Exercice 1 :

Formalize the following problems as state-space problems: define a suitable state space, a start state, set of actions, the action function, and a goal test.

a. In the movie *Die hard: With a vengeance*, Bruce Willis and Samuel L. Jackson are given a water jug riddle, which they need to solve in order to disarm a bomb: There is a water fountain and two jugs that can hold 3 and 5 liters of water, respectively. How do you measure up 4 gallons of water (to disarm the bomb) using only the two jugs?

b. (From Russel & Norvig, Exercise 3.9): The *missionaries and cannibals problem* is usually stated as follows: Three missionaries and three cannibals are on one side of a river, along with a boat that can hold either one or two people. Find a way to get everyone to the other side of the river without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place.

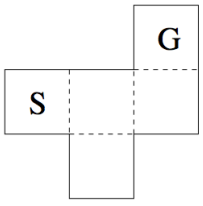
Solution:

a. States are the possible filling amounts (in full gallons) of the two jugs. For example, $(2, 0)$ is the state corresponding to the case where the first jug contains 2 gallons and the second jug contains 0 gallons. The start state is $(0, 0)$. The 'fill second jug' action leads to the state $(0, 5)$ from which the state $(3, 2)$ can be reached by the action 'fill first jug from second jug'. A goal state is any state where the second jug contains 4 gallons.

b. Here is one possible representation: A state is a six-tuple of integers listing the number of missionaries, cannibals, and boats on the first side, and then the second side of the river. The goal is a state with 3 missionaries and 3 cannibals on the second side. The cost function is one per action, and the successors of a state are all the states that move 1 or 2 people and 1 boat from one side to another.

Exercice 2 :

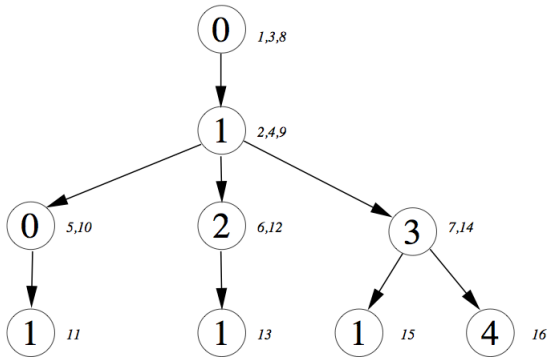
Consider the following robot navigation problem where we want to go from S to G:



- Draw the search tree for this problem (or as much of the search tree as is needed to answer the following:
- Show how iterative deepening search will solve this problem: show the order in which nodes of the search tree will be selected and expanded.

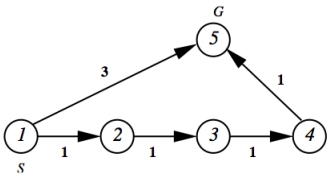
Solution:

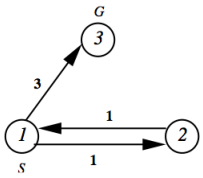
The numbers inside the nodes correspond to the node numbers in the solution to the previous exercise. We assume that branches are explored from left to right. The small numbers beside the nodes then give the order in which nodes are expanded when performing iterative deepening search.



Exercice 3 :

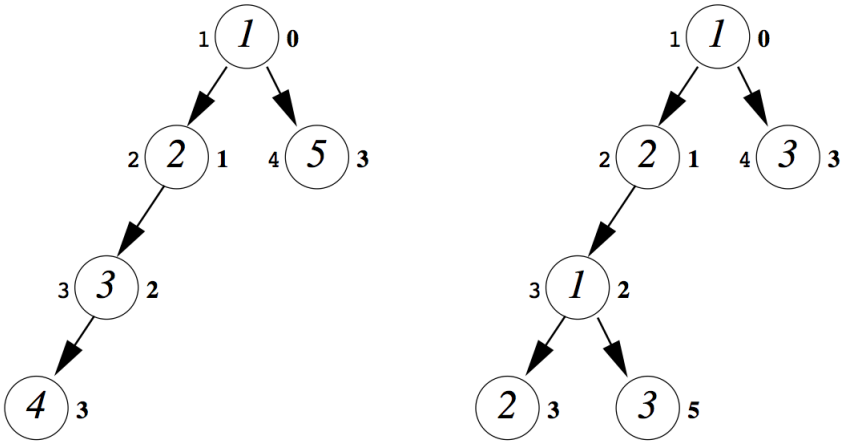
Consider the following two state spaces:





