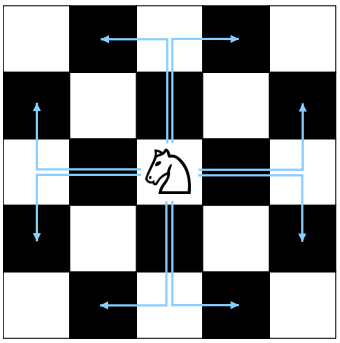
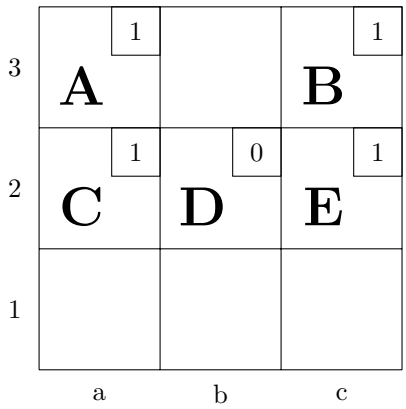
02180 Intro to AI
Exercises for week 3, 16/2-21
SOLUTIONS

Exercise 1

This exercise is from the exam of the course in 2018. A knight in the game of chess can move and attack as shown by the arrows in the figure below.



In the following, we want to place five knights on a 3x3 board such that none of them can attack each other. Let A, B, C, D and E denote the five knights. Initially, the knights are placed as shown below.



In this state, A can attack E by moving to position 2c, A can also move to position 1b without attacking another knight.

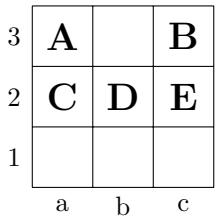
Let the number of conflicts of a knight be equal to the number of knights that can attack it.

1. For the initial state of the board shown above, write the number of conflicts of each knight in the small box next to its letter.

   SOLUTION. See above.

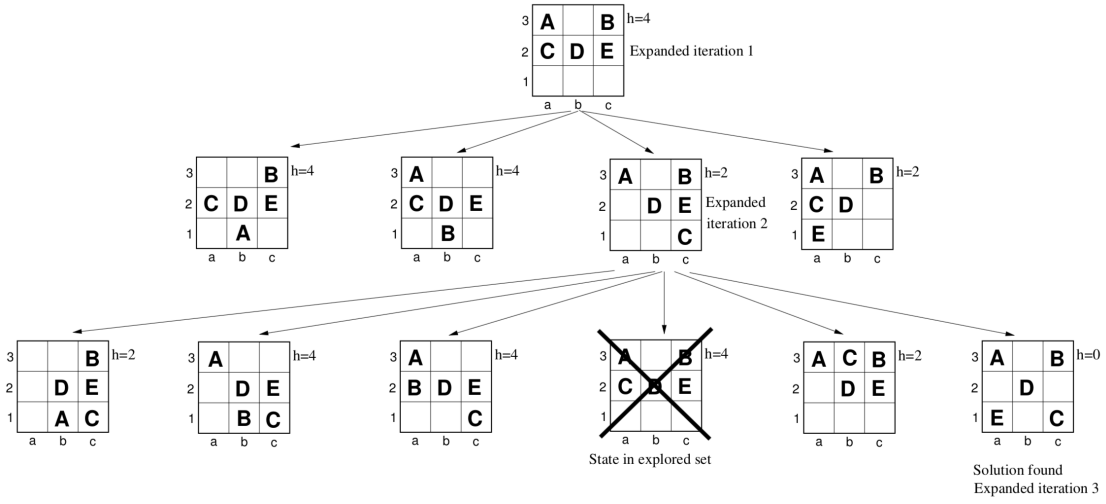
2. Consider the search problem in which the initial state is the initial configuration of the board shown above, the applicable actions are the legal moves of the knights, and a goal state is one in which none of the knights have a conflict. The cost of each action is assumed to be 1. Consider the heuristic function $h(n)$, where the value of $h(n)$ is defined to be the total sum of conflicts of the knights in the state of n (e.g., in a state of n where A, B, C and D each has 1 conflict and E has no conflicts, we have $h(n) = 4$).

