

CS221 Problem Workout

Week 3

1) Problem 1

Sabina has just moved to a new town, which is represented as a grid of locations (see below). She needs to visit various shops S_1, \dots, S_k . From a location on the grid, Sabina can move to the location that is immediately north, south, east, or west, but certain locations have been blocked off and she cannot enter them. It takes one unit of time to move between adjacent locations. Here is an example layout of the town:

| | | | | |
|-----------------------|--------------------|--------------------|--------------------|--------------------|
| | (2,5) | (3,5) | (4,5) | |
| (1,4) | S1 (2,4) | (3,4) | S2 (4,4) | (5,4) |
| (1,3) | (2,3) | | (4,3) | (5,3) |
| | (2,2) | (3,2) | (4,2) | S3 (5,2) |
| House (1,1) | (2,1) | S4 (3,1) | (4,1) | (5,1) |

Sabina lives at (1, 1), and no location contains more than one building (Sabina's house or a shop).

- (a) Sabina wants to start at her house, visit the shops S_1, \dots, S_k **in any order**, and then return to her house as quickly as possible. We will construct a search problem to find the fastest route for Sabina. Each state is modeled as a tuple $s = (x, y, A)$, where (x, y) is Sabina's current position, and A is some auxiliary information that you need to choose. If an action is invalid from a given state, set its cost to infinity. Let V be the set of valid (non-blocked) locations; use this to define your search problem. You may assume that the locations of the k shops are known. You must choose a minimal representation of A and solve this problem for general k . Be precise!

- Describe A : _____

- $s_{\text{start}} =$ _____

- $\text{Actions}((x, y, A)) = \{N, S, E, W\}$

- $\text{Succ}((x, y, A), a) =$

- $\text{Cost}((x, y, A), a) =$

- $\text{IsGoal}((x, y, A)) =$

(b) Sabina is considering a few different methods to visit the shops in as few steps as possible. For each of the following, state whether the algorithm will be able to find a path to visit all shops in as few steps as possible, and if so, provide a running time assuming an $N \times N$ grid.

- Depth-First Search (DFS)
- Backtracking search

(c) Recall that Sabina is allowed to visit the shops **in any order**. But she is impatient and doesn't want to wait around for your search algorithm to finish running. In

response, you will use the A* algorithm, but you need a heuristic. For each pair of shops (S_i, S_j) where $i \neq j$ and $1 \leq i, j \leq k$, define a **consistent** heuristic $h_{i,j}$ that approximates the time it takes to ensure that shops S_i and S_j are visited and then return home. Computing $h_{i,j}(s)$ should take $O(1)$ time.

2) Extra: Problem 2

In 16th century England, there were a set of $N + 1$ cities $C = \{0, 1, 2, \dots, N\}$. Connecting these cities were a set of bidirectional roads R : $(i, j) \in R$ means that there is a road between city i and city j . Assume there is at most one road between any pair of cities, and that all the cities are connected. If a road exists between i and j , then it takes $T(i, j)$ hours to go from i to j .

Romeo lives in city 0 and wants to travel along the roads to meet Juliet, who lives in city N . They want to meet.

- (a) Fast-forward 400 years and now our star-crossed lovers now have iPhones to coordinate their actions. To reduce the commute time, they will both travel at the same time, Romeo from city 0 and Juliet from city N .

To reduce confusion, they will reconnect after each traveling a road. For example, if Romeo travels from city 3 to city 5 in 10 hours at the same time that Juliet travels from city 9 to city 7 in 8 hours, then Juliet will wait 2 hours. Once they reconnect, they will both traverse the next road (neither is allowed to remain in the same city). Furthermore, they must meet in the end in a city, not in the middle of a road. Assume it is always possible for them to meet in a city.

Help them find the best plan for meeting in the least amount of time by formulating the task as a (single-agent) search problem. Fill out the rest of the specification:

- Each state is a pair $s = (r, j)$ where $r \in C$ and $j \in C$ are the cities Romeo and Juliet are currently in, respectively.
- $\text{Actions}((r, j)) =$ _____
- $\text{Cost}((r, j), a) =$ _____
- $\text{Succ}((r, j), a) =$ _____
- $s_{\text{start}} = (0, N)$
- $\text{IsGoal}((r, j)) = \mathbb{I}[r = j]$ (whether the two are in the same city).

- (b) Assume that Romeo and Juliet have done their CS221 homework and used Uniform Cost Search to compute $M(i, k)$, the minimum time it takes one person to travel from city i to city k for all pairs of cities $i, k \in C$.