- Show how Uniform-Cost Search will find for the two problems below an optimal path from start state S to goal state G (draw the relevant part of the search tree, and indicate in which order nodes are expanded).
- For each node n in the two graphs of the exercise above determine the cost $h^*(n)$ of an optimal solution starting from that node.
- Show how A^* finds the optimal solutions for these two problems when $h(n) = h^*(n)$.

Solution: Bold numbers to the right of the nodes show the cost associated with the node. Typewriter font numbers to the left show the order in which the nodes are chosen for expansion. Since the cost of both the frontier nodes at the end is 3, the choice between them is arbitrary. The algorithm might also choose to expand the left branch by one more node, before it finds the optimal path by going down the right branch.



Costs for the left graph (node number n : $h^*(n)$):

$$1:3, 2:3, 3:2, 4:1, 5:0$$

Costs for the right graph (node number n : $h^*(n)$):

$$1:3, 2:4, 3:0$$

The bold numbers now show the sum $f(n) = g(n) + h^*(n)$. A^* chooses the node with the lowest $f(n)$ value. Thus, the optimal path (down the right branch) is immediately found.

Exercice 4 :

Consider the state space from Figure 1.

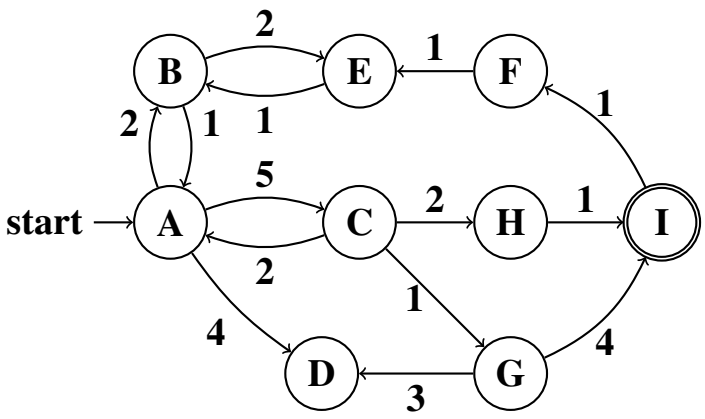


Figure 1: State space.

As a heuristic estimate for a state s, use the minimal number of edges that are needed to reach a goal state from s (or ∞ if s is not solvable, e.g., $h(C) = 2$).

As tie-breaking criteria, if the choice of the next state to be expanded is not unique, expand the lexicographically smallest state first.

(i) Run Uniform-cost-search on this problem. Draw the search graph and **annotate each node with the g value as well as the order of expansion**. Draw duplicate nodes as well, and mark them accordingly by crossing them out. Give the solution found. Is this solution optimal? Justify your answer.

(ii) Run A^* search on this problem. Draw the search graph and **annotate each node with the g and h value as well as the order of expansion**. Draw duplicate nodes as well, and mark them accordingly by crossing them out. Give the solution found by A^* search. Is this solution optimal? Justify your answer.

(iii) Run the hill climbing algorithm, as stated in the lecture, on this problem. For each state, provide all applicable actions and the states reachable using these actions. **Annotate states with their heuristic value. Specify which node is expanded in each iteration of the algorithm.** If the choice of the next state to be expanded is not unique, expand the lexicographically smallest state first. Does the algorithm find a solution? If yes, what is it and is it optimal?

(iv) Could hill-climbing stop in a local minimum without finding a solution? If yes, give an example heuristic $h : \{A, B, ..., I\} \to \mathbb{N}_0^+ \cup \{\infty\}$ for the state space depicted in Figure 1 that leads hill-climbing into a local minimum, and explain what happens. If no, please explain why.