   Draw below the search tree of the GRAPH-SEARCH version of greedy best-first search using the heuristic function $h(n)$. Write each search node as a board state and take note of its h-value. Indicate the expanded nodes by ✓. The root of the tree is already shown. You can assume that no new nodes are expanded as soon as a goal state has been generated. Assume that when expanding a node, moves of knight A are considered first, then moves of knight B, etc.



$h =$

   SOLUTION.



3. Enter the solution found by the algorithm into the table below. You might not need all rows.

Move number	Letter of piece	moved from	moved to
1			
2			
3			
4			
5			
6			

SOLUTION.

Move number	Letter of piece	moved from	moved to
1	C	2a	1c
2	E	2c	1a

4. Assume $A^\star$ is used instead of greedy best-first search. Moves are considered in the same order as before, and it is still assumed that no new nodes are expanded as soon as a goal state has been generated. Will the search tree then be the same? Explain your answer.

   SOLUTION. Yes. In level 1 we will have the f-values 1 higher than the h-values, but of course at level 1 we will still expand one with minimum h-value. Already at level 2 we find a solution.

5. Assume BFS is used instead of greedy-best search. Moves are considered in the same order as before. Will the search tree then be the same as for greedy best-first search? Explain your answer.

   SOLUTION. No. The leftmost child of the root will be expanded before the child with minimal h-value.

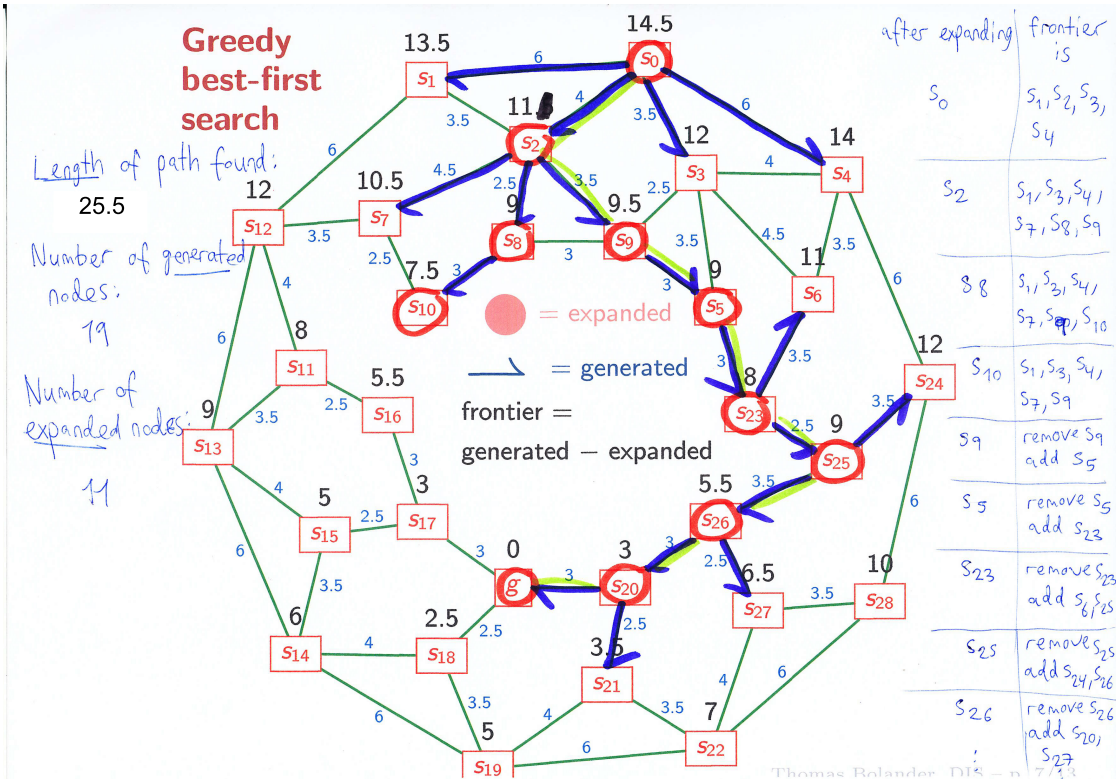
6. Is $h(n)$ an admissible heuristics for the search problem? Explain your answer.

   SOLUTION. No. The expanded child of the root above has $h = 2$ but is only one move away from the goal.

