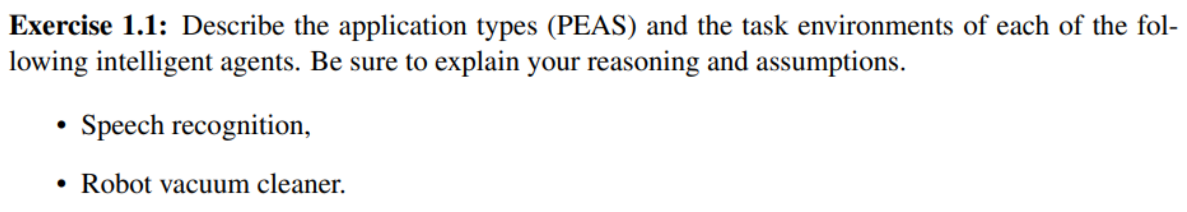
EKI

Table of Contents

Exercise 1.1



Exercise 1.1: Describe the application types (PEAS) and the task environments of each of the following intelligent agents. Be sure to explain your reasoning and assumptions.

- Speech recognition,
- Robot vacuum cleaner.

Speech Recognition:

- Performance Measure: Accuracy of transcribing spoken words into text.
- Environment: Audio input from microphone, with background noise and varying accents.
- Actuators: Display to verify the text in- and output.
- Sensors: Microphone or audio input devices.

Robot Vacuum Cleaner:

- Performance Measure: Cleanliness of the environment (coverage, dirt collected, time taken).
- Environment: Physical indoor space with various surfaces and obstacles.
- Actuators: Wheels/tracks for movement, brushes/suction for cleaning, sensors for obstacle detection.
- Sensors: Proximity sensors, bumper sensors, wheel encoders, cameras/lidar/sonar for mapping.

Exercise 1.2

Exercise 1.2: Let $f(n) = c \cdot g(n) + d \cdot h(n)$ be an evaluation function, where c and d are constants.

(a) Define c, d, $h(\cdot)$, $g(\cdot)$ such that A* with this evaluation function acts as a breadth-first search.

(b) Define c, d, $h(\cdot)$, $g(\cdot)$ such that A* with this evaluation function acts as a depth-first search.

(c) Define c, d, $h(\cdot)$, $g(\cdot)$ such that A* with this evaluation function acts as a uniform cost search.

You may assume that nodes contain all the information that we discussed in the lecture.

Breadth-first search

c = 1
g(n) = depth(n)
d = 0
h(n) = 0
f(n) = 1*depth(n) + 0*0 = depth(n)

Depth-first search

c = -1
g(n) = depth(n)
d = 0
h(n) = 0
f(n) = -1*depth(n) + 0*0 = - depth(n)

Uniform-cost search

c = 1
g(n) = pathCost(n)
d = 0
h(n) = 0
f(n) = -1* pathCost(n) + 0*0 = pathCost(n)

Exercise 1.3

Exercise 1.3: Provide a proof or a counterexample for each of the following claims:

(a) Every consistent heuristic (with $h(G) = 0$ for each goal node) is admissible.

(b) Every admissible heuristic is consistent.

Please try to keep your counterexample(s) as simple as possible. The graph(s) you use should have no more than five nodes.

Useful definitions

Definition

A heuristics h is admissible, if for every node n the following holds:

1. $h(n) \leq h^*(n)$ where $h^*(n)$ is the true cost from n (h is "optimistic");
2. $h(n) \geq 0$;
3. $h(g) = 0$ for every goal g (follows from 1 and 2).

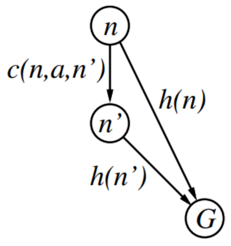
… it never overestimates the cost to reach the goal

Definition

A heuristic is consistent if, for every node n, every successor n' of n and every operator a,

$$h(n) \leq c(n, a, n') + h(n')$$

holds, where $c(n, a, n')$ are the path costs for a.



a) **Proof by Induction**

$Abbrevation: c = c(n, a, n')$
$c \ldots true\ cost\ from\ n\ to\ n'$

$Let\ k\ be\ the\ shortest\ path\ from\ n'\ to\ G$

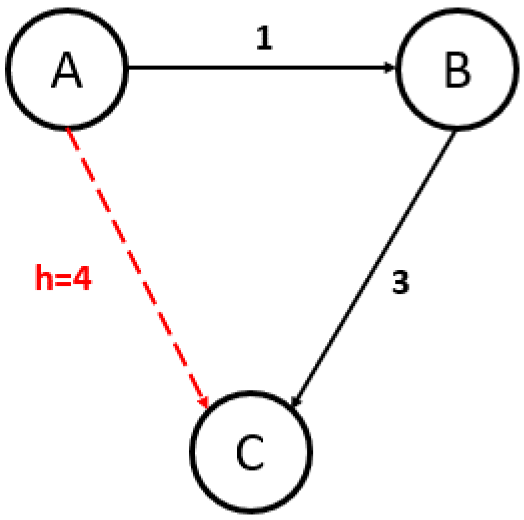
$\boldsymbol{Base\ Case}\ (\boldsymbol{k = 0})$:
$h(n) \leq c + h(n')$
$if\ h(n') = 0 \rightarrow h(n) \leq c = h^*(n)$

$\boldsymbol{Proposition}$:
$h(n')\ is\ admissible, therefore$
$h(n') \geq 0\ and\ h(n') \leq h^*(n)$

$\boldsymbol{Inductive\ step}$:
$h(n) \leq c + h(n') \leq c + h^*(n') = h^*(n)$
$\rightarrow$
$h(n) \leq h^*(n)$

