Heuristic Search– Solution

1) Solve Problem **??** for A* with a) the 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 $|1 - 5| + |1 - 3| = 6$.

Answer:

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	4.5
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	4.5

Expansion order (with f values): E1 (4.5), D1 (4.6), C1 (4.8), E2 (5.1), B1 (5.2), C2 (5.2), A1 (6), C3 (6), B3 (6), A3 (6).
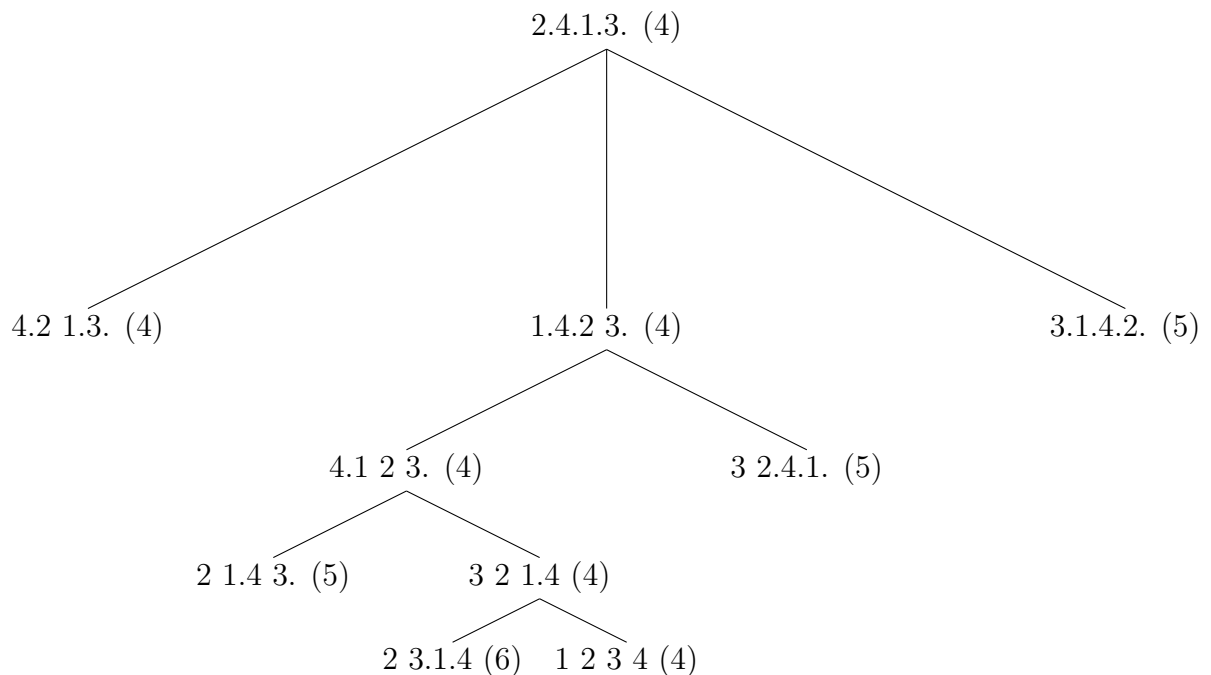
2) We are given a sequence of integers and want to sort them in ascending order. The only operation available to us is to reverse the order of the elements in some prefix of the sequence. For instance, by reversing the first three elements of (1 2 3 4), we get (3 2 1 4). This problem is also known as 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?)

Answer:
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:

2.4.1.3. (4)

4.2 1.3. (4)   1.4.2 3. (4)   3.1.4.2. (5)

4.1 2 3. (4)   3 2.4.1. (5)

2 1.4 3. (5)   3 2 1.4 (4)

2 3.1.4 (6)   1 2 3 4 (4)

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?

Answer:
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?

Answer:
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 $0 \le h_3(s) \le c(s, s') + h_3(s')$. Without loss of generality, assume that $h_1(s) \ge h_2(s)$ (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:

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

We get:

$$0 \le h_3(s) = max(h_1(s), h_2(s)) = h_1(s) \le c(s, s') + h_1(s')$$

$$\le c(s, s') + max(h_1(s'), h_2(s')) = c(s, s') + h_3(s')$$

□

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.

Answer:
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 ⌈ number of arrows pointing down / 2 ⌉ 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.

Answer:
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, $0 \leq h(s) \leq c(s, s') + h(s')$. $\qquad\square$

7) What are the advantages and disadvantages of a) uniform-cost search and b) pure heuristic search over A* search?

Answer:
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.

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

8) Give a simple example that shows that A* with inconsistent but admissible heuristic values is not guaranteed to find a cost-minimal path if it expands every state at most once.

Answer:
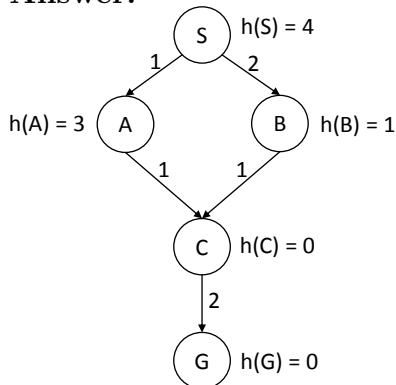
In this example (where the heuristic is admissible but not consistent), when searching for a path from S to G, the states are expanded in the following order (parentheses show f-values):

S $(0 + 4 = 4)$
B $(2 + 1 = 3)$
C $(3 + 0 = 3)$

$A \ (1 + 3 = 4)$
$G \ (5 + 0 = 5)$

Note that expanding A finds a shorter path from S to C, but this is not reflected in the solution since C was expanded before A and we are not allowed to expand a state more than once.