Exercise 2

1. Consider the state space on slide 8 from today (`slides03.pdf`). The h-values are in black, the step costs are in blue. The h-values are straight-line distances to g, and the step costs are straight-line distances between pairs of points. Run greedy best-first graph search by hand on this state space, finding a path from the initial state s_0 to the goal state g. Make sure to take note of the order in which the nodes are expanded as well as the content of the frontier after each iteration of the main loop of Graph-Search. Highlight each edge traversed as well as all expanded nodes.

SOLUTION. In the solution below, the frontier is only shown for the first iterations of the loop.



2. What is the length of the path found by greedy best-first search? How many nodes are generated by the search? How many nodes are expanded?

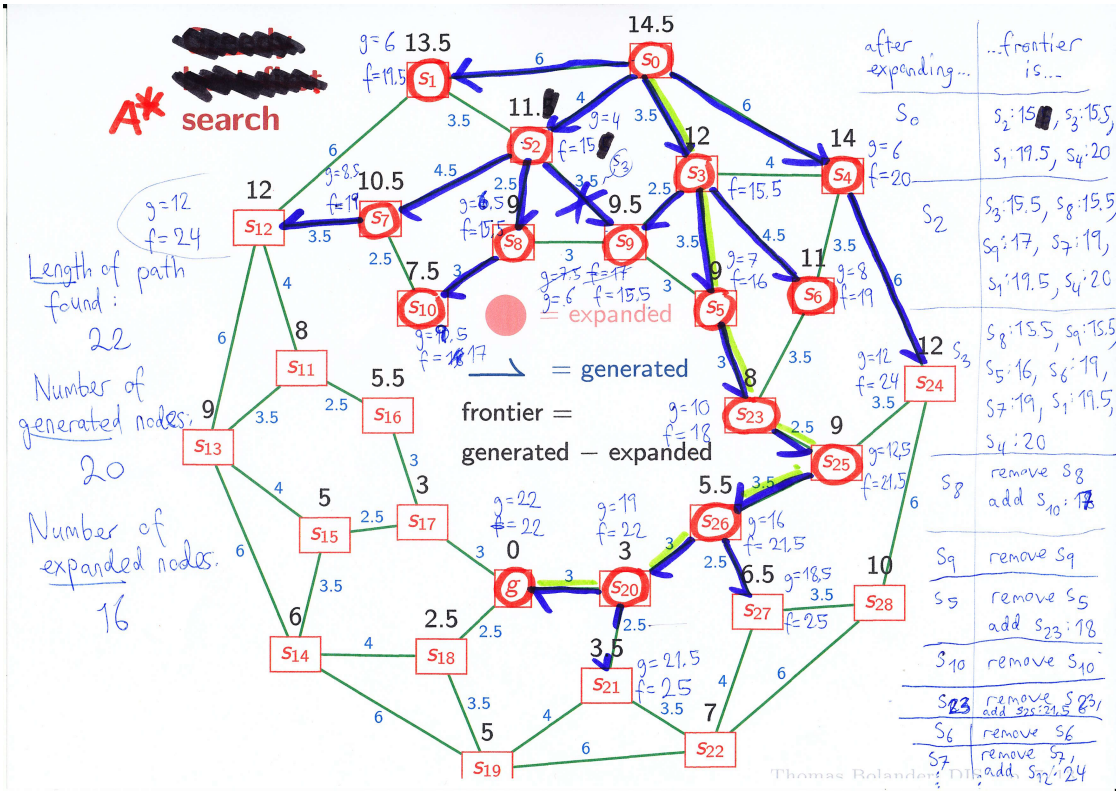
   SOLUTION. These numbers are included in the picture above.

3. In the slides and in the textbook, $A^\star$ is claimed to be an instance of the general graph-search algorithm. There is however one complication. $A^\star$ is a best-first search algorithm relying both on a heuristic value (h-value) and a path cost (g-value). The h-value of a node n is a constant, but the g-value will depend on the path taken to get to n from the initial state. This means that the g-value potentially needs to be updated during the search if a shorter path to n is found. Otherwise we can not be guaranteed to find the shortest path, even if the heuristics is consistent. The solution to this problem is to add the following line at the end of the GRAPH-SEARCH algorithm on page 2 of `slides03.pdf` (inside the **for**-loop):

   if $m \in frontier$ and $g(m) > g(n) + c(n, m)$ **then** let n be the new parent of m

   Here $c(n, m)$ is the cost of the edge from n to m. Note that when n is made the new parent of m, the g- and f-values of m will also implicitly be updated. Now repeat the search from above, but using $A^\star$ instead of greedy best-first search. Make sure to do the same book-keeping as for greedy best-first search and, in addition, mark each generated state with its g- and f-values.

