $h(s)$	0	3	2	2	1	2	0	0

Explanation:

a) Not admissible: Since the cost from state F to H is 1, which is smaller than h(B) = 2, the admissible property is not valid.

b) Not consistent: Since there exists node pair <A, C> who has a cost of 1, which is less than h(C) - h(A) = 2, the consistent property is not valid.

Problem 5:

(5. a)

(5. a. i) The original state space S is :

$$S = \{(x, y)|x, y \in \{A, B, C, D, E, F, G, H\}\ (x \neq H)\ and\ x, y\ are\ adjacent\ or\ digonal\}$$

After modification, the current state space S' is:

$$S' = S\ \cup\ \{(x, y)|x, y \in \{A, ...G\}\ and\ x, y\ are\ not\ adjacent\}$$

Notice that in each game, only one 'teleportation' can be used.

(5. a. ii) Yes, Nancy is right. The example is as follows:

State s	A	B	C	D	E	F	G	H
Heuristic $h(s)$	3	3	2	2	1	0	0	0

Explanation:

In old game rule, for any node pair <u, v>, we all have h(u) - h(v) ≤ actual cost between u and v. For example, node pair <C, B> has a cost of 1, which is no less than h(A) - h(B) = 1. So it satisfies the consistent property (also satisfies admissible property).

In this new game, there exists a node pair <C, F>, which can cost only 1 by operating 'teleportation', which is smaller than h(C) - h(F) = 2.

Therefore, consistent property is no larger valid in this case.

(5. b) We assume that if we encounter with a skull on our way, we need to wait for a cycle until the skull go to another cell, let the cost of wait be 1.

(5. b. i) The original state space S is :

$$S = \{(x, y)|x, y \in \{A, B, C, D, E, F, G, H\}\ (x \neq H)\ and\ x, y\ are\ adjacent\ or\ digonal\}$$

After modification, the current state space S'' is:

$$S'' = S\ \setminus \{(x, y)|x, y \in E, F\ and\ skull\ is\ in\ E\ or\ F\ when\ chosen\}$$

Notice that we need to abandon those cases that the skull hinder our way.

(5. b. ii) No, Ethan is wrong. The proof is as follows:

Proof:

Assume a consistent assignment that in old game rule, for any node pair <u, v>, we all have h(u) - h(v) ≤ actual cost between u and v.

In this new game, if we add a skull into the board, the only function that it can bring is hindering the agent from arriving at the key, which means that the cost from u to v (for any node pair <u, v>) will be more or not changed.

Therefore, for any node pair <u, v>, we have:

$$h(u) - h(v) \leq original\ cost \leq current\ cost$$

Thus, the consistent property is still valid.