Solution:

(i) The solution is A, C, H, I. Uniform-cost search is guaranteed to return an optimal solution if there are no negative costs.

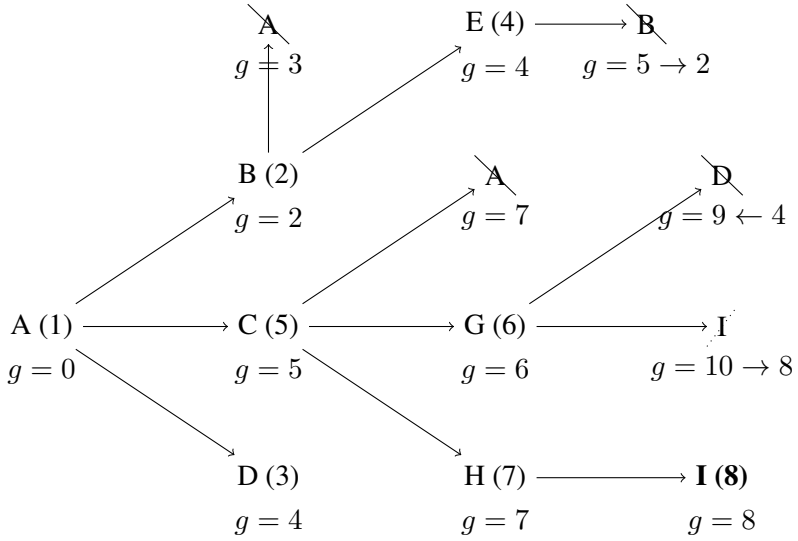


Figure 2: Solution to part (i). The algorithm stops when the goal node I is selected for expansion.

(ii) The solution is A, C, H, I. A^* is guaranteed to return an optimal solution if the heuristic that is being used is consistent. Thus, it remains to show that the described heuristic h is consistent, i.e., that for every transition $s \xrightarrow{a} s'$, it holds that $h(s) - h(s') \leq c(a)$. By definition of our heuristic the following holds: $\forall s, s' \in S \setminus \{D\} : s \xrightarrow{a} s' \implies |h(s) - h(s')| = 1$. That means that the difference in heuristic values between any state s and any successor state s' of s is maximally 1, except for state D. In the special case of state D being the successor state, $h(s) - h(s') = -\infty$, which is less than the cost of any action (since D is a dead end it does not have any successors). Otherwise the minimal cost for any transaction is 1, so the heuristic is consistent.

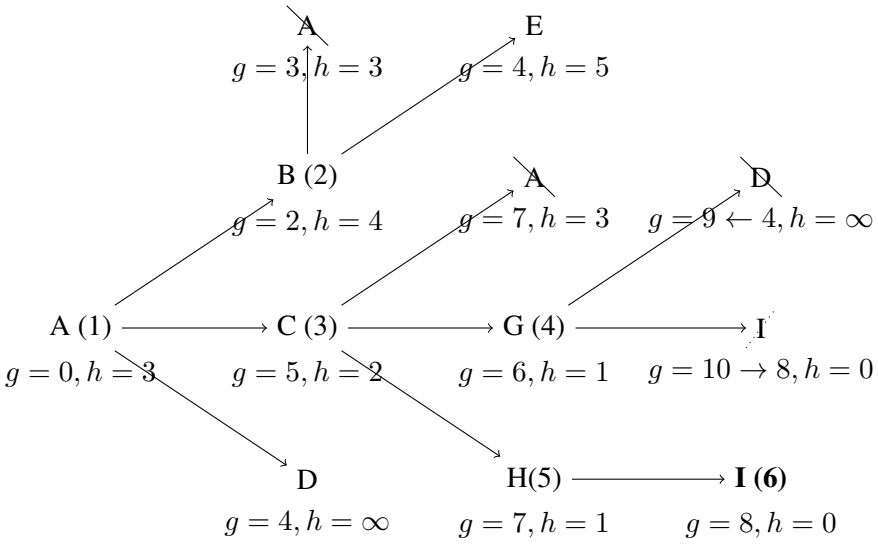


Figure 3: Solution to part (ii). The algorithm stops when the goal node I is selected for expansion.