$\boldsymbol{Conclusion}$:
$Every\ consistent\ heuristic\ is\ admissable$

b) Counterexample



Given this Heuristic

$h(C) = 0$
$h(B) = 1$
$h(A) = 4$

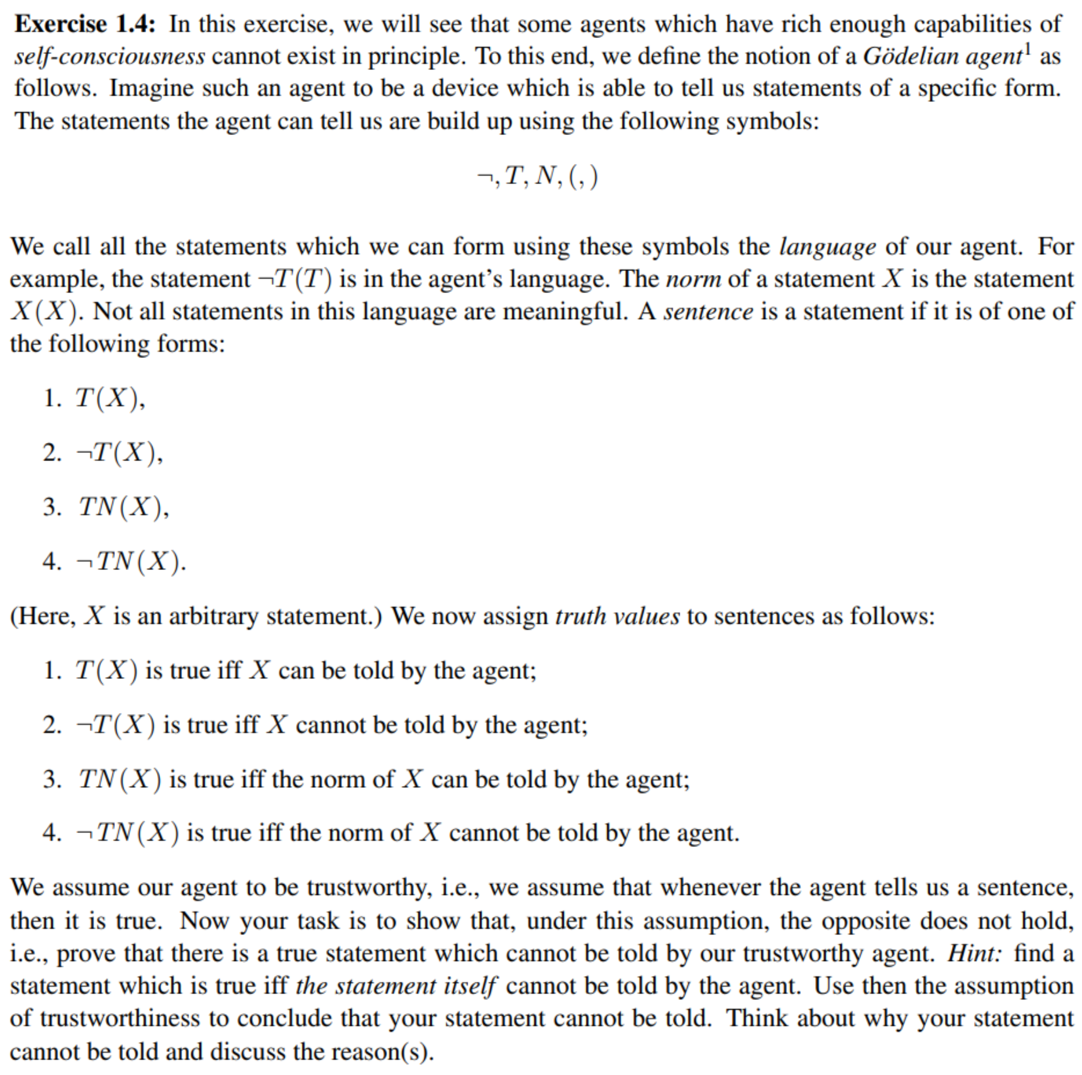
Admissibility is granted

$h(C) = 0 \leq 0$
$h(B) = 1 \ \leq 3$
$h(A) = 4 \leq 4$

Inconsistency

$h(C) = 0 \leq c(C) = 0: true \rightarrow consistent$
$h(B) = 1 \leq 3 + h(C) = 3: true \rightarrow consitent$
$h(A) = 4 \leq 1 + h(B) = 2: false \rightarrow inconsistent$

Exercise 1.4



Exercise 1.4: In this exercise, we will see that some agents which have rich enough capabilities of *self-consciousness* cannot exist in principle. To this end, we define the notion of a *Gödelian agent*[1] as follows. Imagine such an agent to be a device which is able to tell us statements of a specific form. The statements the agent can tell us are build up using the following symbols:

$$\neg, T, N, (,)$$

We call all the statements which we can form using these symbols the *language* of our agent. For example, the statement $\neg T(T)$ is in the agent's language. The *norm* of a statement X is the statement $X(X)$. Not all statements in this language are meaningful. A *sentence* is a statement if it is of one of the following forms:

1. $T(X)$,
2. $\neg T(X)$,
3. $TN(X)$,
4. $\neg TN(X)$.

(Here, X is an arbitrary statement.) We now assign *truth values* to sentences as follows:

1. $T(X)$ is true iff X can be told by the agent;
2. $\neg T(X)$ is true iff X cannot be told by the agent;
3. $TN(X)$ is true iff the norm of X can be told by the agent;
4. $\neg TN(X)$ is true iff the norm of X cannot be told by the agent.

We assume our agent to be trustworthy, i.e., we assume that whenever the agent tells us a sentence, then it is true. Now your task is to show that, under this assumption, the opposite does not hold, i.e., prove that there is a true statement which cannot be told by our trustworthy agent. *Hint:* find a statement which is true iff *the statement itself* cannot be told by the agent. Use then the assumption of trustworthiness to conclude that your statement cannot be told. Think about why your statement cannot be told and discuss the reason(s).

n … symbolizes not

$\boldsymbol{S = T(S)}$:

$S = true \rightarrow T(S) = true \rightarrow S = true \ldots consistent$

$S = false \rightarrow T(S) = false \rightarrow S = false \ldots consistent$

$\boldsymbol{S = nT(S)}$:

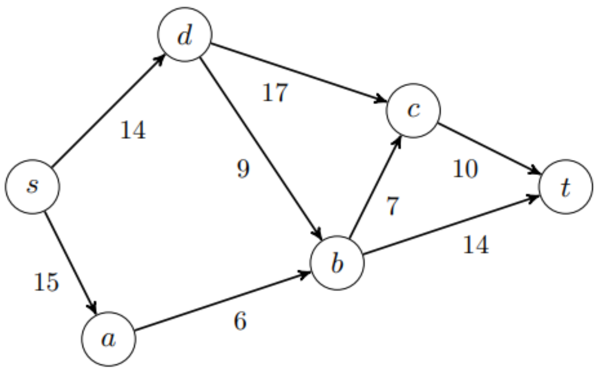
$S = true \rightarrow nT(S) = false \rightarrow S = false \ldots inconsistent$

$S = false \rightarrow nT(S) = true \rightarrow S = true \ldots inconsistent$

Since my chosen statement is self-referential, it can break the language.

Exercise 1.5

Exercise 1.5: Perform the A* algorithm using the given heuristic function h on the following graph in order to find a shortest path from s to t. In which order are the nodes expanded? Show the contents of the priority queue at each iteration. If multiple nodes have the same priority, expand the one that comes first alphabetically.



$$h(s) = 20, h(a) = 15, h(b) = 8, h(c) = 5, h(d) = 12, h(t) = 0$$

A^*: Use evaluation function $f(n) = g(n) + h(n)$
(and avoid expanding paths that are already expensive)

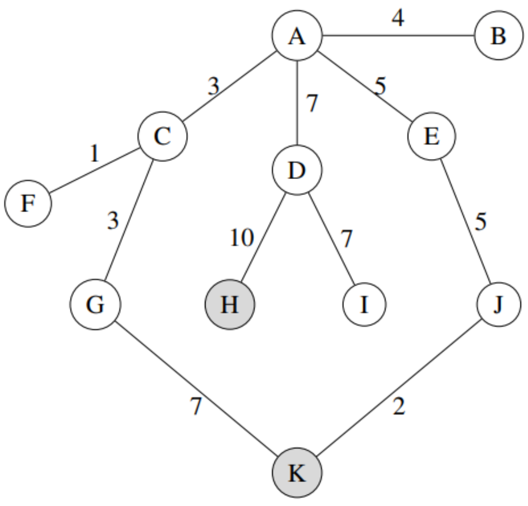
- $g(n)$: path costs from start to n, (i.e., costs so far up to n)
- $h(n)$: estimated cost to goal from n (like in greedy search)
- $f(n)$: estimated total cost of path through n to goal

							Current Values				
Step	**Node**	**g(n)**	**h(n)**	**f(n)**	**s**	**a**	**b**	**c**	**d**	**t**	**Chosen Node**
1	s	0	20	20	20	-	-	-	-	-	s
2	a	15	15	30	20	30	-	-	26	-	d
	d	14	12	26							
3	c	31	5	36	20	30	31	36	26	-	a
	b	23	8	31							
4	b	21	8	29	20	30	29	36	26	-	b
5	c	28	5	33	20	30	29	33	26	35	c
	t	35	0	35							
6	t	35	0	35	20	30	29	33	26	35	t
7					20	30	29	33	26	35	t

Visited Nodes

Exercise 1.6

Exercise 1.6: Consider the following graph (the grey nodes are goal nodes):



Use the listed search strategies on the given graph to look for a goal node, starting from the node A (depth 0). In case you can expand several nodes and the search strategy does not specify the order, choose the nodes in alphabetic order (or place them in the collection in such way that they are processed alphabetically). Where applicable, specify the contents of the *frontier* and the *explored set* for each step or the contents of the call stack for recursive approaches.

