University of Cagliari

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

Artificial Intelligence

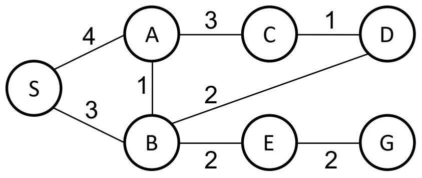
Academic Year: 2025/2026

Instructors: Ambra Demontis, Giorgio Fumera

Exercises on search algorithms

1. Consider the state space of a graph search problem shown below. States are denoted by letters. 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 goal states. The table on the right shows the values of an *admissible* heuristic function (i.e., it never overestimates the minimum path cost from any given state to the goal state G).

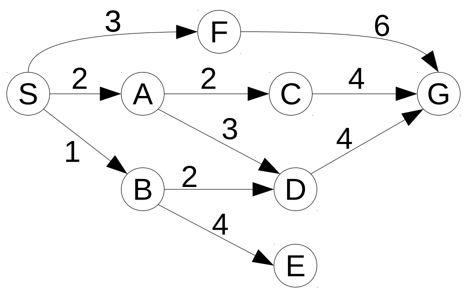
   Show how the depth-first and A* search strategies solve this problem, avoiding repeated states in the same path. In case of a tie, the choice of the next node to expand has to be made according to *alphabetical ordering*; for A*, in case of a further tie, the *deepest* node should be selected. For the sake of clarity, draw *separately* the search tree obtained after each expansion step, and *clearly* indicate the expanded node (e.g., by numbering the expanded nodes according to the order of their expansion). For A* also indicate the path cost and the value of the heuristic function of each node. For depth-first search, pay attention to the nodes that should be deleted from search tree (if any).



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

2. The graph in the figure below shows the state space of a search problem whose actions are not reversible, as indicated by the fact that the graph is *oriented* (e.g., it is possible to move from state S to A, but not vice versa). The table on the right shows the value of an admissible heuristic function, considering G as the goal state.

state space



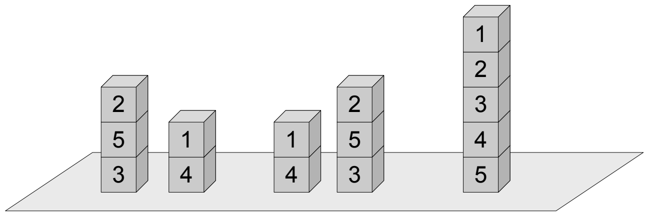
heuristic function (goal state: G)

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 breadth-first search, depth-first search and A*, drawing the corresponding search trees as indicated in the previous exercise (breaking ties, if any, following the alphabetical order), and showing also the path cost of each node for breadth-first and depth-first search.

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.

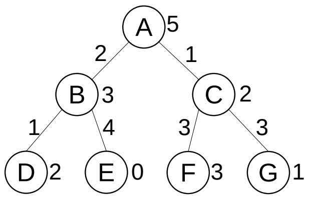
   Consider a simple version of this problem, in which there are five blocks with identical shape and size, numbered from 1 to 5. The figure below shows three possible configurations of such blocks; note that the relative position of the piles does not matter, thus the configuration on the left is equivalent to the one in the middle. Every block can be either on the table or on the top of another block, so there can be one to five piles. Starting from any given configuration, 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.



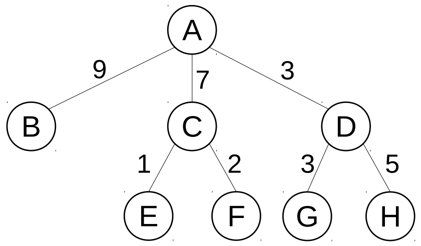
   (a) Formulate the above version of the block's world as a search problem, by *precisely* defining the state space, the initial state, the goal test, the set of possible actions and the path cost.

