CS360 – Homework #5

Search

1) Solve Problem 3 from Homework 4 for A* with a) the (consistent) Manhattan distance heuristic and b) the straight line distance heuristic. The Manhattan distance heuristic between two grid cells (x_1, y_1) and (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$ (the length of a shortest path between the two cells, assuming that there are no obstacles on the grid). For instance, the Manhattan distance between A1 and E3 is $|1 - 5| + |1 - 3| = 6$. Remember to expand every state at most once. Explain which of the two heuristics one should prefer and why.

Manhattan distance heuristic:

	A	B	C	D	E
1	2	3	4	5	6
2	■	■	3	■	5
3	g	1	2	■	4
4	1	2	3	■	5
5	2	3	4	5	6

Expansion order (with f values): E1 (6), D1 (6), C1 (6), B1 (6), A1 (6), C2 (6), C3 (6), B3 (6), A3 (6)

Straight-line distance heuristic (approximately):

	A	B	C	D	E
1	2	2.2	2.8	3.6	4.2
2	■	■	2.2	■	4.1
3	g	1	2	■	4
4	1	1.4	2.2	■	4.1
5	2	2.2	2.8	3.6	4.2

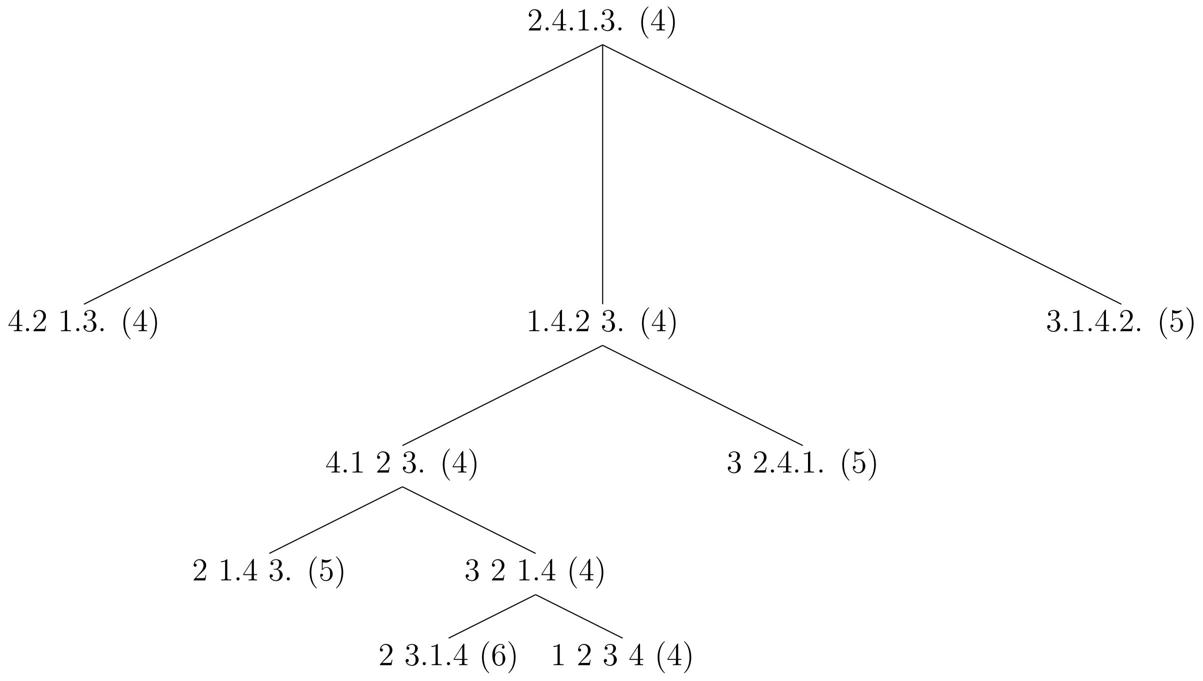
Expansion order (with f values): E1 (4.2), D1 (4.6), C1 (4.8), E2 (5.1), B1 (5.2), C2 (5.2), A1 (6), C3 (6), B3 (6), A3 (6).

In this case, Manhattan distance heuristic should be preferred because it dominates the straight-line distance heuristic (and, therefore, results in no more state expansions by A*, possibly with the exception of states where f-values equal the length of a shortest path from the start to the goal). Note that this argument holds because both heuristics are consistent.

2) We are given a sequence of integers and want to sort them in ascending order. The only operation available to us is to reverse the order of the elements in some prefix of the sequence. For instance, by reversing the first three elements of (1 2 3 4), we get (3 2 1 4). This problem is also known as the “pancake flipping" problem. We model this problem as a search problem, where each state corresponds to a different ordering of the elements in the sequence. Given an initial sequence (2 4 1 3), in which order does A* expand the states, using the breakpoint heuristic described below? Assume that ties are broken toward states with larger g-values, and, if there are still ties, they are broken in lexicographic order. That is, (2 1 4 3) is preferred to (2 4 1 3).

Breakpoint heuristic: A breakpoint exists between two consecutive integers if their difference is more than one. Additionally, a breakpoint exists after the last integer in the sequence if it is not the largest integer in the sequence. For instance, in (2 1 4 3), there are two breakpoints: one between 1 and 4 (since their difference is more than 1), the other after 3 (since it is at the end and of the sequence and is not the largest integer in the sequence). The breakpoint heuristic is the number of breakpoints in a given sequence. (Bonus question: Is this heuristic a) admissible and b) consistent? Why?)

