

# Q1. [15 pts] CSP: Air Traffic Control

We have five planes: A, B, C, D, and E and two runways: international and domestic. We would like to schedule a time slot and runway for each aircraft to **either** land or take off. We have four time slots:  $\{1, 2, 3, 4\}$  for each runway, during which we can schedule a landing or take off of a plane. We must find an assignment that meets the following constraints:

- Plane B has lost an engine and must land in time slot 1.
  - Plane D can only arrive at the airport to land during or after time slot 3.
  - Plane A is running low on fuel but can last until at most time slot 2.
  - Plane D must land before plane C takes off, because some passengers must transfer from D to C.
  - No two aircrafts can reserve the same time slot for the same runway.
- (a) [3 pts] Complete the formulation of this problem as a CSP in terms of variables, domains, and constraints (both unary and binary). Constraints should be expressed implicitly using mathematical or logical notation rather than with words.

**Variables:** A, B, C, D, E for each plane.

**Domains:** \_\_\_\_\_

**Constraints:** (you do not have to use all lines)

---

---

---

---



---

---

---

---

- (b) For the following subparts, we add the following two constraints:

- Planes A, B, and C cater to international flights and can only use the international runway.
- Planes D and E cater to domestic flights and can only use the domestic runway.

- (i) [2 pts] With the addition of the two constraints above, we completely reformulate the CSP. You are given the variables and domains of the new formulation. Complete the constraint graph for this problem given the original constraints and the two added ones.

**Variables:** A, B, C, D, E for each plane.

**Constraint Graph:**

**Domains:**  $\{1, 2, 3, 4\}$



- (ii) [4 pts] What are the domains of the variables after enforcing arc-consistency? Begin by enforcing unary constraints. (Cross out values that are no longer in the domain.)

A	1	2	3	4
B	1	2	3	4
C	1	2	3	4
D	1	2	3	4
E	1	2	3	4

- (iii) [4 pts] Arc-consistency can be rather expensive to enforce, and we believe that we can obtain faster solutions using only **forward-checking** on our variable assignments. Using the Minimum Remaining Values heuristic, perform backtracking search on the graph, breaking ties by picking lower values and characters first. List the *(variable, assignment)* pairs in the order they occur (including the assignments that are reverted upon reaching a dead end). Enforce unary constraints before starting the search.

(You don't have to use this table, it won't be graded.)

A	1	2	3	4
B	1	2	3	4
C	1	2	3	4
D	1	2	3	4
E	1	2	3	4

**Answer:**

- (c) [2 pts] Suppose we have just one runway and  $n$  planes, where no two planes can use the runway at once. We are assured that the constraint graph will always be tree-structured and that a solution exists. What is the runtime complexity in terms of the number of planes,  $n$ , of a CSP solver that runs arc-consistency and then assigns variables in a topological ordering?

- ☐  $O(1)$
- ☐  $O(n)$
- ☐  $O(n^2)$
- ☐  $O(n^3)$
- ☐  $O(n^n)$
- ☐ None of the Above

### Q3. [14 pts] State Representations and State Spaces

For each part, state the size of a minimal state space for the problem. Give your answer as an expression that references problem variables. Below each term, state what information it encodes. For example, you could write  $2 \times MN$  and write “whether a power pellet is in effect” under the 2 and “Pacman’s position” under the  $MN$ . State spaces which are complete but not minimal will receive partial credit.

Each part is independent. A maze has **height**  $M$  and **width**  $N$ . A Pacman can move *NORTH*, *SOUTH*, *EAST*, or *WEST*. There is initially a **pellet in every position** of the maze. The **goal is to eat all of the pellets**.

(a) [4 pts] Personal Space

In this part, there are  $P$  Pacmen, numbered  $1, \dots, P$ . Their turns cycle so Pacman 1 moves, then Pacman 2 moves, and so on. Pacman 1 moves again after Pacman  $P$ . Any time two Pacmen enter adjacent positions, the one with the lower number dies and is removed from the maze.