- Breadth-first search with goal test at generation time,
- Uniform cost search,
- Depth-first search with goal test at expansion time, iterative,
- Depth-limited search (use a limit of 2),
- Iterative deepening search.

BFS

Step	Node	List (FIFO)	Explored
1	A	A	
2	A	BCDE	A
3	B	CDE	AB
4	C	DEFG	ABC
5	D	EFGHI	ABCD
6	H	EFGI	ABCDH

Note: Controls if the new node in the list is a goal

DFS

Step	Node	List (FIFO)	Explored
1	A	A	
2	A	BCDE	A
3	B	CDE	AB
4	C	DEFG	ABC
5	F	DEG	ABCF
6	G	DEK	ABCFG
7	K	DEJ	ABCFGK

Note: Controls if the new node in the list is a goal

UCS

Step	Node	Note	List (Path Cost)	Explored
1	A		A0	
2	A		C3 B4 E5 D7	A
3	C		B4 F4 E5 G6 D7	AC
4	B		F4 E5 G6 D7	ACB
5	F		E5 G6 D7	ACBF
6	E		G6 D7 J10	ACBFE
7	G		D7 J10 K13	ACBFEG
8	D		J10 K13 I14 H17	ACBFEGD
9	J	K12	K12 I14 H17	ACBFEGDJ
10	K		I14 H17	ACBFEGDJK

Note: Does not controls if the new node in the list is a goal

DFS (Limit=2)

Step	Node	Limit - Depth	Stack	Explored
1	A	2	A	
2	A	2	BCDE	A
3	B	1	CDE	AB
4	C	1	DEFG	ABC
5	F	0	DEG	ABCF
6	G	0	DE	ABCFG
7	D	1	EHI	ABCFGD
8	H	0	EI	ABCFGDH

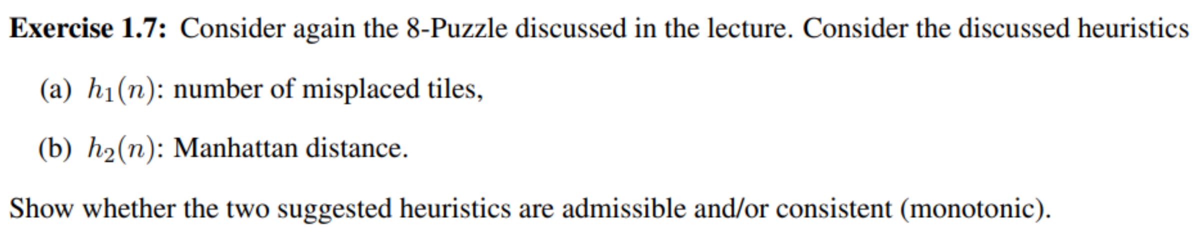
Note: Controls if the new node in the list is a goal

IDS

Step	Node	Limit - Depth	Stack	Explored
1	A	0	A	
2	A	1	A	
3	A	1	BCDE	A
4	B	0	CDE	AB
5	C	0	DE	ABC
6	D	0	E	ABCD
7	E	0		ABCDE
8	A	2	A	
9	A	2	BCDE	A
10	B	1	CDE	AB
11	C	1	DEFG	ABC
12	F	0	DEG	ABCF
13	G	0	DE	ABCFG
14	D	1	EHI	ABCFGD
15	H	0	EI	ABCFGDH

Note: Controls if the new node in the list is a goal

Exercise 1.7


Exercise 1.7: Consider again the 8-Puzzle discussed in the lecture. Consider the discussed heuristics

(a) $h_1(n)$: number of misplaced tiles,

(b) $h_2(n)$: Manhattan distance.

Show whether the two suggested heuristics are admissible and/or consistent (monotonic).

a) h1(n): Number of misplaced tiles

Is admissible:

h(g) = 0 … if the puzzle is solved, all tiles are correct
h(n) >= 0 … either the puzzle is correct, or incorrect and therefore 0 <= h(n) <= 8
h(n) <= h*(n) … every misplaced tile must be moved at least once, to get it to the correct position

Is consistent

h(n) <= c(n, a, n') + h(n') … Three scenarios can occur after a move:
1. A misplaced tile is now correct, therefore h(n) <= 1 + h(n') with h(n') = h(n)-1
2. A misplaced tile is still misplaced, therefore h(n) <= 1 + h(n') with h(n') = h(n)
3. A correct placed tile is now misplaced, therefore h(n) <= 1 + h(n') with h(n') = h(n)+1

b) h2(n): Manhattan distance

Is admissible:

h(g) = 0 … if the puzzle is solved, all tiles are correct
h(n) >= 0 … either the puzzle is correct, or incorrect and therefore 0 <= h(n)
h(n) <= h*(n) … every misplaced tile must be moved at least the Manhattan distance to its place, to get it to the correct position

Is consistent:

h(n) <= c(n, a, n') + h(n') … Four scenarios can occur after a move:
1. A misplaced tile is now correct, therefore h(n) <= 1 + h(n') with h(n') = h(n)-1
2. A misplaced tile is still misplaced but further, therefore h(n) <= 1 + h(n') with h(n') = h(n)+1
3. A misplaced tile is still misplaced but closer, therefore h(n) <= 1 + h(n') with h(n') = h(n)-1
4. A correct placed tile is now misplaced, therefore h(n) <= 1 + h(n') with h(n') = h(n)+1

Exercise 1.8

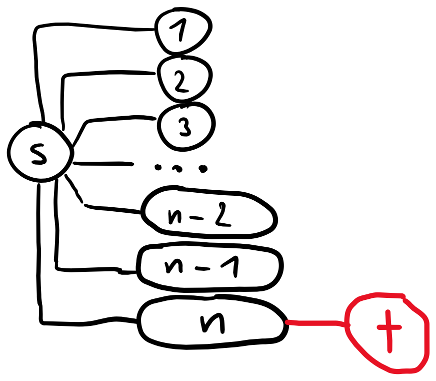
Exercise 1.8: Decide and explain which of the following statements are true and which are false? Back up your answers with proofs or counterexamples.

(a) If we consider an *arbitrary* search space, then there exists a graph on which neither breadth-first search nor depth-first search would be complete. Provide a proof or argue what would be the smallest counterexample in this case.

Answer: True

BFS and DFS are only complete if the search space is finite.

Therefore, this infinite graph cannot be solved:



Since the number of steps to find s → t via BFS or DFS are n+1, and as n tends to infinity, so do the number of steps.

