University of Cagliari

MSc programs in Computer Engineering, Cybersecurity and Artificial Intelligence, and in Electronic Engineering

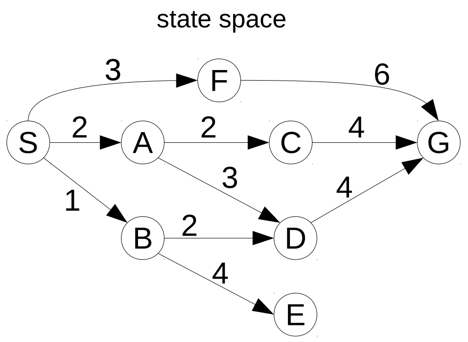
Artificial Intelligence

Academic Year: 2023/2024

Instructor: Giorgio Fumera

Exercises on search algorithms

1. The graph in the figure below shows the state space of a hypothetical search problem. States are denoted by letters, and the cost of each action is indicated on the corresponding edge. Note that the graph is *oriented*, which means that the actions are *not* reversible (e.g., it is possible to move from state `S` to `A`, but not vice versa). The table next to the state space shows the value of an admissible heuristic function, considering `G` as the goal state (it is easy to verify that such an heuristic is admissible, i.e., it never overestimates the minimum path cost from any given state to the goal state `G`).



heuristic function (goal state: G)

S	A	B	C	D	E	F	G
6	4	5	2	2	8	4	0

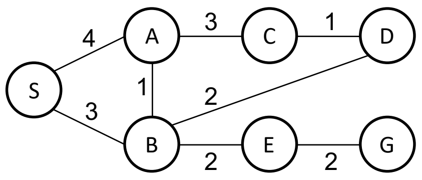
Considering `S` as the initial state, find a solution using each of the following search strategies, drawing the corresponding search tree:

- breadth-first search
- depth-first search
- A* search with the above heuristic

When drawing the search tree you should *clearly* indicate: the order of expansion of each node (e.g., by numbering the expanded nodes according to the order of their expansion); the action corresponding to each edge of the tree; the state, the path cost and the value of the heuristic of each node. In the case of depth-first search you separately draw the search tree after any node is deleted.

2. Consider the state space of a graph search problem shown below. The number on each edge denotes the cost of the corresponding action (note that actions are reversible), and `S` and `G` denote respectively the initial and the goal state. The table on the right shows the values of an admissible heuristic function.

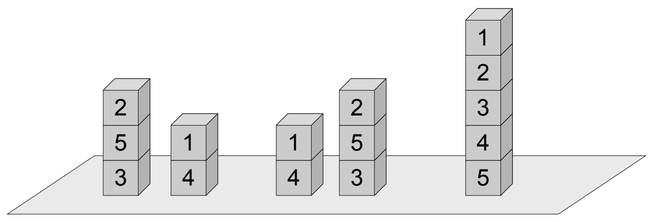
   Show how the depth-first and A* search strategies solve this problem, avoiding repeated states in the same path; in case of a tie, the choice of the next node to expand has to be made according to *alphabetical ordering*. For the sake of clarity, draw *separately* the search tree obtained after each node expansion.



S: 5	A: 3	C: 1
D: 2	B: 2	E: 1

3. The *block's world* is a well-known toy domain used in AI for planning problems related to robots. It consists of a set of blocks of various size, shape, and color, possibly identified by a letter or by a number, and placed on a surface (e.g., on a table); the blocks can be stacked into one or more piles, possibly with some constraints (depending, e.g., on their size); the goal is to form a target set of piles starting from a given initial configuration, by moving one block at a time, either to the table or on top of another pile of blocks, and only if it is not under another block.

   Consider a simple version of this problem, in which there are five blocks with identical shape and size, numbered from 1 to 5. The figure below shows three possible configurations of such blocks; note that the relative position of the piles does not matter, thus the configuration on the left is equivalent to the one in the middle. Every block can be either on the table (there are no constraints on the number of piles) or on the top of another block. Starting from any given set of piles, the goal is to stack the blocks in a single pile, in the order shown in the rightmost configuration in the figure, by moving the smallest possible number of blocks. A block can be moved only if it is on the top of a pile or on the table, and can be placed either on the table or on top of another pile.