   (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 A*, avoiding repeated states, and breaking ties (if any) randomly. As in the previous exercises, when drawing the search tree you should *clearly* indicate the order of expansion of each node, the path cost and the value of the heuristic of each node.

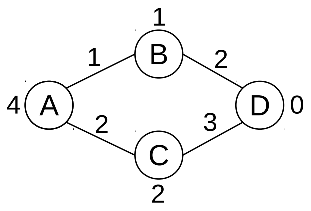
4. Assume that the search tree shown below has been obtained for some search problem after the first three expansion steps by A*. Numbers on the edges denote the cost of the corresponding actions, letters inside each node denote the corresponding states, and numbers next to each node denote the value 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?


$$\begin{tikzcd}
 & & & A\ 5 \arrow[dll, "2"'] \arrow[drr, "1"] & & & \\
 & B\ 3 \arrow[dl, "1"'] \arrow[dr, "4"] & & & & C\ 2 \arrow[dl, "3"'] \arrow[dr, "3"] & \\
D\ 2 & & E\ 0 & & F\ 3 & & G\ 1
\end{tikzcd}$$

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, both of them, or none of them, clearly motivating your answer.



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*? (Give a *detailed* motivation of your answer.)



Additional questions

(a) When is a search strategy *optimal*? Under what condition is A* optimal?

(b) When is a search strategy *complete*? Is Breadth-first search complete? Why?

(c) What are the *time* and *space complexity* of graph search algorithms? What is the "unit of measurement" that is used to express them?

(d) Explain why the worst-case *time complexity* of Breadth-first search is $\mathcal{O}(2^{n+1})$, with reference to a hypothetical search problem with constant branching factor $b = 2$ (i.e., each node has exactly two successors), and shallowest solution(s) at depth n.

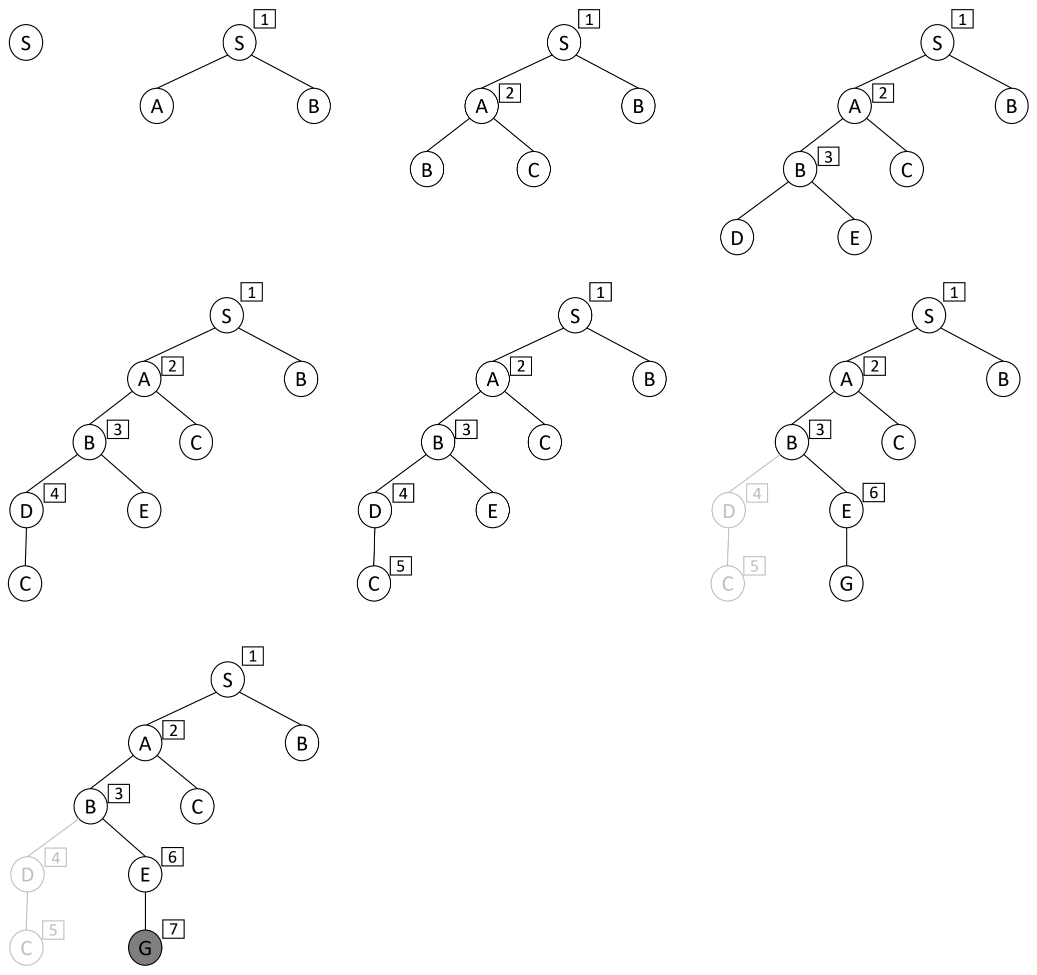
(e) Explain why Depth-first search has $\mathcal{O}(d)$ worst-case *space complexity*, with reference to a hypothetical problem with constant branching factor (i.e., all nodes have the same number of successors), maximum depth equal to d, and all the solutions at the same depth d.