(iii) Hill climbing returns the solution A, C, G, I, which is not optimal. The expansion order is as follows:

- State A
  $A \to B$: $h(B) = 4$
  $A \to C$: $h(C) = 2 \Leftarrow$ expand
  $A \to D$: $h(D) = \infty$
- State C
  $C \to A$: $h(A) = 3$
  $C \to G$: $h(G) = 1 \Leftarrow$ expand
  $C \to H$: $h(H) = 1$
- State G
  $G \to D$: $h(D) = \infty$
  $G \to I$: $h(I) = 0 \Leftarrow$ expand
- State I
  $I \to F$: $h(F) = 6$
- I is a local minimum, search stops.

(iv) Yes it is possible that hill-climbing stops without finding a solution. Consider the following heuristic:

State s	A	B	C	D	E	F	G	H	I
h(s)	2	2	2	2	2	2	2	2	0

Hill climbing starts at A (with $h(A) = 2$) and chooses randomly one of its child nodes (B, C, or D). Since either one of them has an equal heuristic value of 2, Hill Climbing stays at node A, which is then returned as a local minimum without finding the solution.

Exercice 5 :

Consider a variant of the Vacuum Cleaner problem from the lecture where a robot has to clean a 3×3 square room (see Figure 4). There are four possible actions: up, left, right, and down. There is no suck action since the robot automatically cleans any dirty spot it stands on. Hence, its task is moving around the room and visit all dirty spots. Throughout the exercise, we use Manhattan distance.

Which of the following heuristics are admissible? Why / Why not? Give clear proof arguments. (Formal proofs are not needed.)

(i) $h_1 =$ Number of dirty spots.

(ii) $h_2 =$ Number of clean spots.

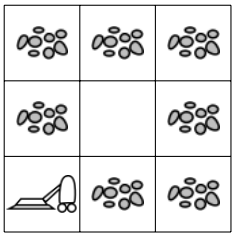


Figure 4: Illustration of the Vacuum Cleaner problem

(iii) $h_3 =$ Sum of the distances from the robot to all the dirty spots.

(iv) $h_4 = h_1 + h_2$.

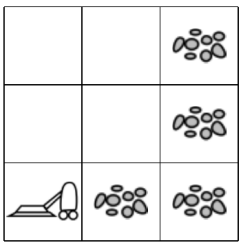
(v) $h_5 =$ Minimum distance from the robot to any dirty spot.

Note: To prove a heuristic admissible, you need to show that it is admissible for every state of the illustrated 3×3 example. To show that a heuristic is not admissible, a counter example is sufficient.

Solution:

(i) h_1: Admissible, because to clean one dirty spot it takes at least one action and one action can clean at most one spot. the robot needs to reach at least one dirty spot.

(ii) h_2: Not admissible (Not even a heuristic, since $h_2(G) \neq 0$)). Consider the following counter example:



Here, $h_2 = 5$, while $h^* = 4$, which makes it not admissible.

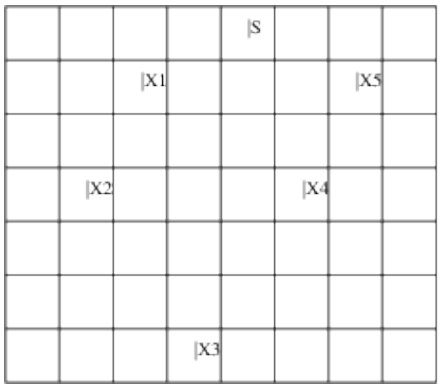
(iii) h_3: Not admissible. Consider the following counterexample, where $h_3 = 1 + 2 = 3 > 2 = h^*$ (assume that the rest of the board has already been cleaned):

(iv) h_4: Not admissible. Consider again the counter example from h_3. It is $h_5 = h_1 + h_2 = 2 + 1 = 3$, but $h^* = 2$. (Since it incorporates h_2 which is not a heuristic, the combination is also not admissible).

