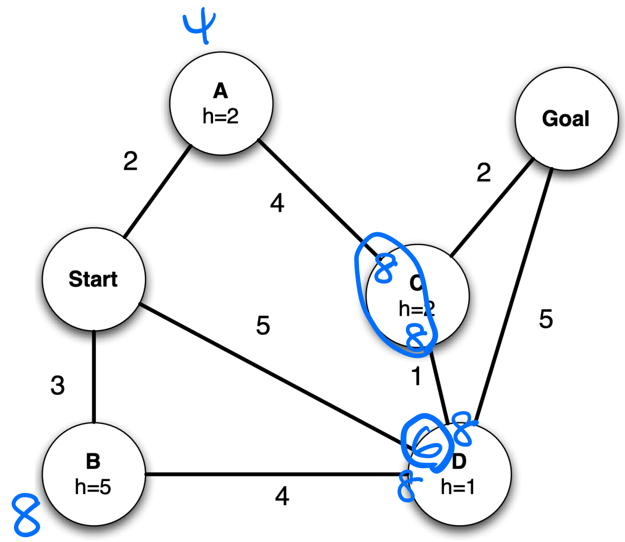
Due: 03/26/2024 23:59

Foundations of Artificial Intelligence: Homework 1

Instructor: Shang-Tse Chen & Yun-Nung Chen

Problem 1

(10 points)



Write down the order of state expansion and the final path returned by each of the graph search (as oppose to tree search) algorithms below. You can assume ties are resolved alphabetically.

a) Depth-first search.

State expansion: Start → A → C → D → B → Goal

Final path : Start → A → C → D → Goal

b) Breadth-first search.

State expansion: Start → A → B → D → C → Goal

Final path : Start → D → Goal

c) Uniform cost search.

State expansion: Start → A → B → C → D → Goal

Final path : Start → A → C → Goal

d) Greedy search with the heuristic h shown on the graph.

State expansion: Start → D → Goal

Final path : Start → D → Goal

e) A^* search with the same heuristic.

State expansion: Start → A → D → C → Goal

Final path : Start → A → C → Goal

Problem 2 (10 points)

function A* GRAPH SEARCH(*problem*)
    fringe ← an empty priority queue
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[*problem*]), *fringe*)
    closed ← an empty set
    ADD INITIAL-STATE[*problem*] to *closed*
while **loop**
        if *fringe* is empty **then**
            return failure
        end if
        node ← REMOVE-FRONT(*fringe*)
        if GOAL-TEST(*problem*, STATE[*node*]) **then**
            return *node*
        end if   s_cost ← PATH-COST(node) + STEP-COST(node, successor, problem)
if STATE[node] not in closed then ADD STATE[node] to closed
        for *successor* in GETSUCCESSORS(*problem*, STATE[*node*]) **do**
            if *successor* not in *closed* ~~**then**~~ or s_cost < PATH-COST(successor) then
                ADD *successor* to *closed*
                fringe ← INSERT(MAKE-SUCCESSOR-NODE(*successor*, *node*), *fringe*)
            end if   order by PATH-COST(successor)+heuristic(successor)
        end for
    end loop
end function

The implementation of the A^* graph search algorithm above is incorrect. Briefly explain the bug in this implementation and justify your answer.

在加入 successor 進 closed set 之前，要考慮的條件有2個:
1. successor 尚未在 closed set (這邊題目有)
2. 新找到的 successor path cost 若小於原本的，也要把這個 successor 狀態更新後加入 closed set. (這邊要做處理) 請看上面

另外，沒有提 PQ 如何處理節點的總估計成本，即 $f(n) = g(n)+h(n)$

Problem 3 (10 points)

You are scheduling for 5 classes on the same day taught by 3 instructors. Of course, each instructor can only teach one class at a time.

The classes are:

- Class 1 - Intro to Programming: 8:00-9:00am
- Class 2 - Intro to Artificial Intelligence: 8:30-9:30am
- Class 3 - Natural Language Processing: 9:00-10:00am
- Class 4 - Computer Vision: 9:00-10:00am
- Class 5 - Machine Learning: 10:30-11:30am

The instructors are:

- Instructor A - Can teach Classes 1, 2, and 5.
- Instructor B - Can teach Classes 3, 4, and 5.
- Instructor C - Can teach Classes 1, 3, and 4.