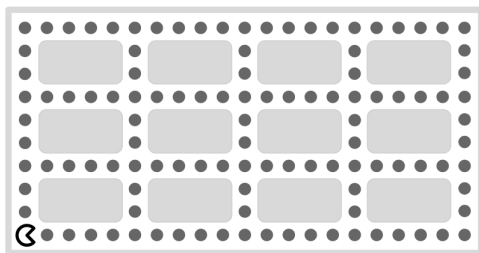
**Answer:**

(b) [4 pts] Road Not Taken

In this part, there is one Pacman. Whenever Pacman enters a position which he has visited previously, the maze is reset – each position gets refilled with food and the “visited status” of each position is reset as well.

**Answer:**

(c) [6 pts] Hallways



In this part, there is one Pacman. The walls are arranged such that they create a grid of  $H$  hallways **total**, which connect at  $I$  intersections. (In the example above,  $H = 9$  and  $I = 20$ ). In a single action, Pacman can move from one intersection into an adjacent intersection, eating all the dots along the way. Your answer should only depend on  $I$  and  $H$ .

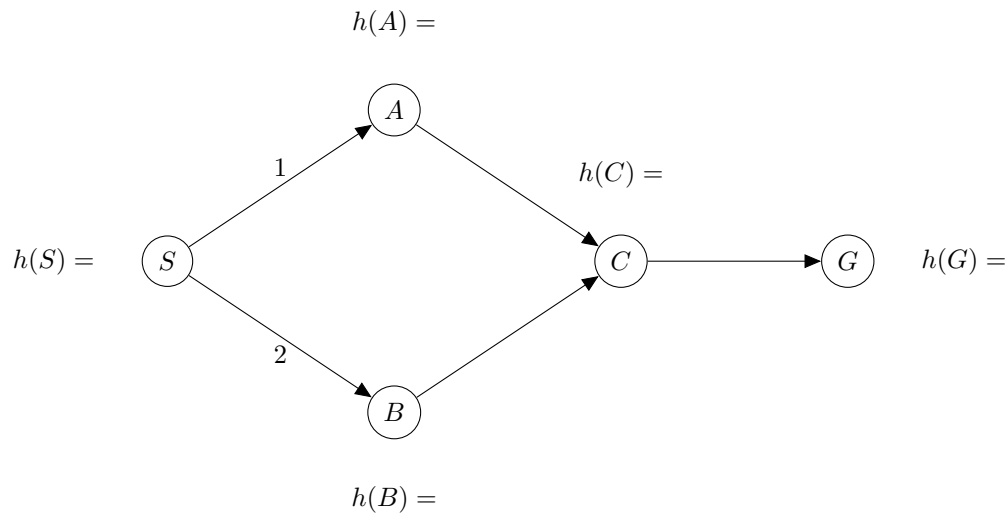
(note:  $H$  = number of vertical hallways + number of horizontal hallways)

**Answer:**

## Q4. [14 pts] Informed Search and Heuristics

(a) [6 pts] Consider the state space shown below, with starting state  $S$  and goal state  $G$ . Fill in a cost from the set  $\{1, 2\}$  **for each blank edge** and a heuristic value from the set  $\{0, 1, 2, 3\}$  **for each node** such that the following properties are satisfied:

- The heuristic is admissible but not consistent.
- The heuristic is monotonic non-increasing along paths from the start state to the goal state.
- A\* graph search finds a suboptimal solution.
- You will never encounter ties (two elements in the fringe with the same priority) during execution of A\*.



(b) [8 pts] **Don't spend all your time on this question.** As we saw in class, A\* graph search with a consistent heuristic will always find an optimal solution when run on a problem with a finite state space. However, if we turn to problems with infinite state spaces, this property may no longer hold. Your task in this question is to provide a concrete example with an infinite state space where A\* graph search fails to terminate or fails to find an optimal solution.

Specifically, you should describe the state space, the starting state, the goal state, the heuristic value at **each node**, and the cost of **each transition**. Your heuristic should be consistent, and all step costs should be strictly greater than zero ( $cost \in \mathbb{R}_{>0}$ ) to avoid trivial paths with zero cost. To keep things simple, each state should have a finite number of successors, and the goal state should be reachable in a finite number of actions from the starting state.

You may want to start by drawing a diagram.