(f) Under what condition is a heuristic function *admissible*? Can you make an example of a non-admissible heuristic for the search problem of exercise 1, by modifying a *single* value in the corresponding table? What is the advantage of using an admissible heuristic in the A* search strategy?

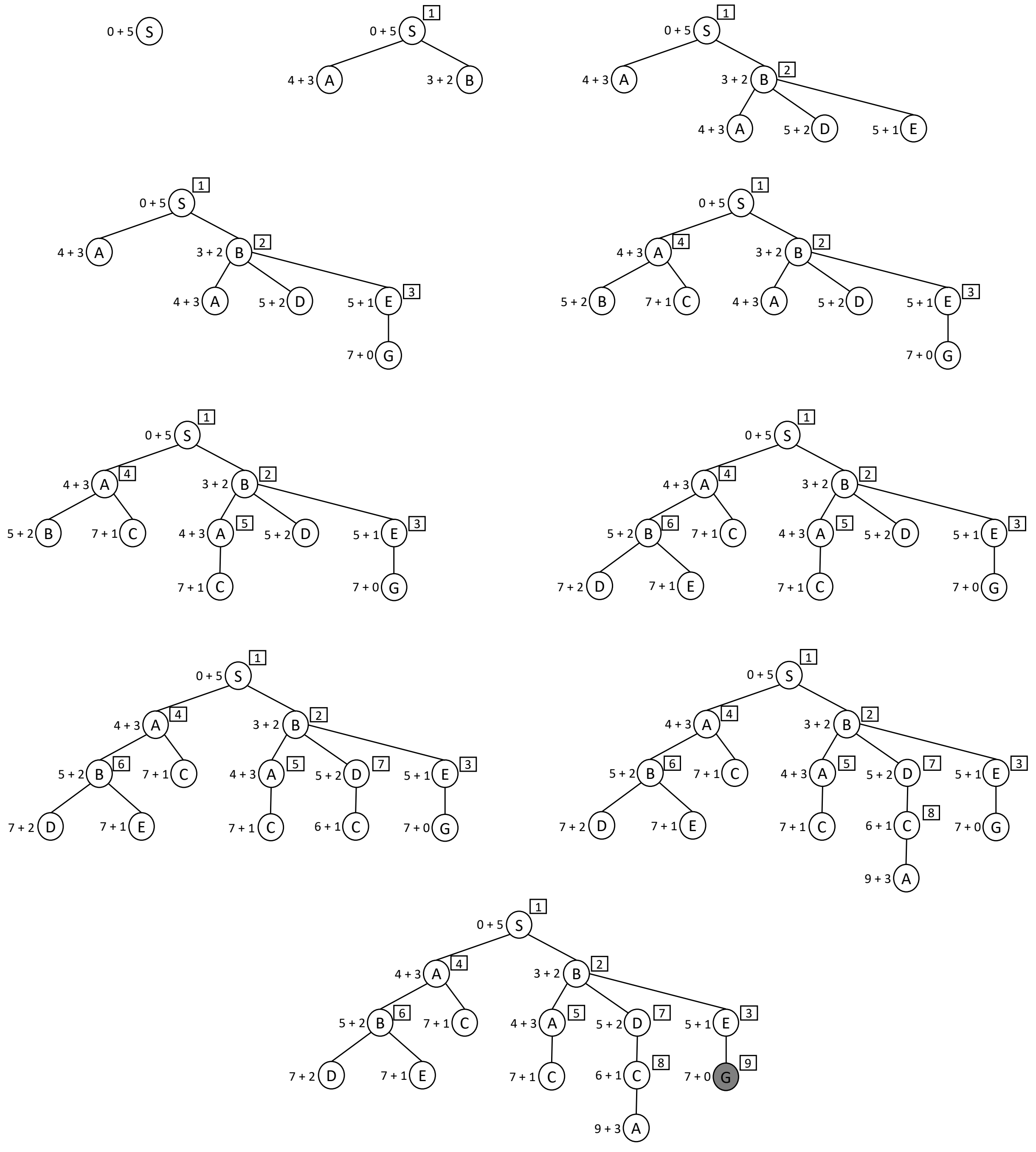
Solution of exercises

Exercise 1

The search tree built by depth-first search is shown below. The expansion steps are indicated by numbers inside squares, next to the corresponding nodes. Note that in step $\boxed{5}$ no successor is found, since all the states reachable from `C` (i.e., `A` and `D`) are already present in the path from the root node; therefore, before proceeding with the next expansion step, the corresponding node is