We start with the sequence (2.4.1.3.) which has 4 breakpoints (dots represent the breakpoints). We have only three actions available to us: reverse the order of the first two elements, first three elements, or all the elements (reversing the order of only the first element is pointless). The search tree is given below:



The order of expansions (and their f-values) are as follows:

(a) (2.4.1.3.) 4

(b) (1.4.2 3.) 4

(c) (4.1 2 3.) 4

(d) (3 2 1.4) 4

(e) (1 2 3 4) 4

The breakpoint heuristic is consistent (and therefore admissible). The goal state (1 2 3 4) has no breakpoints, therefore $h(goal) = 0$. Each action can change the number of breakpoints by at most 1 (reversing the first i elements can only effect the breakpoint after i). Therefore, for any edge (s, s'), $0 \leq h(s) \leq 1 + h(s')$ holds.

3) Does A* always terminate if a finite-cost path exists? Why?

No. If a state has an infinite number of successors (infinite branching factor), A* cannot expand that state in a finite amount of time and, therefore, does not terminate. If we assume that each state has a finite number of successors (finite branching factor), A* may still not terminate because it can get stuck exploring an infinite subspace of the state space. Consider the following state space with states $\{s_{goal}, s_0, s_1, \ldots\}$, and edges $\{(s_0, s_{goal}), (s_0, s_1), (s_1, s_2), \ldots\}$, where $c(s_0, s_{goal}) = 2$ and, for all i, $c(s_i, s_{i+1}) = (1/2)^i$. An A* search from s_0 to s_{goal} that uses a 0-heuristic ($h(s) = 0$ for all states s) would have to expand $s_0, s_1, \ldots$ before expanding s_{goal} (since, for all i, $f(s_i) = 1+1/2+1/4+\cdots+(1/2)^{i-1} < 2 = f(s_{goal})$). Since an infinite number of expansions are required before expanding s_{goal}, A* does not terminate.

4) Given two consistent heuristics h_1 and h_2, we compute a new heuristic h_3 by taking the maximum of h_1 and h_2. That is, $h_3(s) = max(h_1(s), h_2(s))$. Is h_3 consistent? Why?

Yes. First, for any goal state s, we show that $h_3(s) = 0$. Since, h_1 and h_2 are consistent, $h_1(s) = 0$ and $h_2(s) = 0$. Therefore, $h_3(s) = max(h_1(s), h_2(s)) = 0$.

Then, for any edge (s, s') in the graph, we show that $0 \leq h_3(s) \leq c(s, s') + h_3(s')$. Without loss of generality, assume that $h_1(s) \geq h_2(s)$ (that is, if $h_1(s) < h_2(s)$, we could simply switch h_1 and h_2, and continue with the proof). Then, $h_3(s) = max(h_1(s), h_2(s)) = h_1(s)$. Since h_1 is consistent, we have:

$$0 \leq h_1(s) \leq c(s, s') + h_1(s')$$

We get:

$$\begin{aligned} 0 \leq h_3(s) = max(h_1(s), h_2(s)) = h_1(s) &\leq c(s, s') + h_1(s') \\ &\leq c(s, s') + max(h_1(s'), h_2(s')) = c(s, s') + h_3(s') \end{aligned}$$

□

5) In the arrow puzzle, we have a series of arrows pointing up or down, and we are trying to make all the arrows point up with a minimum number of action executions. The only action available to us is to chose a pair of adjacent arrows and flip both of their directions. Using problem relaxation, come up with a good heuristic for this problem.

We can relax the problem by allowing our action to flip the directions of any two arrows, instead of two adjacent arrows. In this case, we need to use at least $\lceil$ number of arrows pointing down / 2 $\rceil$ actions, which can be used as a consistent heuristic to solve the original problem.

6) Explain why heuristics obtained via problem relaxation are not only admissible but also consistent.

Relaxing a problem means creating a supergraph of the state space by adding extra edges to the original graph of the state space and using goal distances on this supergraph as heuristics for an A* search in the original graph. We start by showing that the goal distances in the supergraph form a consistent heuristic for the supergraph. The goal distance of the goal is 0, so $h(goal) = 0$. For any edge (s, s') of the supergraph, $0 \leq h(s) \leq c(s, s') + h(s')$ because, otherwise, $h(s)$ would not be the goal distance of s, since there were a shorter path to the goal via s'. Since the edges in the original graph is a subset of the edges in the supergraph, for all edges (s, s') of the original graph, $0 \leq h(s) \leq c(s, s') + h(s')$. □

7) What are the advantages and disadvantages of a) uniform-cost search and b) greedy best-first search over A* search?

When a consistent heuristic is used, A* expands no more states than uniform-cost search would expand (possibly with the exception of states where f-values equal the length of a shortest path from the start to the goal). In this case, the only advantage uniform-cost search has is that it does not have to compute a heuristic (which can be very expensive in some domains). Both search methods guarantee optimality.

Greedy best-first search offers no optimality guarantees, although it typically finds solutions with many fewer node expansions and thus much faster than A*.