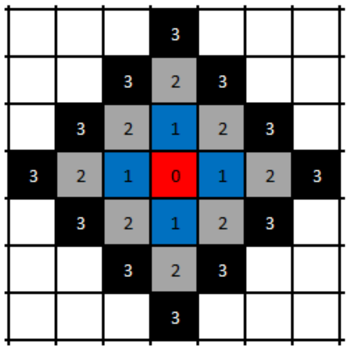
(b) In chess, a rook can move on any number of squares on the board in a straight line, vertically or horizontally, but cannot jump over other figures. Then, the Chebyshev distance is an admissible heuristic for the problem of moving the rook from square A to square B via the shortest path, where the costs are the number of travelled fields.

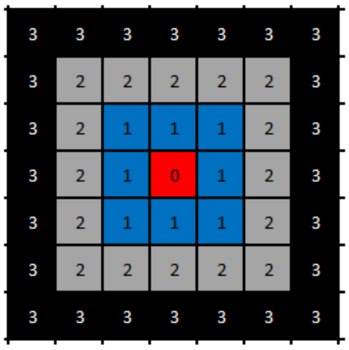
The Chebyshev distance between two points $(x_1, y_1), (x_2, y_2)$ on a plane is given by:

$$\max\{|x_1 - x_2|, |y_1 - y_2|\}$$

Answer: True

h(n) <= h*(n) … Correct, since the Chebyshev Distance is always smaller than the Manhattan distance





h(n) >= 0 … Correct, since the bigger number of two positive numbers (absolute value) is positive
h(g) = 0 … Correct, since A == B and therefore the rook does not have to move

(c) The A* algorithm yields an optimal path in a graph search if the used heuristic is admissible.

Answer: True/False

There are 2 variants of the A* algorithm, therefore there are also 2 answers:

1) A* tree search algorithm delivers an optimal solution when used with an admissible heuristic.
   Tree search: We do not keep a closed list → the same node can be visited multiple times
2) A* graph search algorithm delivers an optimal solution when used with a consistent heuristic (stronger condt.)
   Graph search: We keep a closed list of visited nodes → the same node won't be visited multiple times

Proof by contradiction of A* with tree search:

1. Suppose A* with an admissible heuristic wasn't optimal.

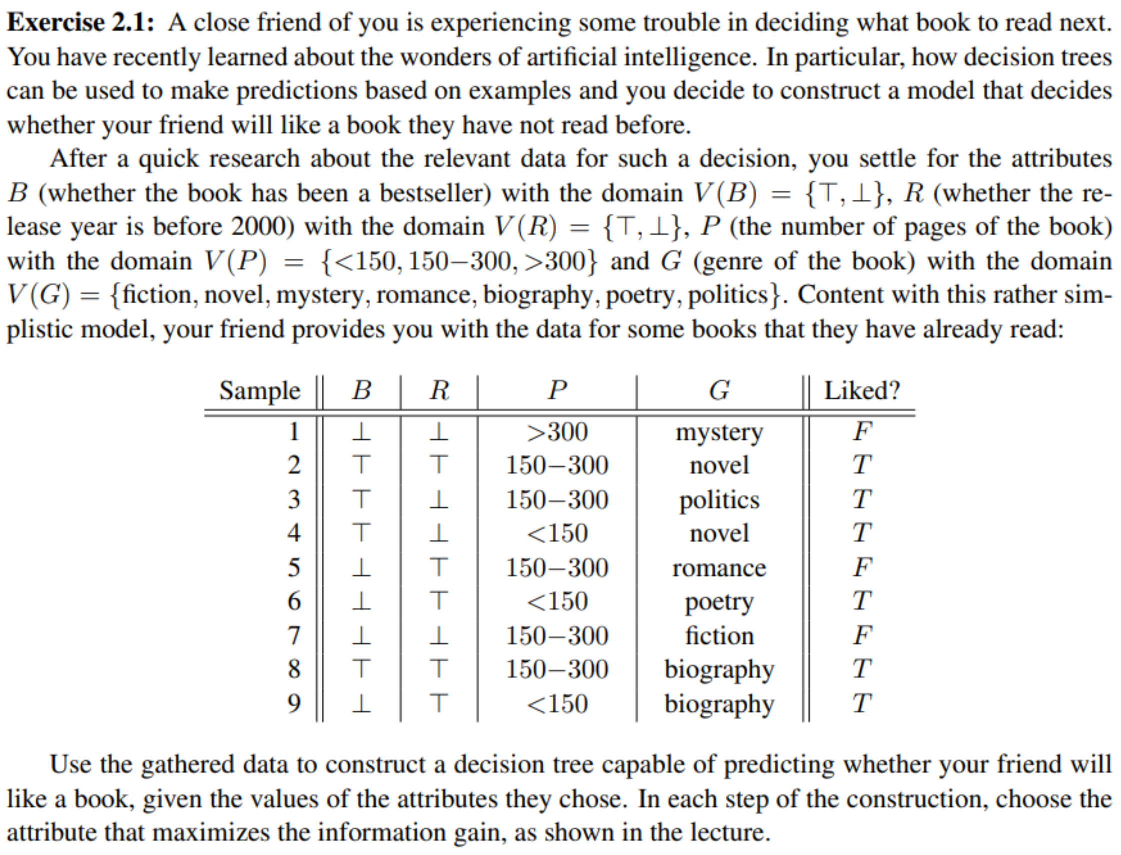
2. This would mean A* finds a longer path to the goal first, while there exists a shorter one.

3. Given the heuristic is admissible, the estimated cost of the shorter path must be less than its true cost.

4. A* always picks the path with the lowest estimated total cost next, contradicting the assumption that it found a longer path first.

5. Hence, A* must be optimal when using an admissible heuristic

Exercise 2.1



Exercise 2.1: A close friend of you is experiencing some trouble in deciding what book to read next. You have recently learned about the wonders of artificial intelligence. In particular, how decision trees can be used to make predictions based on examples and you decide to construct a model that decides whether your friend will like a book they have not read before.

After a quick research about the relevant data for such a decision, you settle for the attributes B (whether the book has been a bestseller) with the domain $V(B) = \{\top, \bot\}$, R (whether the release year is before 2000) with the domain $V(R) = \{\top, \bot\}$, P (the number of pages of the book) with the domain $V(P) = \{<150, 150–300, >300\}$ and G (genre of the book) with the domain $V(G) = \{\text{fiction, novel, mystery, romance, biography, poetry, politics}\}$. Content with this rather simplistic model, your friend provides you with the data for some books that they have already read:

Sample	B	R	P	G	Liked?
1	$\bot$	$\bot$	>300	mystery	F
2	$\top$	$\top$	150–300	novel	T
3	$\top$	$\bot$	150–300	politics	T
4	$\top$	$\bot$	<150	novel	T
5	$\bot$	$\top$	150–300	romance	F
6	$\bot$	$\top$	<150	poetry	T
7	$\bot$	$\bot$	150–300	fiction	F
8	$\top$	$\top$	150–300	biography	T
9	$\bot$	$\top$	<150	biography	T

Use the gathered data to construct a decision tree capable of predicting whether your friend will like a book, given the values of the attributes they chose. In each step of the construction, choose the attribute that maximizes the information gain, as shown in the lecture.