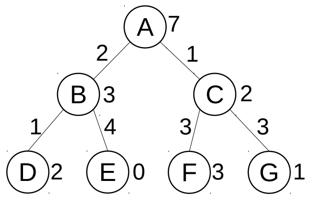


   (a) Formulate the above version of the block's world as a search problem, by *precisely* defining the state space, the initial state, the goal test, the set of possible actions and the path cost.

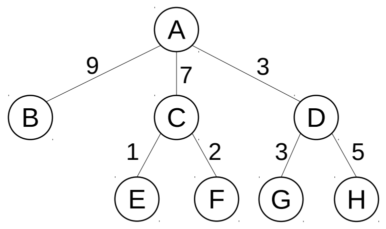
   (b) Assume that the initial configuration is the one shown on the left in the figure above, and consider a heuristic function defined as the number of blocks that are not in the highest pile (or in one of the highest piles). Under this setting, draw the search tree obtained after the first *four* expansion steps by the A* search strategy, avoiding repeated states. As in the previous exercise, when drawing the search tree you should *clearly* indicate: the order of expansion of each node; the action corresponding to each edge of the tree; the state, the path cost and the value of the heuristic of each node.

   (c) Define another possible admissible heuristic, and prove its admissibility.

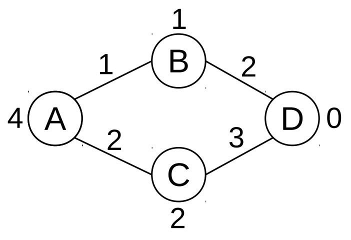
4. Assume that the search tree shown below has been obtained for some search problem after the first three expansion steps by the A* strategy. Numbers on the edges denote the cost of the corresponding actions, letters inside each node denote the corresponding state, and numbers next to each node denote the value of the heuristic function (which is assumed to be admissibile) for the same state. Which node will A* select for expansion in the next step?



5. Assume that the search tree shown below has been obtained for some search problem after the first three expansion steps by some *unknown* uninformed strategy. The meaning of letters and numbers is the same as in the previous exercise. Determine whether the underlying search strategy can be depth-first search, uniform cost search, or both.



6. Consider the *state space* of a given search problem in the figure below, where `D` is the goal state, numbers on each edge denote the cost of the corresponding action, and numbers next to each state denote the corresponding value of a heuristic function: is this heuristic *admissible*?