Recall that an A* heuristic $h(s)$ is consistent if

$$h(s) \leq \text{Cost}(s, a) + h(\text{Succ}(s, a)). \quad (1)$$

Give a consistent A* heuristic for the search problem in (a). Your heuristic should take $O(N)$ time to compute, assuming that looking up $M(i, k)$ takes $O(1)$ time. In one sentence, explain why it is consistent. Hint: think of constructing a heuristic based on solving a relaxed search problem.

$$h((r, j)) = \underline{\hspace{15cm}} \quad (2)$$

2. Karel Loves Cookies! (30 points)

We revisit a homework question and check in on Karel, our favorite cookie-loving robot. Recall that Karel is a robot placed in an $m \times n$ grid and wishes to reach the cell with the cookie using the least cost. There are also pitfalls scattered on the grid – be careful not to step into them!

At each step, Karel may use one of the following commands:

1. `turnLeft()`: rotate the current direction 90 degrees counter-clockwise (costs 1);
2. `turnRight()`: rotate the current direction 90 degrees clockwise (costs 1);
3. `move()`: shift by 1 cell along the current direction (costs 1);
4. `leap()`: shift by 2 cells along the current direction (costs 3).

Note that moving or leaping onto a pitfall has an additional cost of 100 and terminates the game, but leaping over a pitfall doesn't have such penalty.

Initially, Karel starts at the top-left corner of the grid (1, 1) and faces east. You may assume that the cookie is always placed at the bottom-right corner (m, n) unless otherwise specified.










| | | | | | | | | |
|---|--------|---|--------|---|--------|--|--------|---|
|  | (1, 2) |  | (1, 4) | (1, 5) | (1, 6) |  | (1, 8) | (1, 9) |
|  | (2, 2) | (2, 3) | (2, 4) |  | (2, 6) | (2, 7) | (2, 8) |  |
| (3, 1) | (3, 2) |  | (3, 4) | (3, 5) | (3, 6) |  | (3, 8) |  |

Figure 1: An example 3×9 grid with 7 pitfalls and 1 cookie. Karel starts at (1, 1) and faces east. An example trajectory to the cookie: `move()` into (1, 2), \rightarrow `leap()` into (1, 4) since there's a pitfall in the middle \rightarrow `move()` twice into (1, 6) \rightarrow `leap()` into (1, 8) \rightarrow `move()` into (1, 9) \rightarrow `turnRight()` \rightarrow `leap()` into the cookie!

a. (6 points) Fill out the components of the search problem corresponding to the above problem setting. Each state consists of the cell coordinate (i, j) and the direction $d \in \{E, S, W, N\}$.

In your answer write up, you may use helper functions `counter-clockwise(d)`, `clockwise(d)`, $\Delta i(d)$, $\Delta j(d)$ whose values are defined in Table 1:

- $s_{\text{start}} = (1, 1, E)$.
- $\text{Actions}(i, j, d) = \{a \in \{\text{turnLeft}(), \text{turnRight}(), \text{move}(), \text{leap}()\} : \text{Succ}((i, j, d), a) \text{ doesn't exceed the boundary}\}$

| d | counter-clockwise(d) | clockwise(d) | $\Delta i(d)$ | $\Delta j(d)$ |
|-----|--------------------------|------------------|---------------|---------------|
| E | N | S | 0 | +1 |
| N | W | E | -1 | 0 |
| W | S | N | 0 | -1 |
| S | E | W | +1 | 0 |

Table 1: Helper functions for answer write up.