Useful equations

$$B(q) = -(q * log_2 q + (1-q)\log_2(1-q)) \ldots \mathrm{q = proportion}$$

$$Rem(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p+n} * B\left(\frac{p_k}{p_k + n_k}\right) \ldots p = pos.; n \ldots neg; x_k \ldots k = state((T,F)(m,n,p,f))$$

$$Gain(A) = B\left(\frac{p}{p+n}\right) - Rem(A) \ldots p = pos.; n \ldots neg.$$

Selection of the first attribute

$$Gain(B) = B\left(\frac{6}{6+3}\right) - Rem(B) = 0.918 - 0.539 = 0.379$$

$$Gain(R) = B\left(\frac{6}{6+3}\right) - Rem(R) = 0.918 - 0.845 = 0.073$$

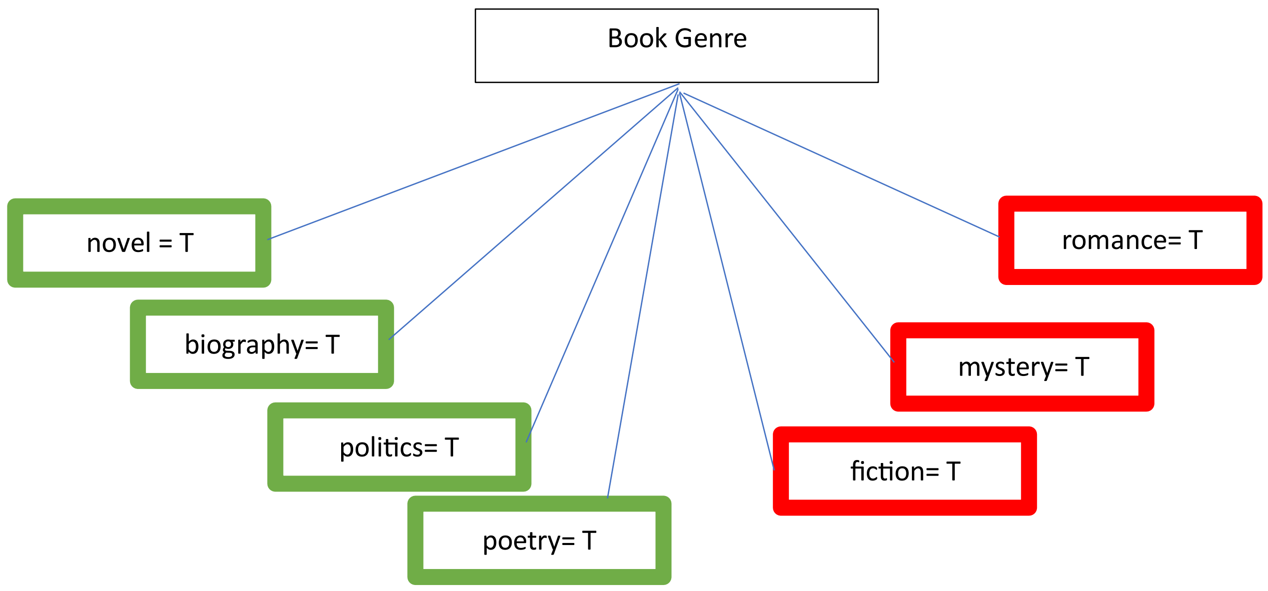
$$Gain(P) = B\left(\frac{6}{6+3}\right) - Rem(P) = 0.918 - 0.539 = 0.379$$

$$Gain(G) = B\left(\frac{6}{6+3}\right) - Rem(G) = 0.918 - 0.000 = 0.918$$

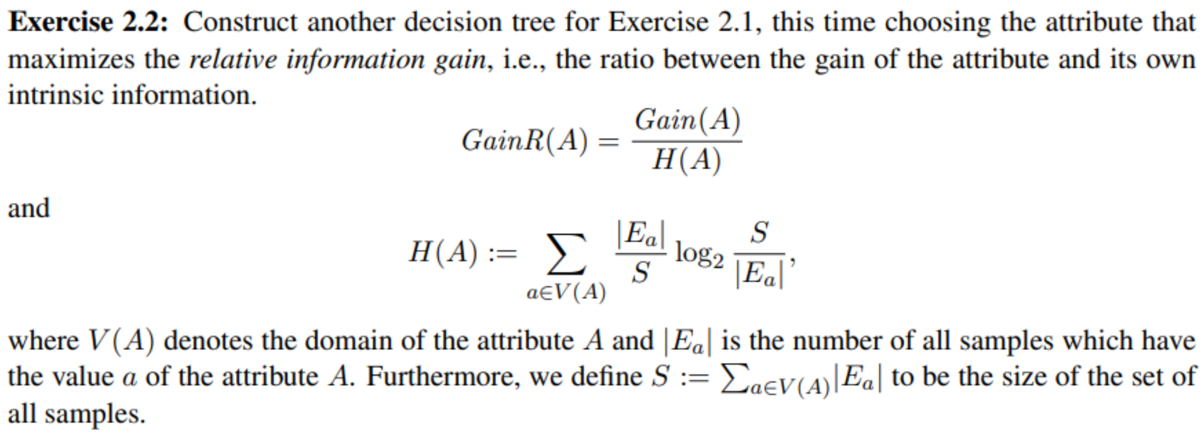
By choosing attribute “G”, we gain the most information (Gain = 0.918)

Conclusion

Further attribute selection is not necessary, since the attribute “G” creates a perfect mapping



Exercise 2.2



Exercise 2.2: Construct another decision tree for Exercise 2.1, this time choosing the attribute that maximizes the *relative information gain*, i.e., the ratio between the gain of the attribute and its own intrinsic information.

$$GainR(A) = \frac{Gain(A)}{H(A)}$$

and

$$H(A) := \sum_{a \in V(A)} \frac{|E_a|}{S} \log_2 \frac{S}{|E_a|},$$

where $V(A)$ denotes the domain of the attribute A and $|E_a|$ is the number of all samples which have the value a of the attribute A. Furthermore, we define $S := \sum_{a \in V(A)} |E_a|$ to be the size of the set of all samples.

Selection of the first attribute (V = 9)

$$GainR(B) = \frac{Gain(B)}{H(B)} = \frac{0.379}{0.520 + 0.471} = 0.382$$

$$GainR(R) = \frac{Gain(R)}{H(R)} = \frac{0.073}{0.471 + 0.520} = 0.074$$

$$GainR(P) = \frac{Gain(P)}{H(P)} = \frac{0.379}{0.528 + 0.471 + 0.352} = 0.280$$

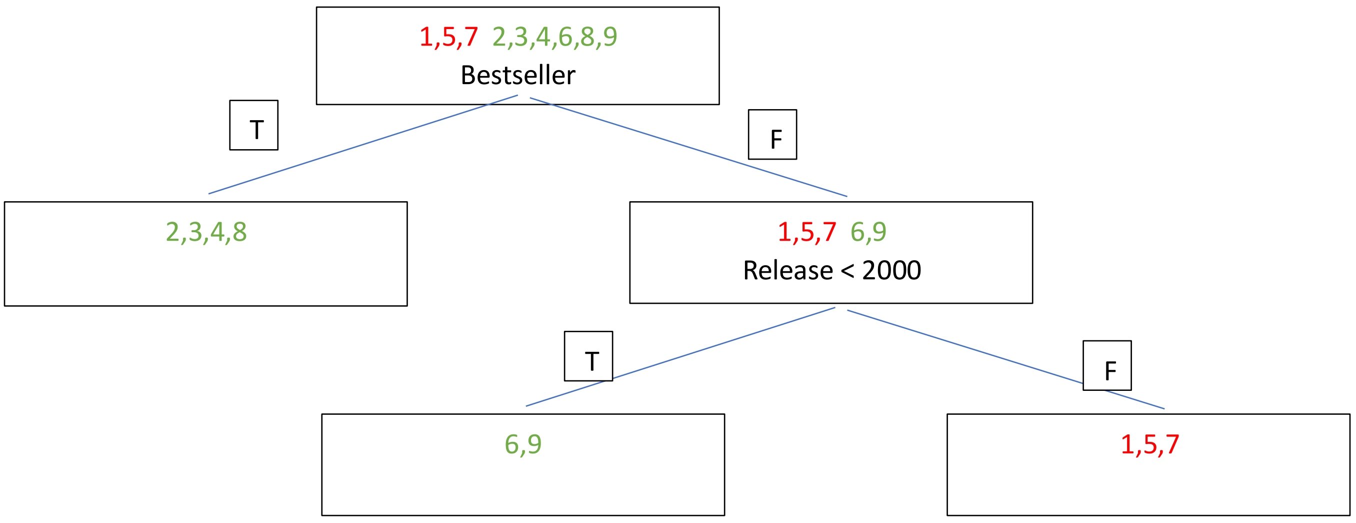
$$GainR(G) = B\frac{Gain(B)}{H(B)} = \frac{0.918}{0.482 + 0.352 + 0.352 + 0.482 + 0.352 + 0.352 + 0.352} = 0.139$$