deleted from the search tree, as well as its parent node, since at this point the parent node itself has no successors anymore; deleted nodes are shaded in the next partial trees. Note also that, according to the general search algorithm, a solution is found at step $\boxed{7}$, when a node containing the goal state is *selected* for expansion, not at step $\boxed{6}$, when it is *generated* by expanding its parent node.



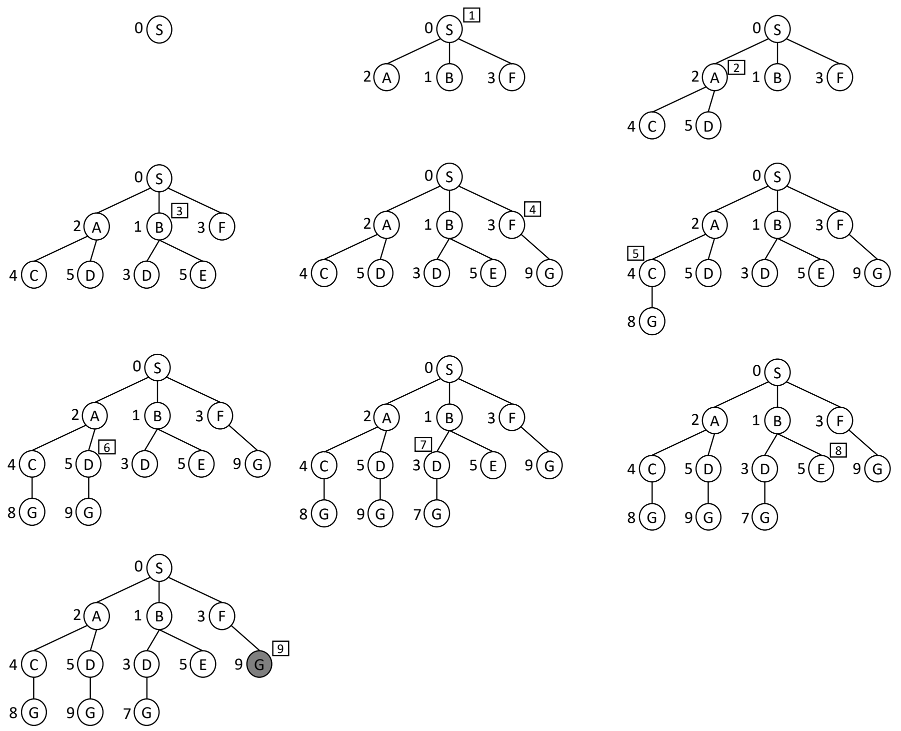
The search tree built by A* is shown below. On the left of each node the sum of the corresponding path cost g and heuristic function h is shown as $g + h$. Note that, as required in the text, in case of ties between nodes corresponding to the same state, the *shallowest* one has been selected for expansion.