Solution

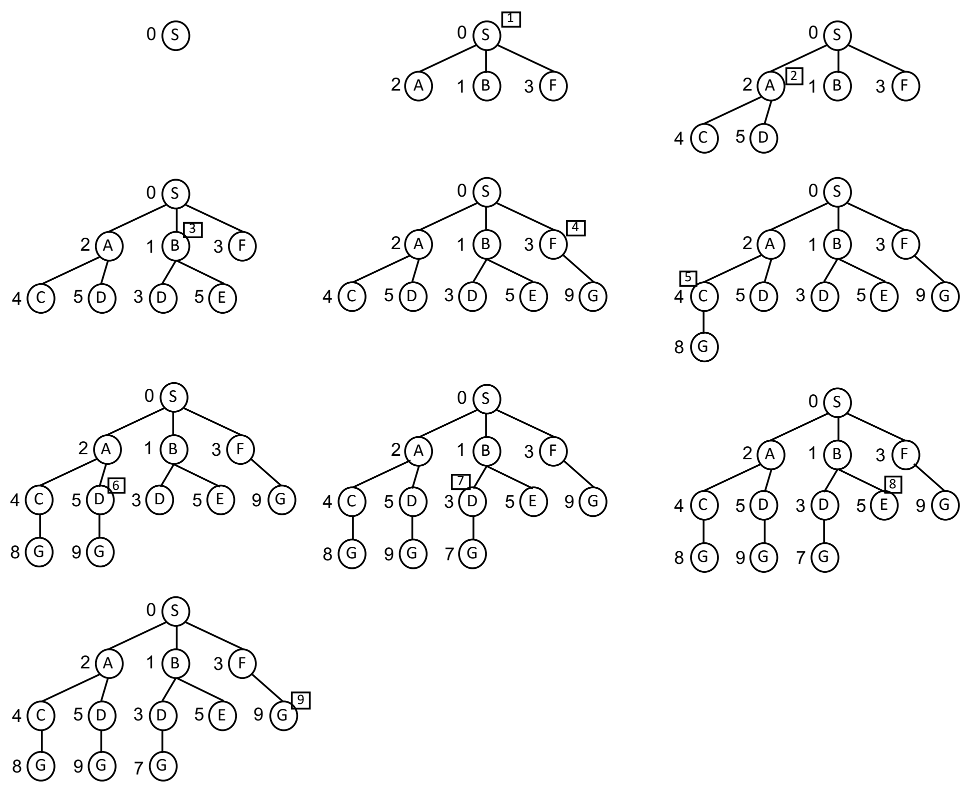
Exercise 1

The search trees built by the three strategies are shown below. For breadth- and depth-first search, the number of the left of each node denotes its path cost (although it is not used by these strategies to choose the next leaf node to expand); for A*–search, the same number denotes the sum of the path cost and of the heuristic function. The order of expansion is denoted by numbers inside squares. For the sake of clarity, the search tree obtained after each expansion step is shown.

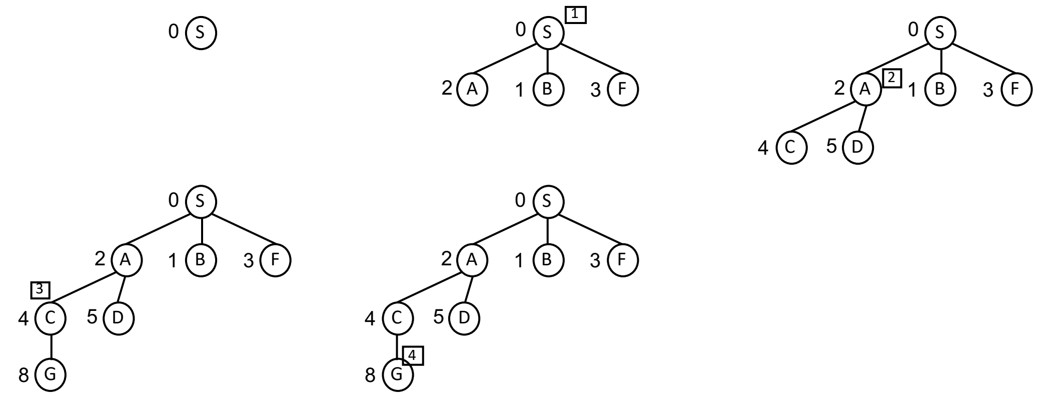
Remember that, when several leaf nodes can be chosen for expansion (i.e., when there is more than one leaf node at the lowest depth for breadth-first search, or at the highest depth for depth-first search, or with the lowest value of the sum of the path cost and the heuristic function for A*–search), a random choice should be made among them; in this exercise we assume that the choice is made by considering the alphabetical ordering among the corresponding states (e.g., if the choice is among nodes `A`, `B` and `F`, node `A` is chosen).

Finally, remember that the goal test is applied when a node is *selected for expansion*: therefore a solution is found not when a node containing a goal state is *generated* (by expanding its parent node), but when it is selected for expansion.