By choosing attribute "B", we gain the most relative information (GainR = 0.382)

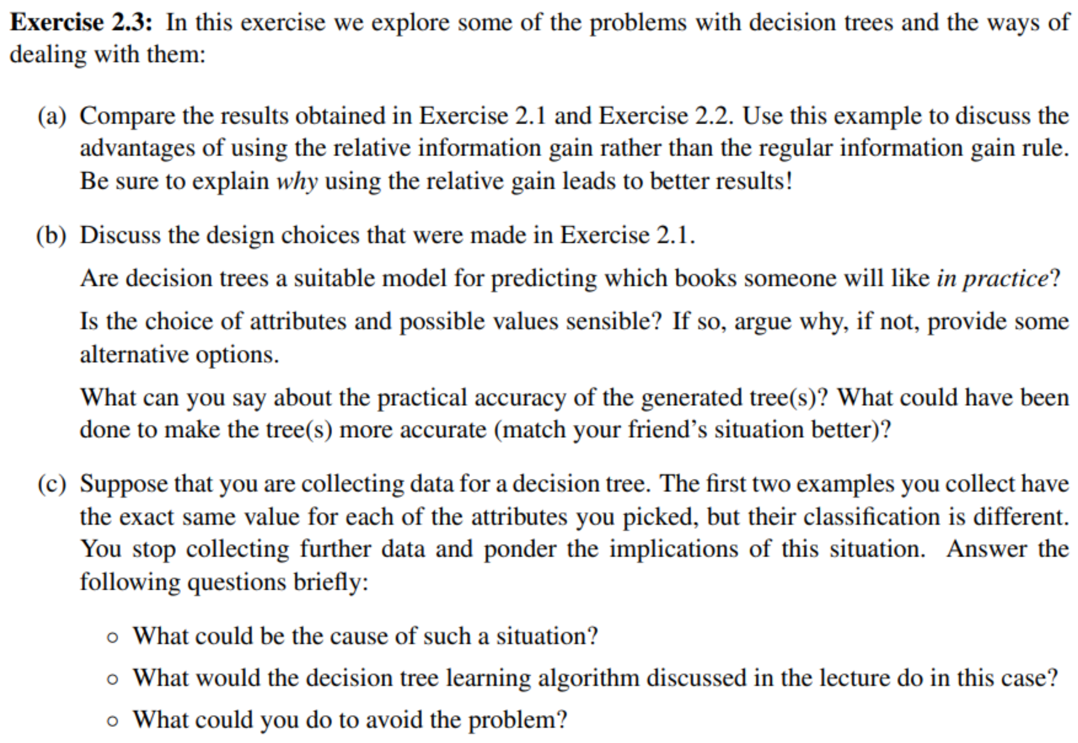
Selection of the first attribute (V = 2)

Sample	B	R	P	G	Liked?
6	⊥	T	<150	poetry	T
9	⊥	T	<150	biography	T

Given this table it is clear, everything could be used to get the correct "Liked?" value, since both times, the "Liked?" value is true.
Therefore I choose R as the next distinguisher:



Exercise 2.3



Exercise 2.3: In this exercise we explore some of the problems with decision trees and the ways of dealing with them:

(a) Compare the results obtained in Exercise 2.1 and Exercise 2.2. Use this example to discuss the advantages of using the relative information gain rather than the regular information gain rule. Be sure to explain *why* using the relative gain leads to better results!

(b) Discuss the design choices that were made in Exercise 2.1.

Are decision trees a suitable model for predicting which books someone will like *in practice*?

Is the choice of attributes and possible values sensible? If so, argue why, if not, provide some alternative options.

What can you say about the practical accuracy of the generated tree(s)? What could have been done to make the tree(s) more accurate (match your friend's situation better)?

(c) Suppose that you are collecting data for a decision tree. The first two examples you collect have the exact same value for each of the attributes you picked, but their classification is different. You stop collecting further data and ponder the implications of this situation. Answer the following questions briefly:

- What could be the cause of such a situation?
- What would the decision tree learning algorithm discussed in the lecture do in this case?
- What could you do to avoid the problem?

a)

Even though the regular information gain is a good measure to search for the most relevant attribute, it can lead to some problems. If the chosen attribute has a lot of distinct values, like the value in our example, which can take 7 different values, it is likely to result in a bigger information gain. If, for example, the value would be evaluated on a scale from 1 to 100, it quite certainly would lead to a huge information gain, especially if the gathered data is such a small sample, but this approach does not lead to a better decision tree. This issue can be mitigated by using the relative information gain, because the number of distinct values for each attribute is taken into account.

b)

Yes, but only if the parameters are chosen "correctly". For example, the release date parameter (< 2000) is horrible for books. This might produce a valid tree but does not help at all. If we have a huge sample size and "good" it might be feasible. Better options would have been: writing style or theme.

To get better trees, we would need a bigger sample size and better parameters and update the tree every 10 books.

c)

Maybe I am missing a vital parameter.

The DTL algorithm would still build a decision tree, but the final node could still not be classified.

"Clean" the dataset, by either deleting both examples, or only pick more sensible one and delete the other.
Add another parameter and try again.

Exercise 2.4

Exercise 2.4: Suppose that an attribute splits the set of examples E into k subsets $E_1, \ldots, E_k$ where each subset E_i has p_i positive and n_i negative examples. Show that the attribute has zero information gain if the ratio $\frac{p_i}{p_i+n_i}$ is equal for all $i \in \{1, \ldots, k\}$. Provide arguments for all properties that you use.

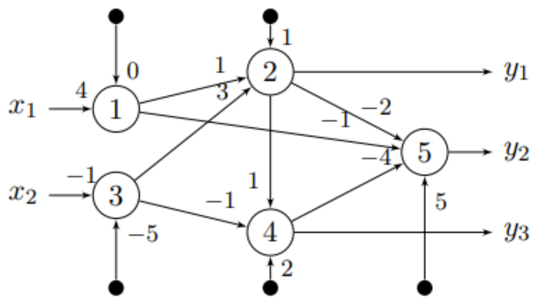
$$
\begin{aligned}
Rem(A) &= \sum_k \frac{p_k + n_k}{p+n} * B\left(\frac{p_k}{p_k+n_k}\right) \\
&= B\left(\frac{p}{p+n}\right) * \sum_k \frac{p_k+n_k}{p+n} \\
&= B\left(\frac{p}{p+n}\right) * 1
\end{aligned}
$$

Since $\frac{\mathrm{p_i}}{\mathrm{p_i+n_i}}$ is equal for all i, we can pull the B() out of the sum

$$
\begin{aligned}
Gain(A) = {} & B\left(\frac{p}{p+n}\right) - Rem(A) \\
&= B\left(\frac{p}{p+n}\right) - B\left(\frac{p}{p+n}\right) \\
&= 0
\end{aligned}
$$

Exercise 2.5

Exercise 2.5: Consider a neural network with five nodes of the following form:



The numbers next to the arrows denote the respective weights. Node 4 has $g(x) = -x$ as its activation function. Node 2 uses the identity function $g(x) = x$. Nodes 1 and 3 use a hard limiter with a threshold of 0.5, while node 5 uses the sigmoid function $g(x) = 1/(1 + e^{-x})$.