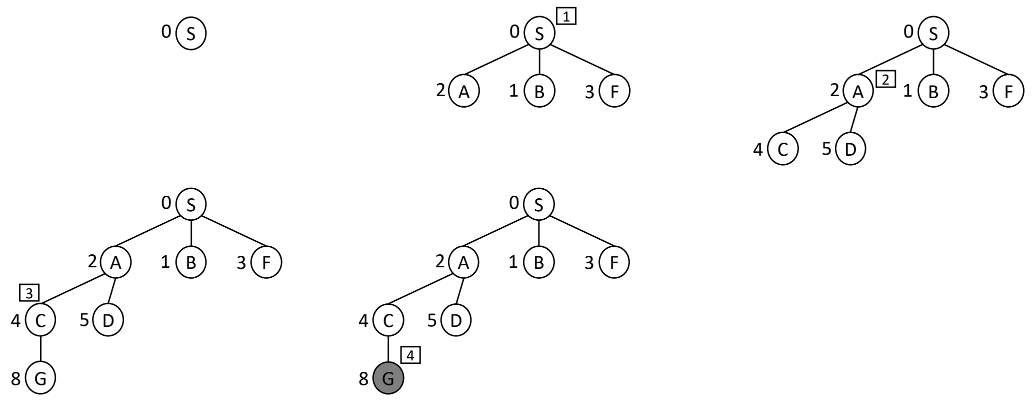
Exercise 2

In the search trees shown below the value on the left of each node is the path cost, for breadth-first and depth-first search, and the evaluation function (path cost plus heuristic function) for A*. The order of expansion is denoted by numbers inside squares, as in the previous exercises.

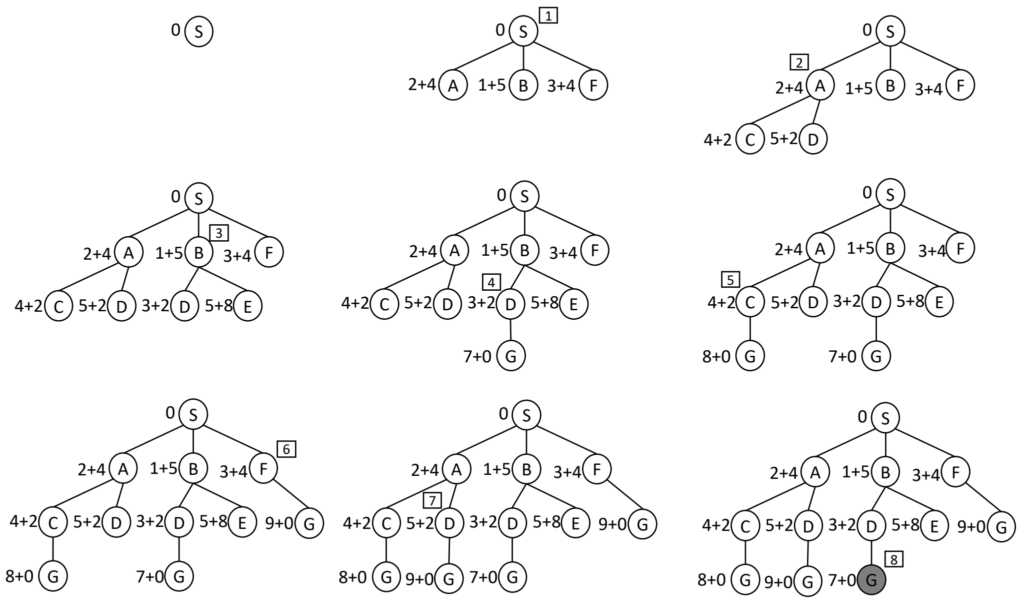
The search tree built by breadth-first search is shown below. According to the state space the state `E` has no successors (i.e., no other state can be reached from it): therefore, when a node corresponding to `E` is selected for expansion (step $\boxed{8}$) no leaf nodes are added to the tree.