(v) h_5: Admissible, because the robot needs to reach at least one dirty spot.

Exercice 6 :

Previous exam question: Consider a robot that can move in the grid below. The cost of moving from one cell to another is equal to the number of steps required for the move, where by a single step the robot can move horizontally or vertically to an adjacent cell (no diagonal steps allowed).



The cells marked X_1, X_2, X_3, X_4, and X_5 are *cells of interest*, each holding one or two of the symbols a, b, and c:

- X_1: (a, c)
- X_2: (b)
- X_3: (b, c)
- X_4: (c)
- X_5: (a)

That is, X_1 holds the symbols a and c. When visiting a cell the robot collects the symbols located at that cell.

Starting in cell S, the robot's task is to visit cells of interest and collect exactly one copy of each of the three symbols (a, b, c) before returning to cell S. The robot is *not* allowed to move to a cell containing, e.g., an a if the robot has previously visited a cell holding an a.

Determine which of the cells of interest to visit and in which order these cells should be visited by the robot, so that the cost of the path traveled (when starting and ending in cell S) is minimized.

1. Define a suitable state and action representation for this problem.
2. Define a heuristic function that at any given state provides an underestimate of the cost of reaching a goal state (i.e., a state where the robot is located in cell S with the symbols (a, b, c)).
3. Show the search tree for this problem as it would be expanded using A* search.

Solution:

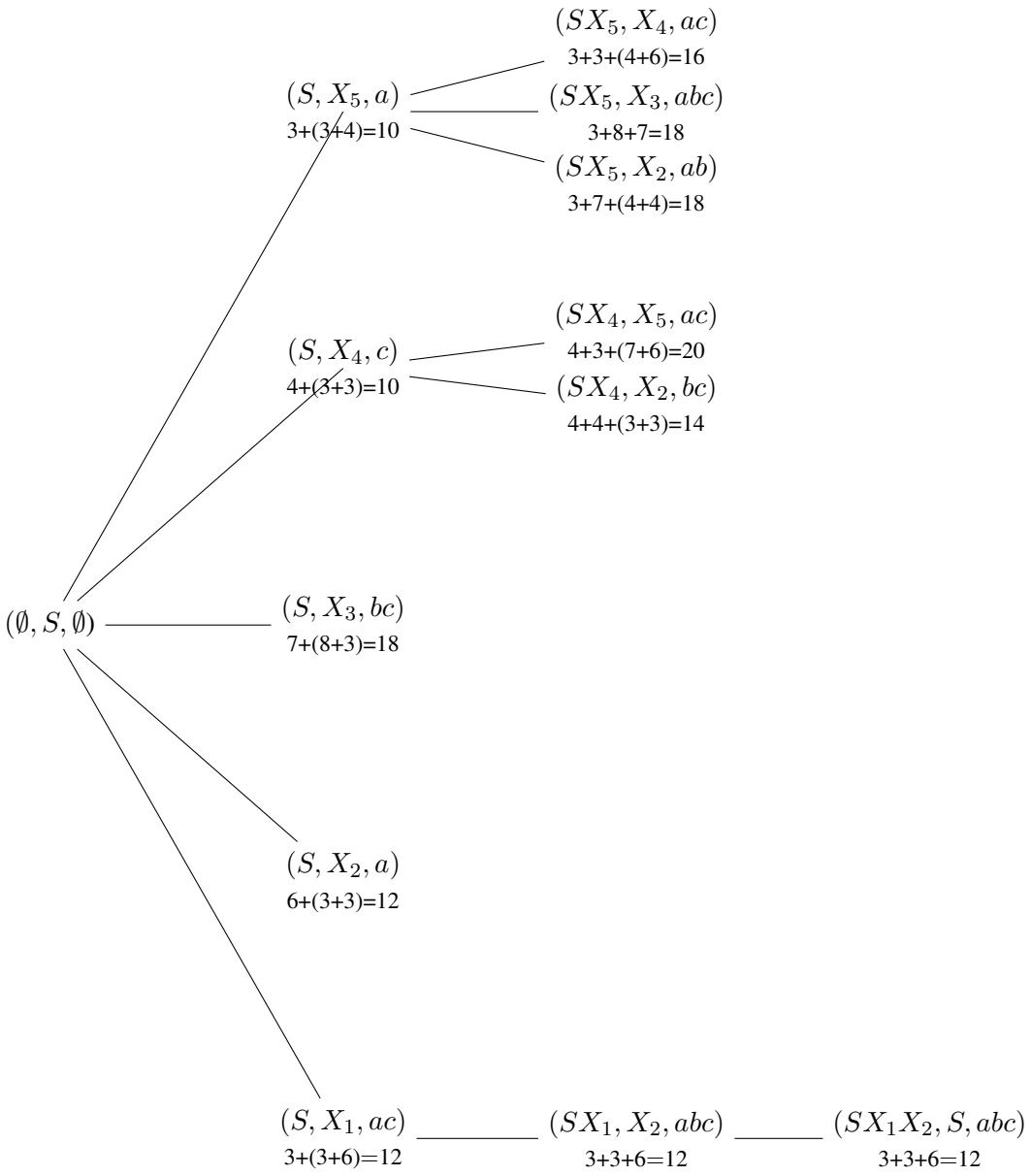
Sub-problem 1

One possible state representation could be $(\textit{cell history}, \textit{current cell}, \textit{symbols cllected})$, where we keep track of the cells previously visited, the current cell the robot is in, and which symbols that have been collected so far. We can, however, also do without including the cell history in the state representation.

Sub-problem 2

One possible heuristic function could be the minimal cost incurred by visiting an admissible cell holding a symbol not yet collected and afterwards returning to S.

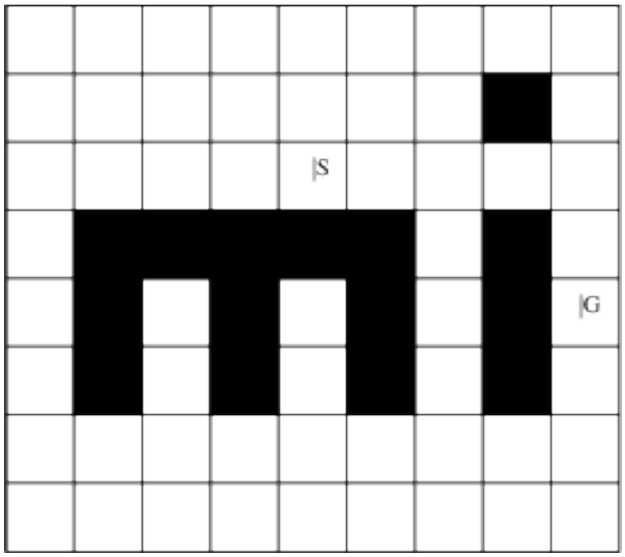
Sub-problem 3



Exercice 7 :

Previous exam question: Consider the problem of finding a path from position S to G in the grid shown below. You can only move horizontally and vertically and only one step at a time; no step can be made into

the shaded areas or outside the grid. The cost of the path between two positions is the number of steps on the path.



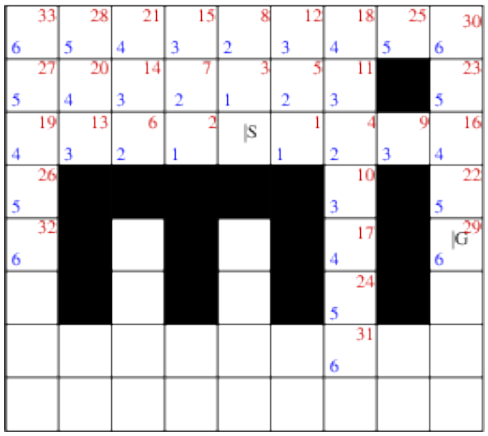
1. Number the positions (cells in the grid) in the order in which they are added to the frontier in a uniform-cost search for G starting at S. Assume that the ordering of the operators is *right*, *down*, *left*, and *up*, and that there is multiple path pruning. When multiple lowest-cost paths exists, choose the path first added to the frontier.
2. Define a heuristic function that underestimates the cost of the lowest-cost path.
3. Number the positions (cells in the grid) in the order in which they are added to the frontier according to an A^* search using your heuristic function. Resolve ambiguities using the operator ordering and the procedure above.