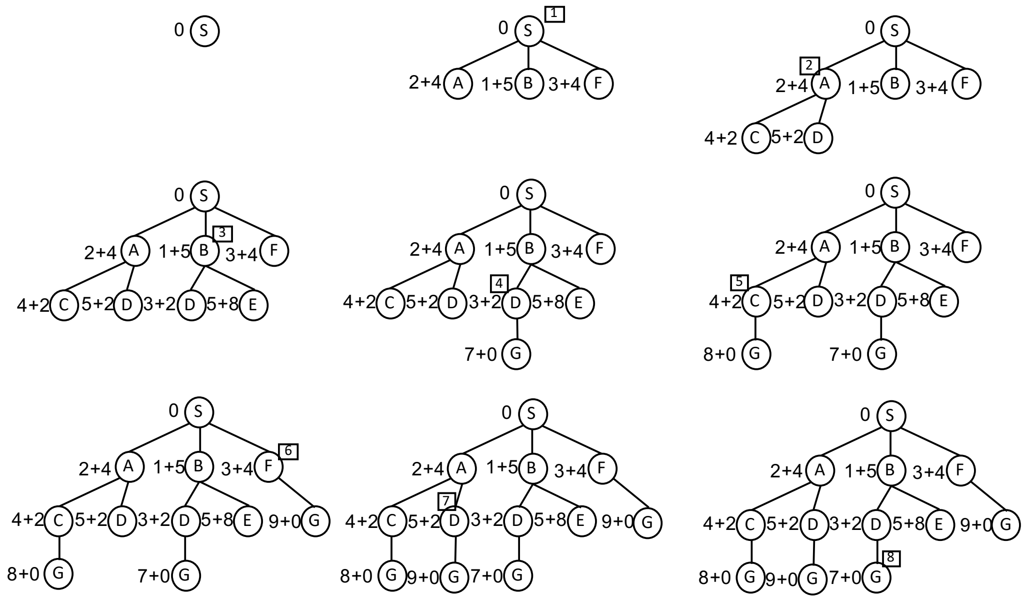
The search tree built by **breadth-first search**, step by step, is shown in the figure below. Note that different nodes can correspond to the same state (e.g., this happens to states `D` and `G`): this means that such a state can be reached from the initial state through *different* paths in the state space (i.e., different sequences of actions), possibly with different path costs: therefore such nodes are kept distinct in the search tree. Note also that node `E` has no successors (no other state can be reached from state `E`, according to the state space): therefore when it is selected for expansion (in step $\boxed{8}$) no children nodes are added to the tree. Finally, note that the solution found by breadth-first search (`S`$\to$`F`$\to$`G`) is not the optimal one (i.e., the one with minimum path cost): this is due to the fact that breadth-first is not optimal.



The search tree built by **depth-first search** is shown below. In this case a solution is found along the path which is explored first (when the node corresponding to state `G` is selected for expansion at step $\boxed{4}$). Therefore no backtracking step is carried out, and none of the already expanded nodes is deleted. The solution found by depth-first search is not the minimum-cost one, as for breadt-first, since depth-first search is not optimal, either.



The search tree built by **A*–search** is shown below. Next to each node the corresponding path cost g and heuristic h are shown as $g + h$. Remember that, at each step, the leaf node with the minimum value of $g+h$ is chosen for expansion (ties are broken randomly, as for all search strategies). The solution found at step $\boxed{8}$ is guaranteed to be the minimum-cost one, since A* is optimal when an admissible heuristic is used.

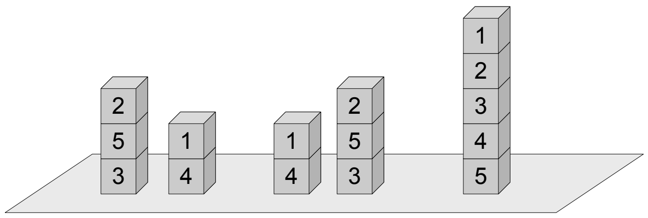


Exercise 2

The solution is left as an exercise.

Exercise 3

(a) The state space is made up of the set of distinct arrangements of the five blocks into one or more (up to five) piles. A state can be represented by a *set*[1] of piles, and each pile can be represented as a *sequence* of block numbers, where the numbers from left to right correspond, e.g., to block from the top to the bottom of the pile. For instance, the two equivalent configurations of piles (states) shown on the left and in the middle of the figure below are represented by the set $\{(2,5,3),(1,4)\}$, and the goal state on the right is represented by $\{(1,2,3,4,5)\}$.



The actions can be formally described as follows: given a state $\{(b_{1,1},\ldots,b_{1,n_1}),\ldots,(b_{p,1},\ldots,b_{p,n_p})\}$, where p denotes the number of piles ($1 \leq p \leq 5$) and n_k the number of blocks in the k-th pile ($n_k \geq 1$ for each k, and $\sum_{k=1}^{p} n_k = 5$), the actions consist of moving one of the p blocks $b_{k,1}$ (on the top of one of the piles) either to the table, thus generating a new pile (only if the original pile contains more than one block, i.e., $n_k > 1$), or on top of one of the other $p-1$ piles (if any).

Actions can be implemented by a *successor function SF*: it receives as an argument the description s of a state, and returns all the pairs (s', a) where s' is one of the states obtained as described above, and a the description of the corresponding action. Each action can be described by indicating the number of the block that has been moved, $b_{k,1}$, and its new position, i.e., the number of the block on top of which it has been placed, or the table (which can be denoted by the number 0). For instance, from state $\{(2,5,3),(1,4)\}$ the action $(2,0)$ consists of moving block 2 to the table, leading to the state $\{(5,3),(2),(1,4)\}$; similarly, action $(1,2)$ consists of moving block 1 on top of block 2, leading to the state $\{(1,2,5,3),(4)\}$.