The search tree built by depth-first search is shown below. In this case a solution is found along the first path which is explored, therefore no backtracking step is carried out, and none of the already expanded nodes is deleted.

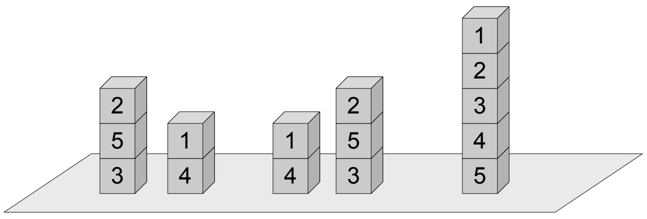


Finally, the search tree built by A* is shown below. Note that the solution found by A* is optimal (which is guaranteed, when an admissible heuristic is used), contary to the ones found by breadth-first and depth-first search (see above); note also that after *generating* the optimal solution in step $\boxed{4}$, A* *generated* also a suboptimal one in step $\boxed{7}$; obviously the latter was *not* selected as the solution to return since its evaluation function is higher than that of the optimal solution.



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 blocks 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* that 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

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

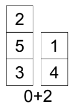
top of which it has been placed, or the table (which can be denoted by the number 0). For instance, from state $\{(2,5,3),(1,4)\}$ the action $(2,0)$ consists of moving block 2 to the table, leading to the state $\{(5,3),(2),(1,4)\}$; similarly, action $(1,2)$ consists of moving block 1 on top of block 2, leading to the state $\{(1,2,5,3),(4)\}$.

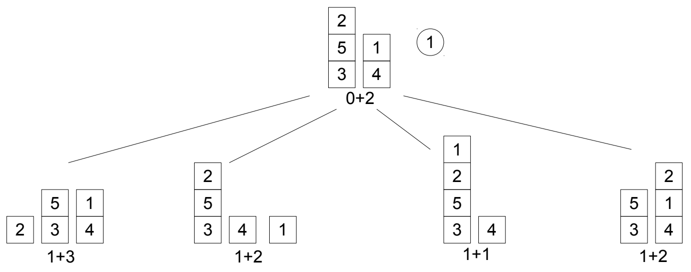
Finally, since the goal is to reach the target configuration by moving the smallest number of blocks, it follows that the path cost is given by the number of actions that have been carried out to reach a given state from the initial one, and thus each action has a cost equal to 1.