NB: When numbering the positions it may be helpful to also annotate the positions with the cost/function values calculated during the search.

Solution:

Answer 1

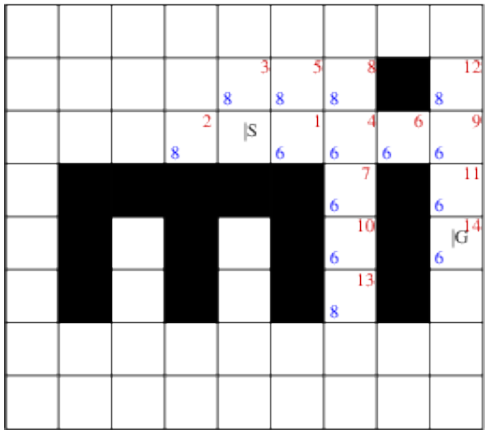
The red numbers specify the order in which the cells are added to the frontier. The blue numbers are the costs associated with the cells.



Answer 2

One possible heuristic function simply calculates the cost of the lowest-cost path ignoring the obstacles (shaded regions).

Answer 3

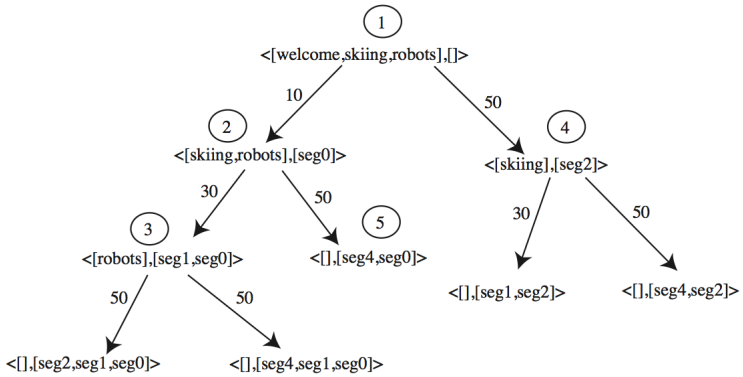


Exercice 8 :

Solve Exercise 3.4 in **PM**. Note that the book uses "monotone" instead of "consistent".

Solution:

a. The nodes are expanded according to the circled numbers. The circled (5) is the first goal node found - this is the shortest presentation that covers all the topics.



- Here are two solutions:

  Solution 1: For each topic, t, let $s(t)$ be the length of the smallest segment that covers topic t. Let $h(\langle TC, Segs\rangle) = \max_{t \in TC} s(t)$. That is, we find the topic t for which $s(t)$ is maximum.

  Solution 2: For each segment let the contribution of the segment be the time of the segment divided by the number of topics the segment covers. For each topic, t, let $s(t)$ be the smallest contribution for all of the segments that covers the topic. Let $h(\langle TC, Segs\rangle) = \sum_{t \in TC} s(t)$. That is, we sum $s(t)$ for all of the topics t in TC. The intuition is that each topic t requires at least $s(t)$ time. Note that we need to divide by the number of topics the segment covers to make sure that we do not double count the time for segments added that cover multiple topics.

  Both of these solution require one pass through the segment database to build the $s(t)$ function, but once this is built, the heuristic function can be computed in time proportional to the length of the To cover list.

b. No, consistency implies admissibility but no viceversa. In the tree from the solution above, consider a heuristic where $h(1) = 60$ and $h(2) = 0$, this is admissible but inconsistent.