What is the output produced by the network when the input is $(x_1, x_2) = (1, 1)$? Determine the input and output of every neuron.

$x1 = 1$
$x2 = 1$

$$g1(x) = \begin{cases} 1, & x \geq 0.5 \\ 0, & x < 0.5 \end{cases}$$
$g2(x) = x$
$g3(x) = g1(x)$
$g4(x) = -x$
$$g5(x) = \frac{1}{1 + e^{-x}}$$

$$g\left(\sum input_i * weight_i\right)$$
$n1: g1(1 * 4 + 1 * 0) = g1(4) = 1$
$n3: g3(1 * -1 - 5) = g3(-6) = 0$
$n2: g2(1 * 1 + 0 * 3 + 1 * 1) = g2(2) = 2$
$n4: g4(2 * 1 + 0 * -1 + 1 * 2) = g4(4) = -4$
$n5: g5(1 * -1 + 2 * -2 + (-4) * -4 + 1 * 5) = g5(16) = 0.999{\sim}1$

$y1 = n2 = 2$
$y2 = n5 = 1$
$y3 = n4 = -4$

Exercise 2.6

Exercise 2.6: Design a neural network with six binary input signals $S_0, S_1, I_0, I_1, I_2, I_3$ and a single binary output signal I_{out}, which behaves as a 4-bit multiplexer. You are free to determine the structure of the network yourself, but try to use as few layers as possible. The following activation function should be used for *all* neurons:

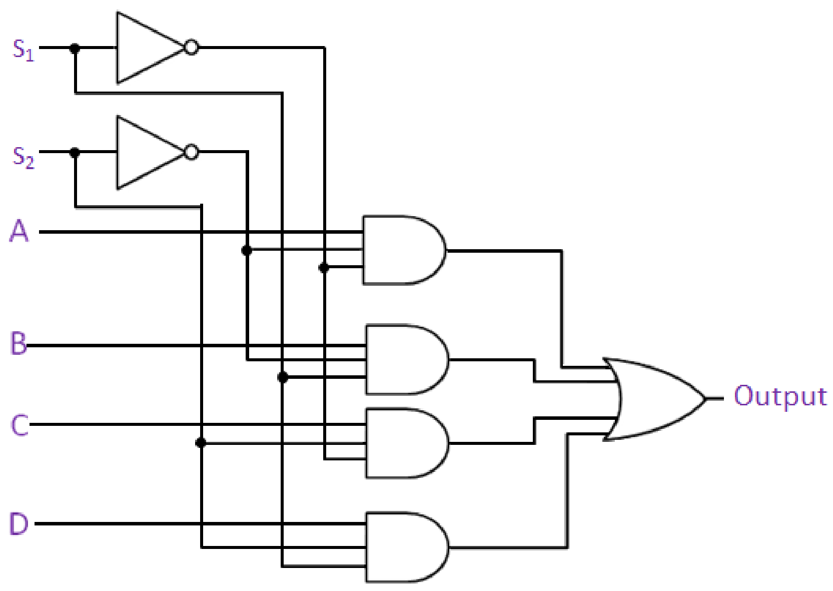
$$g(x) = \begin{cases} 1 & \text{if } 0.2 \cdot x \geq 1\,, \\ 0 & \text{otherwise.} \end{cases}$$

As a reminder, a 4-bit multiplexer uses its inputs (S_0 and S_1) to select and propagate one of the four input values (I_0, I_1, I_2 and I_3). The remaining input values are ignored. The following truth table illustrates this behavior:

S_1	S_0	I_0	I_1	I_2	I_3	I_{out}
0	0	1	$*$	$*$	$*$	1
0	0	0	$*$	$*$	$*$	0
0	1	$*$	1	$*$	$*$	1
0	1	$*$	0	$*$	$*$	0
1	0	$*$	$*$	1	$*$	1
1	0	$*$	$*$	0	$*$	0
1	1	$*$	$*$	$*$	1	1
1	1	$*$	$*$	$*$	0	0

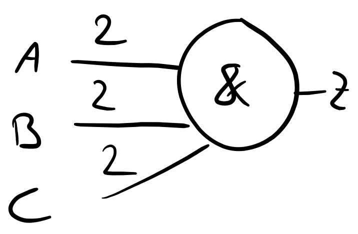
Note: *The $*$ symbol means that the signal at the given position can be either 1 or 0, as its value does not affect the output in any way.*

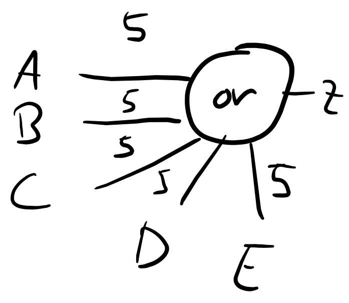
A 4-bit multiplexer can be built with 2 Not Gates, 4 AND Gates and one OR Gate:

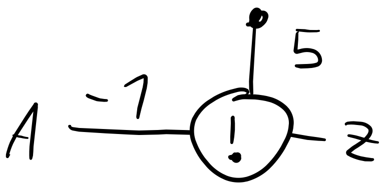


These Gates can be built from neurons and the given g(x):