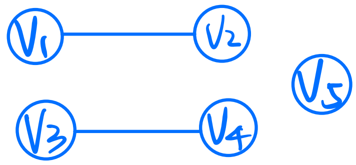
	1	2	3	4	5
8:00					
8:30	A,C				
9:00		A			
9:30			B,C	B,C	
10:00					
10:30					
					A,B
11:30					
A	✓	✓			✓
B			✓	✓	✓
C	✓		✓	✓	

(1) Formulate this problem as a CSP. Describe the variables, domains and constraints.

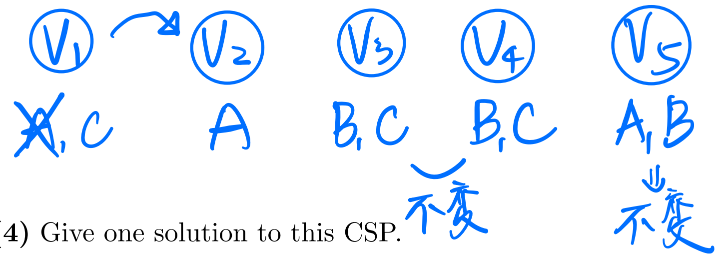
Variables: V_i 表示第i堂課的Instructor, $i \in \{1, 2, 3, 4, 5\}$

Domains: $\begin{cases} D(V_1) = \{A, C\} & D(V_4) = \{B, C\} \\ D(V_2) = \{A\} & \\ D(V_3) = \{B, C\} & D(V_5) = \{A, B\} \end{cases}$ Constraints: $\begin{cases} V_1 \neq V_2 \\ V_3 \neq V_4 \end{cases}$

(2) Draw the constraint graph associated with your CSP.



(3) Show the domains of the variables after running arc-consistency on this initial graph (after having already enforced any unary constraints).



$\Rightarrow \begin{cases} D(V_1) = \{C\} \\ D(V_2) = \{A\} \\ D(V_3) = \{B, C\} \\ D(V_4) = \{B, C\} \\ D(V_5) = \{A, B\} \end{cases}$

(4) Give one solution to this CSP.

$D(V_1) = \{C\}$ $D(V_3) = \{B\}$

$D(V_2) = \{A\}$ $D(V_4) = \{C\}$

$D(V_5) = \{A\}$

(5) Your CSP should look nearly tree-structured. Briefly explain (one sentence or less) why we might prefer to solve tree-structures CSPs.

因為樹狀結構CSP可以透過backtrack，
使得複雜度從$O(d^n)$降為線性時間$O(nd^2)$，
大幅減少了計算複雜性。

Problem 4 (10 points)

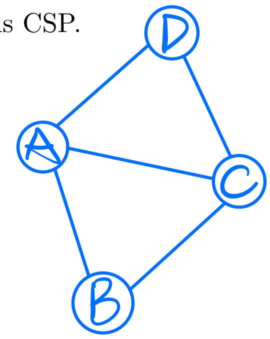
A B C D p q r s

Alice, Bob, Chris, and David are ordering food from pizza, quesadillas, ramen, and sushi. They have some strict preferences:

1. Chris will not order sushi. $D(C): \{p, q, r\}$
2. Alice and Bob want to order different food. $D(A) \neq D(B)$
3. Bob will only order pizza or ramen. $D(B): \{p, r\}$
4. Alice and Chris want to order the same dish as each other but different from the remaining two people.
5. David will not order quesadillas. $D(D): \{p, r, s\}$

$D(A) = D(C) \neq D(B)$
$\neq D(D)$

a) Draw the constraint graph for this CSP.



b) Run the basic backtracking search. Use alphabetical order to both select unassigned variables and iterate over values. Write down the food assignment.

A: p → B: r → C: p → D: r, s
q → B: p, r → C: q → D: p, r, s
r → B: p → C: r → D: p, s
s → B: p, r → C: s → D: p, r

挑一種:

A → B → C → D
p r p s

c) Assume that no variables have been assigned values yet. When running one iteration of forward checking, which value(s) will be removed for each variable if we assign "pizza" to Alice. Write down "None" if no values will be removed.

A → B → C → D
p r p
None (remove p) None None