Foundations of AI HW #1 B08902016 蔡旭樓

Problem 1.

(a) Depth-first search

Expansion Order : Start → A → C → D → B → Goal

Final Path : Start → A → C → D → Goal

(b) Breadth-first search

Expansion Order : Start → A → B → D → C → Goal

Final Path : Start → D → Goal

(c) Uniform-cost search

Expansion Order : Start → A → B → D → C → Goal

Final Path : Start → A → C → Goal

(d) Greedy search

Expansion Order : Start → D → Goal

Final Path : Start → D → Goal

(e) A^* search

Expansion Order : Start → A → D → B → C → Goal

Final Path : Start → A → C → Goal

Problem 2.

Suppose the heuristic overestimate at most ε times the actual cost to the goal, $\varepsilon > 1$.

① If A^* search returns a path with the cost S', and the cost of the optimal path is S, where $S' > \varepsilon \cdot S$.

② Suppose the goal state of the path returned by A^* is A', and the goal state of the optimal path is A.
(A may be equal to A')

Then, at the moment when A' is in the fringe, there must be one state on the optimal path also in the fringe.
We call it P (P may equal to A)

③ At that moment,

$$f(P) = g(P) + h(P) \le g^*(P) + \varepsilon \cdot dist(P, A) \le \varepsilon \cdot (g^*(P) + dist(P,A))$$
$$\le \varepsilon \cdot S$$

$$f(A') = g(A') + h(A') = g(A') = S' > \varepsilon \cdot S$$

$$\therefore f(P) < f(A')$$

And for every state P on the optimal path, $f(P) < f(A')$,
So, A^* search will not return the path(Start $\rightarrow A'$)

④ By Contradiction, we know that the path A^* return whose cost is smaller or equal to $\varepsilon \cdot S$ (S = the cost of optimal path)
#

Problem 3.

This implementation executed "ADD successor to closed" when discover the successors of each expanding node.

However, this will not make sure the result is correct.
The correct implementation is move "ADD successor to closed" to the part when we extract (expand) node from fringe.

At that time, "ADD the expanding node to closed" will make sure the result is correct.
#

Problem 4.

(1) CSP

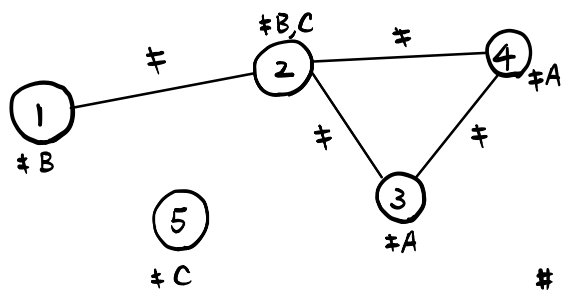
Variables : Class 1 to 5

Domain : Instructor A, B, C

Constraints :

1. Class 2 $\neq$ Class 3 $\neq$ Class 4
2. Class 1 $\neq$ Class 2
3. Class 1 $\neq$ Instructor B
4. Class 2 $\neq$ Instructor B, C
5. Class 3 $\neq$ Instructor A
6. Class 4 $\neq$ Instructor A
7. Class 5 $\neq$ Instructor C #

(2)



#

(3)

Class	1	2	3	4	5
Domain	C	A	BC	BC	AB

#

(4) Example Solution

Class	1	2	3	4	5
Instructor	C	A	B	C	A

#

(5)

$\because$ Tree-Structured CSP needs only $O(nd^2)$-time to solve, which is much faster than the orignal $O(d^n)$-time, where d= size of domain, n= number of variables.