Exercice 9 :

You are planning a dinner for three guests. The menu should consist of at least one appetizer, exactly one main dish, and at least one desert (multiple appetizers or deserts are o.k.). You have a list of candidate dishes, and for each guest you know whether they like that dish or not:

Item	Cost	Guest 1	Guest 2	Guest 3
Appetizer 1	5			o.k.
Appetizer 2	5		o.k.	
Appetizer 3	15		o.k.	o.k.
Appetizer 4	30	o.k.		
Main dish 1	90		o.k.	o.k.
Main dish 2	100	o.k.		o.k.
Dessert 1	30		o.k.	
Dessert 2	50	o.k.	o.k.	

Your menu must contain for each guest at least one item that they like. Use A^* to find a minimal cost solution:

- Define the underlying state space problem
- Define a heuristic function that underestimates the true optimal cost function opt.
- Show how A^* will find the minimal cost solution using this heuristic function.

Solution:

- The states are all possible subsets of dishes: $\emptyset$, $\{A1\}, \{A2\}, \ldots, \{A1, A2, D1\}, \ldots$. The start state is $\emptyset$. Actions are of the form 'add dish X to the menu'. For example, 'add dish D1 to the menu' applied in state $\{A2\}$ leads to state $\{A2, D1\}$. Goal states are all states that contain at least one appetizer, exactly one main dish, at least one desert, and that contain for each guest one dish that guest likes. Thus, for example, a goal state must contain at least one of $A4$, $M2$, or $D3$ (to satisfy guest 1). Note that enumerating all goal states would be quite tedious, but all we need is a goal test, i.e. for a given state we must be able to decide whether it is a goal state.
- If a state is not yet a goal state, then at least one more dish must be added to the state. The cost to reach a goal state, thus is at least the cost of the cheapest dish not yet in the state. This would be a first possible heuristic function. A better heuristic function is obtained by considering what type of dishes are still lacking. Types can be 'appetizer', 'main', 'desert', 'liked by G1', 'liked by G2', 'liked by G3'. If the current state, for example, does not yet contain a desert, nor any dish liked by Guest 3, then the cost of reaching a goal state is at least the cheapest dish that is a desert, or liked by Guest 3. This heuristic function is better than the first, because it underestimates the true function opt not as much as the first function. Further refinements that provide yet closer approximations of opt can also be defined.
- First steps of A^* using the second heuristic function. The start state has 8 neighbors:

State n	$f(n) = cost(n) + h(n)$	State n	$f(n) = cost(n) + h(n)$
$\{A1\}$	$5 + 5$	$\{M1\}$	$90 + 5$
$\{A2\}$	$5 + 5$	$\{M2\}$	$110 + 5$
$\{A3\}$	$15 + 5$	$\{D1\}$	$30 + 5$
$\{A4\}$	$30 + 5$	$\{D2\}$	$50 + 5$

The heuristic function value is 5 for all states, because all states lack either an appetizer, or a dish liked by Guest 2, or a dish liked by Guest 3, all of which can be added in the form of $A1$ or $A2$, i.e. at the cost of 5.

A^* will now pick a state with minimal f value, which is either $\{A1\}$ or $\{A2\}$. Suppose we pick $\{A1\}$, we have the following neighbors:

State n	$f(n) = cost(n) + h(n)$	State n	$f(n) = cost(n) + h(n)$
$\{A1, A2\}$	$10 + 30$	$\{A1, M1\}$	$95 + 30$
$\{A1, A3\}$	$20 + 30$	$\{A1, M2\}$	$115 + 5$
$\{A1, A4\}$	$35 + 5$	$\{A1, D1\}$	$35 + 30$
		$\{A1, D2\}$	$50 + 5$

For $n = \{A1, A2\}$ we now have $cost(n) = 10$, and $h(n) = 30$, because n lacks a main dish, a desert, and a dish liked by Guest 1, and the cheapest dishes that correct one of these defects are $A4$ and $D1$, at the price of 30. In the next step $\{A1, A2\}$ or $\{A1, A4\}$ will be further expanded.