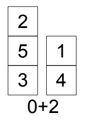
Finally, since the goal is to reach the target configuration by moving the smallest possible number of blocks, it follows that the path cost is given by the number of actions that have been carried out to reach a given state from the initial one, and thus each action has a cost equal to 1.

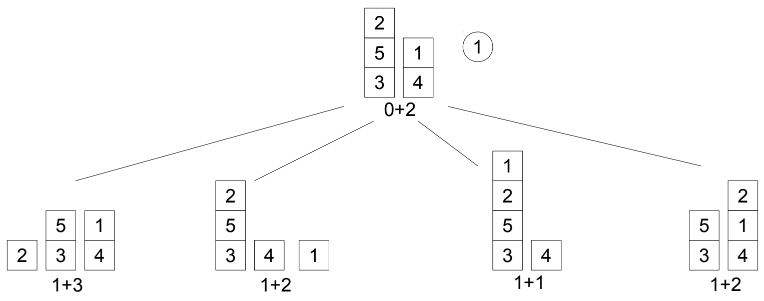
[1]According to the problem definition, the ordering between piles is not relevant.

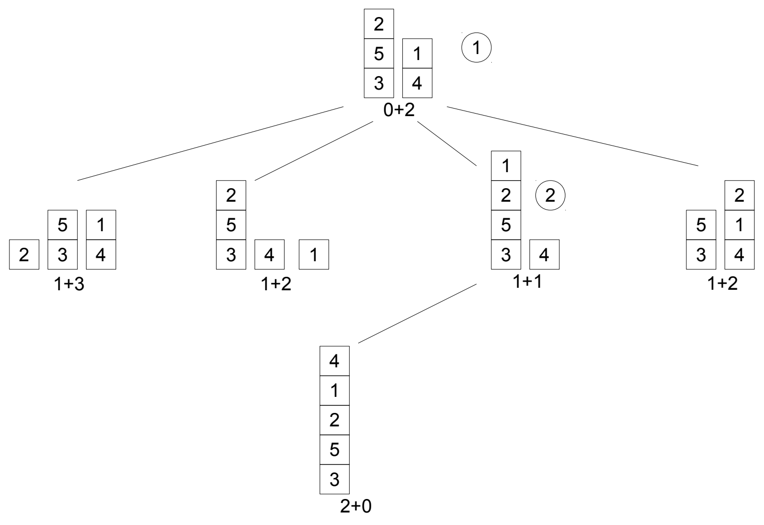
(b) The search tree obtained by the A* search strategy after the expansion of the first four nodes, using the heuristic given in the text and avoiding repeated states, is shown in the figure below. For the sake of clarity, the tree obtained after the expansion of each node is separately shown.

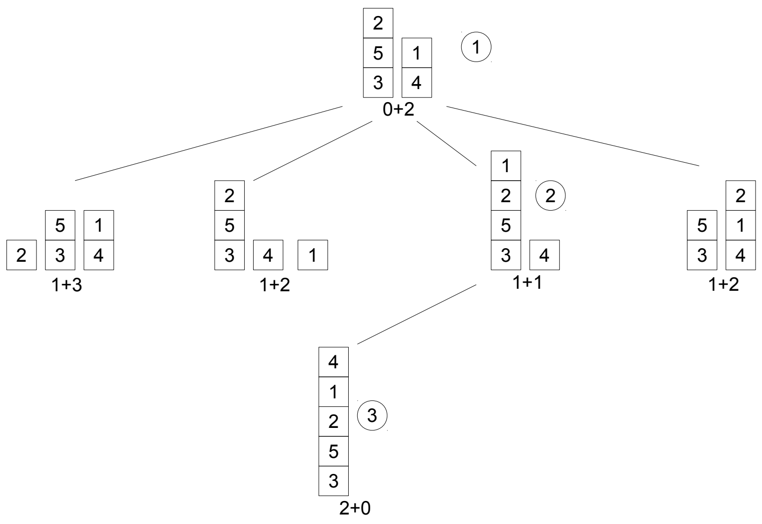
The A* evaluation function (path cost + heuristic) is shown below each node. The numbers inside circles denote the order in which the corresponding node has been expanded. Note that the third node has no successors, since the only action that can be carried out on its state leads to a repeated state (the same as the parent node). Note also that the are two candidate leaf nodes to be expanded at the fourth iteration of A* (the second and fourth node from left in the figure, at depth 1), since they exhibit the same value of the evaluation function, which is lower than the one of the remaining leaf nodes; in this case a random choice has to be made between the candidate nodes: in the figure below the node on the right is chosen for expansion.