SOLUTION. In the solution below, the frontier is only shown for the first iterations of the loop.



4. What is the length of the path found by $A^\star$? How many nodes are generated by the search? How many nodes are expanded?

   SOLUTION. These numbers are included in the picture above.

5. How do you know that the path produced by $A^\star$ is optimal (has minimal cost)?

   SOLUTION. The heuristics and step costs are based on straight-line distances, so if m is reached by executing a in n, then

   $$\begin{array}{rcl} h(n) & = & \text{straight-line distance from } n \text{ to } g \\ & \leq & \text{straight-line distance from } n \text{ to } g \text{ via } m \\ & = & \text{straight-line distance from } n \text{ to } m \text{ plus straight-line distance from } m \text{ to } g \\ & = & c(n,a,m) + h(m). \end{array}$$

   This proves that h is consistent. $A^\star$ graph-search is optimal when h is consistent, hence we can be guaranteed that the path found is optimal.

6. Consider adding an extra edge from s_9 to g. What is the cost of this edge? What is now the cost (length) of a shortest path from s_0 to g? Show that if the extra line mentioned in question 3 is not added to the pseudocode of Graph-Search, $A^\star$ will not find the optimal solution to the search problem with the extra edge added.

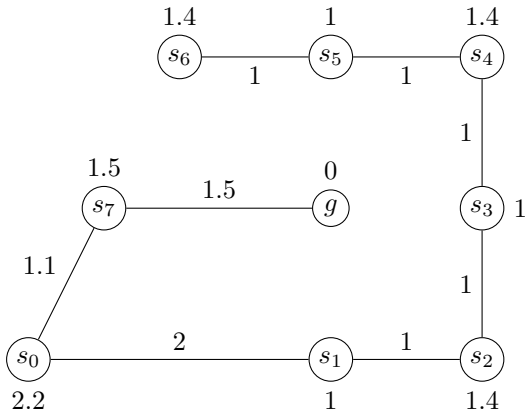
SOLUTION. Cost of new edge = heuristic value of $s_9 = 9.5$. Shortest path is (s_0, s_3, s_9, g) and it has a length of $3.5 + 2.5 + 9.5 = 15.5$. However, $A^\star$ without the added line will find the non-optimal path going via s_2 and having a cost of $4 + 3.5 + 9.5 = 17$.

7. Consider the following modified search problem. We wish to find a shortest path from s_0 to g measured in *number of edges traversed*, that is, we take the step cost to be constant 1. Can we still use $A^\star$ with the existing heuristics to find such a shortest path? If not, then how can we (easily) modify the existing heuristics to ensure that $A^\star$ will find a shortest path?

   SOLUTION. Divide all heuristic values by the length of the longest edge = 6. Then the consistency condition will be satisfied when the edge cost is 1. Since before we had $h(n) \leq c(n, n') + h(n')$. Call the new h values h'. Then $6h'(n) \leq c(n, n') + 6h'(n')$ and hence $h'(n) \leq \frac{c(n,n')}{6} + h'(n') \leq 1 + h'(n')$, which is the consistency condition for edge cost 1.

8. In the example considered here, greedy best-first search generates and expands fewer nodes than $A^\star$. Provide an example of an alternative graph where greedy best-first search generates and expands *more* nodes than $A^\star$. The nodes of the graph should still represent points in the plane, the step cost of edges should be the straight-line distance, and the heuristics should be straight-line distance to the goal.

   SOLUTION.



   $A^\star$ will first generate children s_7 and s_1 with $f(s_7) = 1.1 + 1.5 = 2.6$ and $f(s_1) = 3$. Then $f(s_7)$ will be expanded and we get $f(g) = 2.6$. Next g will be expanded, and we are done. So with $A^\star$, we expand s_0, s_7, g, in total 3 nodes. We additionally generate only s_1, so 4 nodes are generated. With greedy best-first search, the expansion order will be $s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, g$. So all 9 nodes will be expanded (and hence also generated).