# CS471 Assignment 1

Due date: Friday, February 6, 2025 (11:59 PM Eastern Time)

This assignment includes both written exercises and a programming component. Please follow the submission instructions carefully. The "written" portion of the assignment must be typeset with LaTeX using the provided template.

## Part 1: Written Assignment (40 pts)

1. (21 pts) **Search Problem Formulation** — Autonomous Warehouse Inspection

   *Pro-tip: Take a pencil/paper and write down the necessary information as you read!*

   Consider an autonomous warehouse inspection robot operating inside a large logistics warehouse. The robot must navigate the warehouse floor, inspect different categories of storage racks, and return to a fixed charging dock $l_{\text{dock}}$. There are 9 distinct rack categories, indexed from type '1' to type '9'. We want to formulate a search problem that plans the robot's inspection mission.

   Below is the environment's description. Please note that uppercase symbols (e.g., $L_c$) denote state variables, while lowercase symbols (e.g., $l_{\text{dock}}$) denote values that those variables may take.

   - **Power system.** The robot operates on a rechargeable battery with a maximum capacity of 12 energy units. The battery is recharged via an onboard solar charging system and can gain energy at any location (assume it's always daylight). One minute of exposure to daylight restores one unit of battery energy. While charging, the robot must remain idle and cannot perform any other actions.
   - **Warehouse map.** The warehouse floor is discretized into $15 \times 15$ grid cells $L$. For each rack cell $l \in L$, the warehouse map specifies the following two cell properties: **(i)** the type of racks in that cell $R(l)$ (assume at most one category per cell), where $R(l) \in \{0, 1, \ldots 9\}$. $R(l) = 0$ indicates there is no racks in cell $l$. **(ii)** the inspection effort $W(l)$, representing the physical load required to inspect a representative rack of that category. The rack type $R(l)$ and effort $W(l)$ are known for every $l \in L$ in this environment.
   - **Navigation routes.** On the map, between any two locations $l_1$ and $l_2$ (either $l_{\text{dock}}$ or a rack-containing cell), a route $r$ with $\text{EndPoint}(r) = \{l_1, l_2\}$ is provided, along with the distance of that route $d_r$, the number of units of battery charge required $b_r$, and the amount of time (in minutes) $t_r$ needed to complete that route. You may assume all routes on the map are known and satisfy the triangle inequality.
   - **Rack Inspection.** The robot may traverse any available route. Upon arriving at a rack-containing cell, the robot may spend 1 unit of battery energy to inspect one representative rack of that rack category. Inspecting a rack of a given category certifies that category, and no further inspections of that category are required. The time required for inspection is negligible and can be ignored.

   You are given a tradeoff parameter $\alpha$ that converts weight units to distance units (i.e., one unit of inspection effort is equivalent to $\alpha$ units of distance). The objective is to minimize the total operational cost, defined as:

   $$\text{Total distance traveled} + \alpha \times \text{Total inspection effort}$$

**Requirement:** The robot must: **(a)** inspect one representative rack from each of 9 distinct rack categories **(b)** within 3 hours (180 minutes), and **(c)** then return to the charging dock. The robot starts at the dock with a full battery.

Here is a list of variables that might be used to describe the robot's world:

- Inspection categories already completed $X = [X_1, \ldots, X_9]$ (i.e., $X_i = 1$ if category i has been inspected)
- Current robot location (cell on map) $L_c$
- Type of rack at current location $R_c$
- Inspection effort at the current location $W_c$
- Total number of inspection categories completed $N_{\text{total}}$
- Time since last charged (in minutes) $T_{\text{charged}}$
- time since departure from docker (in minutes) $T_{\text{operation}}$
- Current battery charge level $B_{\text{level}}$
- Maximum battery capacity $B_{\text{capacity}}$
- Distance from the current location to dock $D_{\text{dock}}$
- Total accumulated inspection effort $W_{\text{total}}$

(a) (4 pts) From the list of variables above, pick a **smallest subset** to form an efficient and sufficient state representation $s = [V_1, \ldots, V_k]$ (each $V_i$ is a variable that you select into your state representation). This means that you should not include any extraneous variables unless they are absolutely necessary, but the variables you select must also be sufficient for you to solve the search problem.

(b) (6 pts) Specify the goal test. That is, given a state $s$, define when $G(s) = \text{True}$.

(c) (6 pts) Specify the set of actions $a$ available to the robot and define the transition model for each action. Your transition model should specify:

  i. when action $a$ is applicable, and
  ii. if $a$ is applicable at a state $s$, what the resulting state $s' = \text{Result}(s, a)$ is after taking the action.