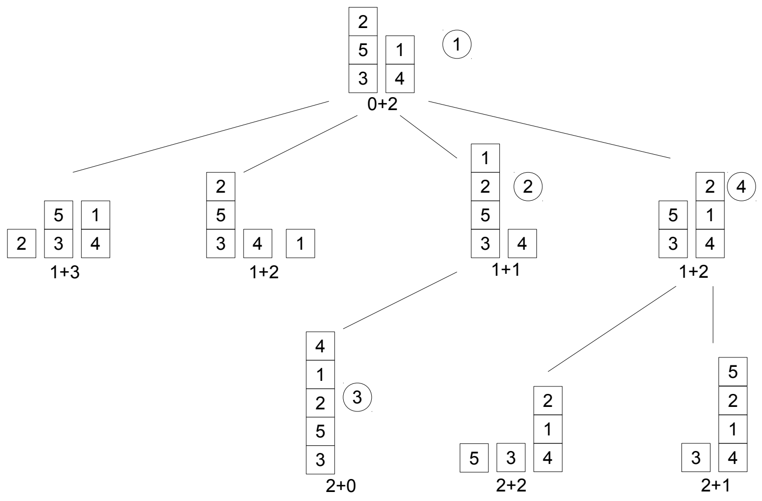
root node:











(c) Another possible admissible heuristic is the following: let n be the number of blocks on top of which there is a block different from the one in the goal state; then the heuristic is defined as $\lfloor n/2 \rfloor$. For instance, assuming that the goal state is $\{(1,2,3,4,5)\}$, the heuristic for state $\{(2,5,3),(1,4)\}$ is computed as follows:

- there is no block on the top of block 1, as in the goal state: this counts as 0;
- there is no block on the top of block 2, contrary to the goal state: this counts as 1;
- block 5, instead of block 2, is on the top of block 3: this counts as 1;
- block 1, instead of block 3, is on the top of block 4: this counts as 1;
- block 2, instead of block 4, is on the top of block 5: this counts as 1;

one therefore obtains: $\lfloor 4/2 \rfloor = 2$; in other words, *at least two* blocks have to be moved to reach the goal state.

It is not difficult to see that this heuristic is admissible: by moving one block from any state s', in the resulting state s'' at most two other blocks will have a different block on the top of them, and therefore at most two more blocks than in s' can have the correct block on the top of them, as in the goal state.

Exercise 4

A* selects for expansion the leaf node n with the lowest value of the evaluation function $f(n) = g(n)+h(n)$, where g and h denote the path cost of n (i.e., the sum of of the step costs – costs of the actions – in the path from the root node to n) and the value of the heuristic function for the corresponding state, respectively. Ties are broken randomly. It immediately follows that A* will select (randomly) either the left-most or the right-most leaf node, whose evaluation function equals 5.

Exercise 5

The search strategy used to obtain the considered search tree cannot be depth-first: indeed, after the expansion of the root node, either node `C` or node `D` has been expanded in the second step; in the former case, either node `E` or node `F` would have been expanded in the third step, instead of node `D`; analogously, in the latter case either node `G` or node `H` would have been expanded in the third step, instead of node `C`.

The search strategy cannot be uniform-cost, either: indeed, in the second step node `D` (the one with the lowest path cost among `B`, `C` and `D`) would have been expanded, but in the third step uniform-cost would have expanded node `G` instead of `C`, whose path cost (6) is lower than the ones of nodes `B` (9), `C` (7) and `H` (8).

Exercise 6

A heuristic function is admissible if it never overestimates the minimum cost from a state to the goal state. It is easy to see that this condition is *not* fulfilled by the considered heuristic. The minimum path cost from state `A` to `D` is 3, which corresponds to the path `A` $\rightarrow$ `B` $\rightarrow$ `D`, whereas the heuristic of `A` is 4, and therefore it *overestimates* the minimum path cost from `A` to `D`.