9) Solve Problem **??** for Iterative Deepening search.

Answer:
In the solution below, we use '*' to mark nodes that are at the depth-limit for the current iteration of Iterative Deepening search.

Expansion order in 1st iteration (depth 0): E1*.

Expansion order in 2nd iteration (depth 1): E1, D1*, E2*.

Expansion order in 3rd iteration (depth 2): E1, D1, C1*, E2, E3*.

Expansion order in 4th iteration (depth 3): E1, D1, C1, B1*, C2*, E2, E3, E4*.

Expansion order in 5th iteration (depth 4): E1, D1, C1, B1, A1*, C2, C3*, E2, E3, E4, E5*.

Expansion order in 6th iteration (depth 5): E1, D1, C1, B1, A1, C2, C3, B3*, C4*, E2, E3, E4, E5, D5*.

Expansion order in 7th iteration (depth 6): E1, D1, C1, B1, A1, C2, C3, B3, A3*.

Note that the nodes marked with * also mark the first time that they are expanded. If we list the nodes in the order that they are first expanded, we get:

E1, D1, E2, C1, E3, B1, C2, E4, A1, C3, E5, B3, C4, D5, A3.

This is equivalent to the BFS expansion order.

10) Develop an iterative-deepening variant of A* (call Iterative Deepening A*), that is, a version of A* that finds cost-minimal paths for consistent heuristics and uses a series of depth-first searches to keep its memory consumption small. (Hint: If you are stuck, look at the lecture slides on heuristic search.) Solve Problem **??** for this version of A* with the (consistent) Manhattan distance heuristic.

Answer:
In Iterative Deepening search, each iteration explores the search tree up to a certain depth (given by a depth limit). In Iterative Deepening A* (IDA*) search, each iteration explores the search tree up to a certain f-value (given by an f-value limit). That is, IDA* expands nodes in a depth-first manner, but it does not expand nodes whose f-values are larger than the f-value limit for the current iteration.

Instead of starting with an f-value limit of 0 and increasing it by 1 with every iteration (which might not guarantee optimality if the action costs are not all 1),

IDA* first sets the f-value limit to $f(start) = h(start)$ and, for each subsequent iteration, updates the limit to the minimum f-value among the nodes that were generated but not expanded in the previous iteration (that is, the nodes whose f-values are above the f-value limit in the previous iteration but the f-values of whose parents are no larger than the f-value limit). This guarantees that the search finds cost-minimal paths if the heuristics are consistent.

Below is the solution of Problem 1 from Homework 1 using IDA*.

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

Since the f-value of node E1 is 6, we start the first iteration of IDA* with f-value limit 6.

Expansion order in 1st iteration (with f-values): E1 (6), D1 (6), C1 (6), B1 (6), A1 (6, backtrack to C1), C2 (6), C3 (6), B3 (6), A3 (6).

Iterative Deepening A* finds a cost-minimal path in its first iteration.

However, if we were searching from E2 (instead of E1) to A3, it would take two iterations for Iterative Deepening A* to find a cost-minimal path:

Expansion order in 1st iteration (with f-values): E2 (5), (E1 not expanded since $7 > 5$), E3 (5), (E4 not expanded since $7 > 5$).

There are no more nodes to expand, so the 1st iteration terminates without finding a path. Since the smallest f-value among nodes generated but not expanded is 7, we start the 2nd iteration of IDA* with f-value limit 7.

Expansion order in 2nd iteration (with f-values): E2 (5), E1 (7), D1 (7), C1 (7), B1 (7), A1 (7, backtrack to C1), C2 (7), C3 (7), B3 (7), A3 (7).

11) (Thanks to Ariel Felner.) Given two admissible 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, $h_3(s) = max(h_1(s), h_2(s))$); and another heuristic h_4 by taking the sum of h_1 and h_2 (that is, $h_4(s) = h_1(s) + h_2(s)$). Is h_3 admissible? Why? Is h_4 admissible? Why?

Answer:
Let $h^*(s)$ denote the true distance from a state s to a closest goal state, called goal distance.

h_3 is admissible because, for any state s, if $h_1(s) \leq h^*(s)$ and $h_2(s) \leq h^*(s)$, then it must hold that $h_3(s) = max(h_1(s), h_2(s)) \leq h^*(s)$.

h_4 is not admissible as the following counter example shows: For some state s, let $h^*(s) = 4$, $h_1(s) = 2$, and $h_2(s) = 3$. Then, $h_4(s) = h_1(s) + h_2(s) = 2 + 3 = 5 > 4 = h^*(s)$.

12) (Thanks to Ariel Felner.) Develop an admissible heuristic for the following variants of the 15-puzzle:

a) Moving tile x costs x instead of 1.

Answer:
We can use a weighted Manhattan distance heuristic where the goal distance of each tile x is multiplied by x.

b) Moving a row of tiles left or right as well as moving a column of tiles up and down costs 1 regardless of the number of tiles that are moved. For example, if a row's initial configuration is (B,1,2,3), where B is the blank (that is, missing tile), then it costs 1 to change this configuration to any of the following configurations: (1,B,2,3) or to (1,2,B,3) or to (1,2,3,B).

Answer:
In the most extreme case, three different tiles can be moved with a cost of 1. We can therefore use the regular Manhattan distance heuristic divided by 3.