- $\text{IsEnd}(i, j, d) =$

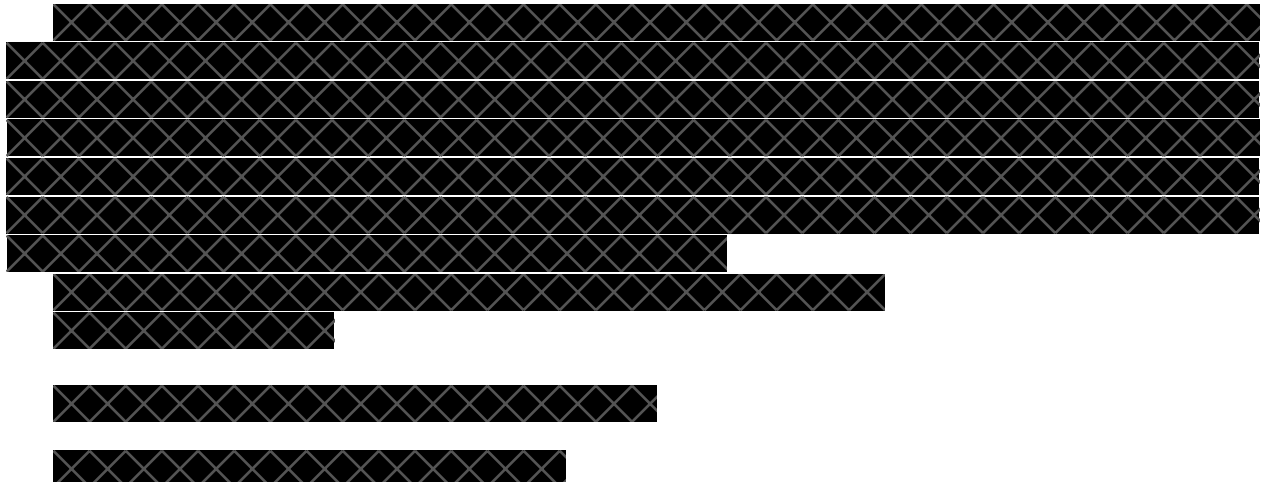
- $\text{Succ}((i, j, d), a) =$

- $\text{Cost}((i, j, d), a) =$

b. (*6 points*) You decide to use the A* algorithm to help Karel find the cookie with the least cost. Define a consistent heuristic function $h(i, j, d)$ for this problem. Briefly explain why your heuristic function is consistent.

[Hint: Think about a relaxed version of the original problem.]

c. (*2 points*) Would dynamic programming be an appropriate algorithm for solving this search problem as well? Explain why or why not.



2. Maze (50 points)

One day, you wake up to find yourself in the middle of a corn field holding an axe and a map (Figure 1). The corn field consists of an $n \times n$ grid of cells, where some adjacent cells are blocked by walls of corn stalks; specifically, for any two adjacent cells (i, j) and (i', j') , let $W((i, j), (i', j')) = 1$ if there is a wall between the two cells and 0 otherwise. For example, in Figure 1, $W((1, 1), (1, 2)) = 0$ and $W((1, 2), (1, 3)) = 1$.

You can either move to an adjacent cell if there's no intervening wall with cost 1, or you can use the axe to cut down a wall with cost c without changing your position. Your axe can be used to break down at most b_0 walls, and your goal is to get from your starting point (i_0, j_0) to the exit at (n, n) with the minimum cost.

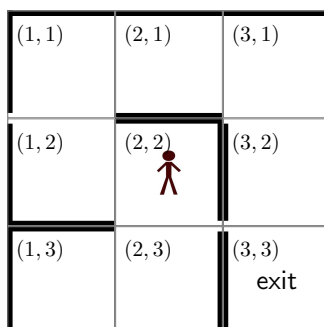


Figure 1: An example of a corn maze. The goal is to go from the initial location $(i_0, j_0) = (2, 2)$ to the exit $(n, n) = (3, 3)$ with the minimum cost.

a. (15 points)

(i) [10 points] Fill out the components of the search problem corresponding to the above maze.

- $s_{\text{start}} = ((i_0, j_0), b_0)$.
- $\text{Actions}(((i, j), b)) = \{a \in \{(-1, 0), (+1, 0), (0, -1), (0, +1)\} : (i, j) + a \text{ is in bounds and } (W((i, j), (i, j) + a) = 0 \text{ or } b > 0)\}$.
- $\text{IsEnd}(((i, j), b)) =$

- $\text{Succ}(((i, j), b), a) =$

- $\text{Cost}(((i, j), b), a) =$

(ii) [5 points] When you use an axe to take down a wall, the wall stays down but the set of walls which have been taken down are not tracked in the state. Why does our choice of state still guarantee the minimum cost solution to the problem?

b. (*10 points*)

Solving the search problem above is taking forever and you don't want to be stuck in the corn maze all day long. So you decide to use A*.

(i) [5 points] Define a consistent heuristic function $h(((i, j), b))$ based on finding the minimum cost path using the relaxed state (i, j) where we assume we have an infinite axe budget and therefore do not need to track it. Show why your choice of h is consistent and what you would precompute so that evaluating any $h(((i, j), b))$ takes $O(1)$ time and precomputation takes $O(n^2 \log n)$ time.

(ii) [5 points] Noticing that sometimes h is the true future cost of the original search problem, you wonder when this holds more generally. For what ranges of b_0 and c would this hold? Assume for this part that there is a path that doesn't require breaking down any walls.

$$\text{_____} \leq b_0 \qquad \text{_____} \leq c$$

Your lower bounds need not be tight, but you need to formally justify why they hold.