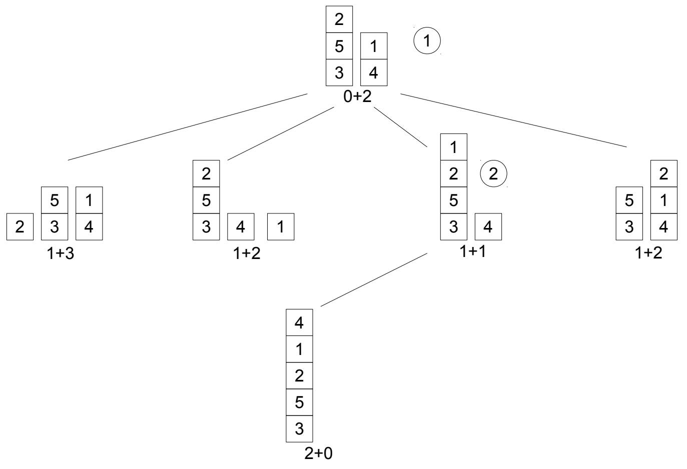
(b) The search tree obtained by A* after the expansion of the first four nodes, using the heuristic given in the text and avoiding repeated states, is shown in the figure below. For the sake of clarity, the tree obtained after the expansion of each node is separately shown.

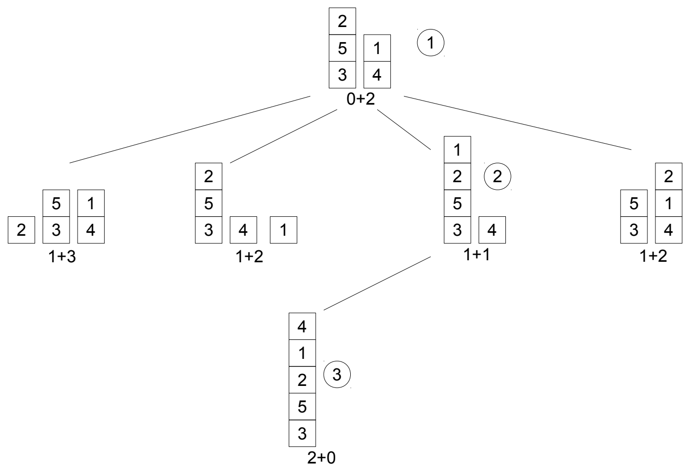
The A* evaluation function (path cost + heuristic) is shown below each node. The numbers inside circles denote the order of the expansion steps. Note that the node selected for expansion at step 3 has no successors, since the only action that can be carried out on its state leads to a repeated state (the same as the parent node). At step 4 the tie between the two nodes with the same, minimum value of the evaluation function is broken by choosing for expansion the one on the right. Finally, note that a solution has not yet been found, after four expansion steps.

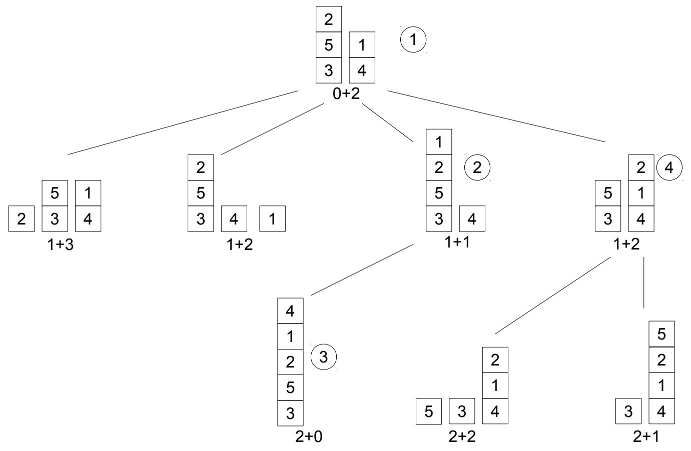
root node:











Exercise 4

A* selects for expansion the leaf node n (or one of the leaf nodes) with the lowest value of the evaluation function $f(n) = g(n) + h(n)$, where $g(n)$ and $h(n)$ denote respectively 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. 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-cost path from state `A` to `D` is `A` $\rightarrow$ `B` $\rightarrow$ `D`, whose exact cost is 3, whereas the heuristic value of state `A` is 4, which *overestimates* the minimum path cost.