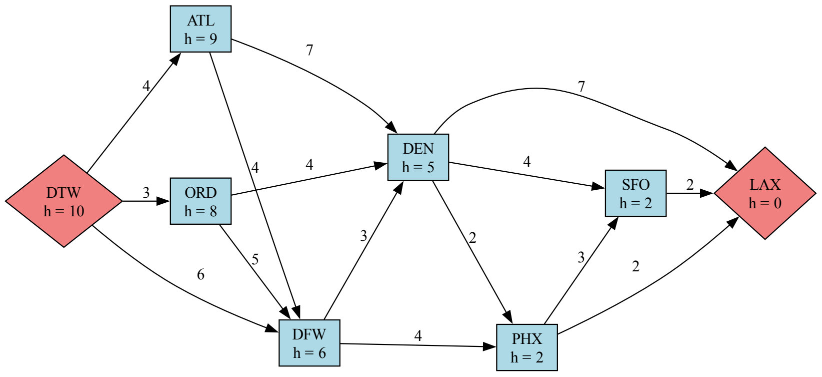
  **Define the values for the current state variables and then specify the updated values after the transition.** For example, at the current state $s = [v_1, v_2]$, after action $a = a_1$, the new state $s' = [v_1 + 1, v_2 - 1]$.
  *(Hint: for actions related to robot movement, you only need to define a generic action for all route r.)*

(d) (3 pts) Specify $\text{ActionCost}(s, a, s')$ that determines the cost of each action $a$ defined in part (c), i.e., the cost of taking action $a$ to move from state $s$ to state $s'$.

(e) (2 pts) Define an admissible heuristic function for this search problem. Justify why it is admissible.

2. (19 pts) **Search Problem Algorithms** — Airline Route Planning

   An airline must schedule a route from Detroit (DTW) to Los Angeles (LAX). Due to air-traffic flow and slot constraints, flights are one-way, and each directed edge has an associated cost. The number shown along an edge represents the cost of going from the starting node to the ending node of that edge. The value $h$ shown at each node represents a provided heuristic function that estimates the remaining cost from that node to the destination (LAX).



   (a) (15 pts) Please use each of the following search algorithms to plan the route for the airline. For each algorithm, write down (i) the order of the nodes expanded when executing the **tree** search version of the algorithm, and (ii) the order of the nodes that the airline actually will visit according to the trip you plan. As you plan the routes, if multiple options are available when expanding the next node, use **alphabetical order** for tie-breaking.

      i. (3 pts) Breadth-first search
      ii. (3 pts) Depth-first search
      iii. (3 pts) Uniform cost search
      iv. (3 pts) Greedy search using the values of $h$ for each node as the heuristic values
      v. (3 pts) A* search with the same heuristic

   (b) (4 pts) A logistics company urgently needs a small batch of high-value cargo transported from DEN to SFO. If the airline includes the DEN to SFO segment as part of its itinerary, the company provides a payout that more than offsets the operating cost of that leg. We can model this situation by assuming the company offers a reward of 6 for flying from DEN to SFO. Since the original cost of the DEN to SFO edge is 4, the net effect is a negative edge cost of -2. Modify the graph accordingly by re-assigning a cost of -2 to the edge DEN to SFO, and re-plan the route using this updated graph.

      i. Can we get an optimal route plan when solving this problem using the A* algorithms listed in Question 1(a)(v)? What is your reasoning?
      ii. What is the updated optimal trip plan?

## Submission

Please upload your answers to the **written** questions (i.e., Part 1) as a **pdf** in Gradescope:

- You should already have received an e-mail with the link to access Gradescope. If you haven't, let the TAs know.
- For your pdf file, use the naming convention `username_hw#.pdf`. For example, your TA with username *alice* would name her pdf file for HW1 as `alice_hw1.pdf`.
- Use the provided LaTeX template to typeset the assignment by editing the tex files under `student_response/`.
- After uploading your submission to Gradescope, mark each page to identify which question is answered on the page. (Gradescope will facilitate this.)
- Follow the above convention and instructions for future assignments as well.

## Part 2: Programming Assignment (60 pts + 28 pts bonus)

For the programming assignments, we use the Pacman project designed for the course CS188 at UC Berkeley. Please see the detailed instructions in `project1.pdf` This classic assignment has been widely adopted at various institutes, e.g., UW, UIUC, UCLA, etc.

Please remember that solutions to any assignment should be your own. Using other people's solutions, within or outside Purdue goes against the academic honesty policy of this course. **The TAs will use code similarity measures to detect plagiarism cases against projects submitted in previous years, submitted by classmates, and code online (e.g., from Github), when grading the assignment.**

This homework uses Python 3.10. Follow the Python tutorial in `project0.pdf` to set up the environment.

1. (60 pts) Complete Questions 1-4 described in `project1.pdf`. Submit your modified versions of `search.py` for grading. We will multiply your original score returned by `autograder.py` (12 pts in total) by 5.
2. (28pts bonus) Complete Questions 5-8 described in `project1.pdf`. Submit your modified versions of `searchAgents.py` for grading. We will multiply your original score returned by `autograder.py` (14 pts in total) by 2.

### Submission

Please upload the following files: `search.py` and `searchAgents.py` to Gradescope.