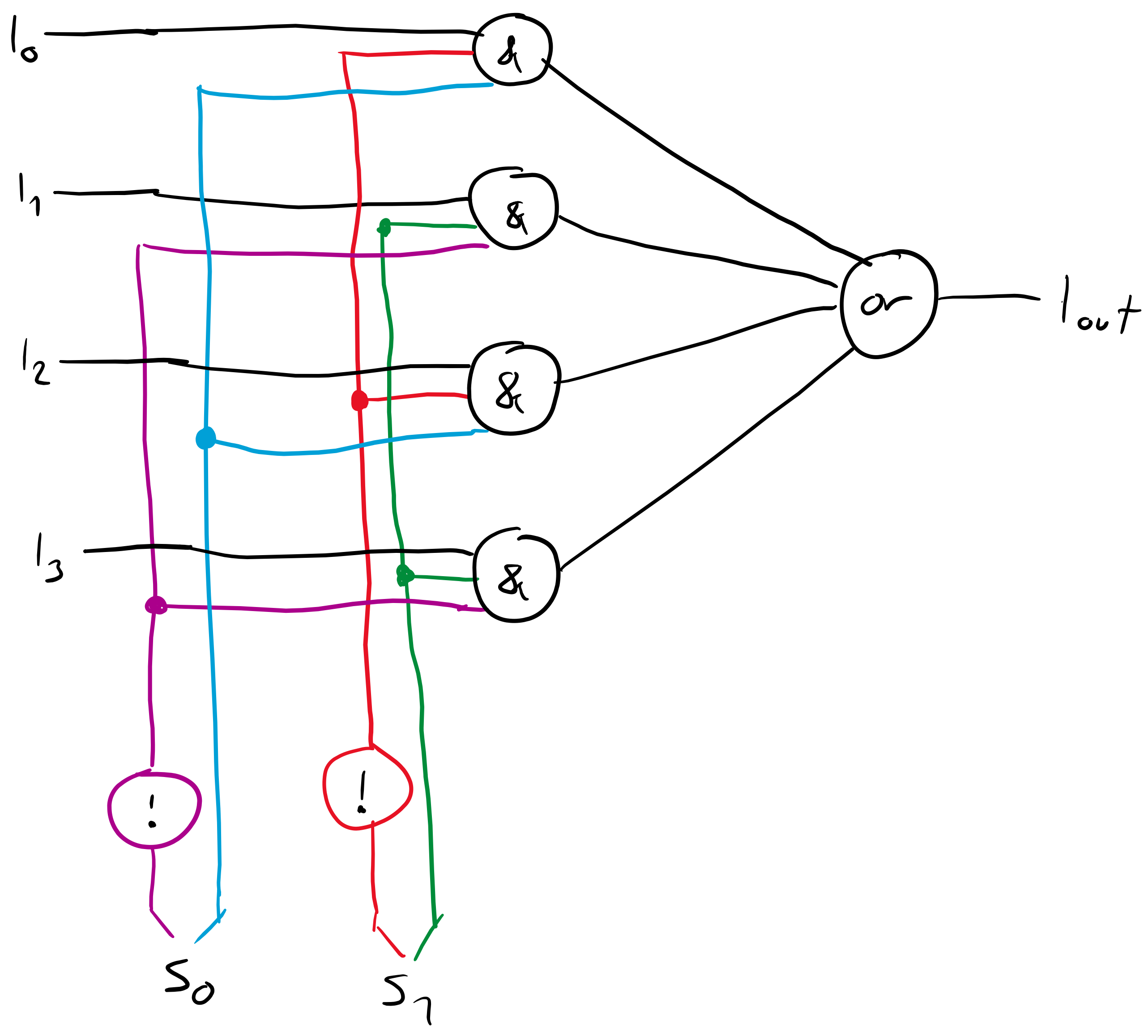
AND Gate: OR Gate Not Gate





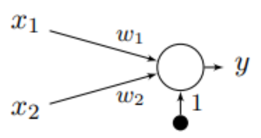


Therefore the multiplexer can be built like this:



Exercise 2.7

Exercise 2.7: Consider a single layer perceptron with two input neurons and one output neuron of the following form:



Train the perceptron using the *Perceptron learning rule* and the identity activation function $g(x) = x$ on the following training data: $f(3,1) = 6$, $f(0,2) = 12$, $f(4,-7) = 12$ and $f(-8,12) = -35$. The weights are initialized to 0 and the learning rate α is 1. The bias weight will not be learned and stays 1.

Express the function $f(x_1, x_2)$ that your network has learned after the training is complete algebraically. Does this function produce the expected outputs for all four training inputs?

If the neuron has not learned the function correctly, provide alternative weights so that the neuron produces the expected outputs for all examples, or prove that this is impossible.

Useful equations

$$in = \sum_i w_i * x_i$$

$$h_w = g(in)$$

$$Err = y - h_w \ldots y = f(x)$$

$$w_i = w_i + \alpha * Err * g'(in) * x_i$$

Initial situation

$w_1 = 0$
$w_2 = 0$
$\alpha = 1$

1. Round: f(3, 1) = 6

$in = 0 * 3 + 0 * 1 + 1 = 1$
$h_w(3,1) = g(1) = 1$
$Err = 6 - 1 = 5$
$g'(1) = 1$
$w_1 = 0 + 1 * 5 * 1 * 3 = 15$
$w_2 = 0 + 1 * 5 * 1 * 1 = 5$

2. Round: f(0, 2) = 12

$in = 15 * 0 + 5 * 2 + 1 = 11$
$h_w(0,2) = g(11) = 11$
$Err = 12 - 11 = 1$
$g'(11) = 1$
$w_1 = 15 + 1 * 1 * 1 * 0 = 15$
$w_2 = 5 + 1 * 1 * 1 * 2 = 7$

3. Round: f(4, -7) = 12

$in = 15 * 4 + 7 * (-7) + 1 = 12$
$h_w(4,-7) = g(12) = 12$
$Err = 12 - 12 = 0$
$g'(12) = 1$
$w_1 = 15 + 1 * 0 * 1 * 4 = 15$
$w_2 = 7 + 1 * 0 * 1 * (-7) = 7$

4. Round: f(-8, 12) = -35

$in = 15 * (-8) + 7 * 12 + 1 = -35$
$h_w(-8,12) = g(-35) = 35$
$Err = -35 - (-35) = 0$
$g'(-35) = 1$
$w_1 = 15 + 1 * 1 * 0 * -8 = 15$
$w_2 = 7 + 1 * 1 * 0 * 12 = 7$

$f(x_1, x_2) = g(w_0 + w_1 * x_1 + w_2 * x_2) = g(1 + 15x_1 + 7x_2) = 1 + 15x_1 + 7x_2$
$f(3,1) = 1 + 15 * 3 + 7 * 1 = 37$
$f(0, 2) = 1 + 15 * 0 + 7 * 2 = 15$
$f(4, -7) = 1 + 15 * 4 + 7 * (-7) = 12$
$f(-8, 12) = 1 + 15 * (-8) + 7 * 12 = -35$

The learned function $f(x_1, x_2)$ does not produce the correct values

By converting the equations into a matrix, we can prove via the gauss elimination method, if a correct solution exists

$$\begin{pmatrix} 3 & 1 & 6 \\ 0 & 2 & 12 \\ 4 & -7 & 12 \\ -8 & 12 & -35 \end{pmatrix}$$

$$III = III - \frac{4}{3} * I$$

$$\begin{pmatrix} 3 & 1 & 6 \\ 0 & 2 & 12 \\ 0 & -\frac{25}{3} & 4 \\ -8 & 12 & -35 \end{pmatrix}$$

$$IV = IV + \frac{8}{3} * I$$

$$\begin{pmatrix} 3 & 1 & 6 \\ 0 & 2 & 12 \\ 0 & -\frac{25}{3} & 4 \\ 0 & \frac{44}{3} & -19 \end{pmatrix}$$

$$III = III + \frac{25}{6} * II$$

$$\begin{pmatrix} 3 & 1 & 6 \\ 0 & 2 & 12 \\ 0 & 0 & 54 \\ 0 & \frac{44}{3} & -19 \end{pmatrix}$$

This would mean 0*x1 + 0*x2 = 54, which cannot be true, therefore no solution exists

Exercise 2.8

Exercise 2.8: Show formally that a single-layer perceptron with the step function as activation function ($g(x) = 1$ for $x \geq t$ for some threshold t and $g(x) = 0$ otherwise) cannot express *XOR*.

A perceptron evaluates to the following function
$f(x_1, x_2) = g(w_0 + x_1 * w_1 + x_2 * w_2)$

XOR should therefore validate the following system of equations
$I{:}\, f(0,0) = 0$
$II{:}\, f(0,1) = 1$
$III{:}\, f(1,0) = 1$
$IV{:}\, f(1,1) = 0$

$I{:}\, f(0,0) = g(w_0)$
$II{:}\, f(0,1) = g(w_0 + w_2)$
$III{:}\, f(1,0) = g(w_0 + w_1)$
$IV{:}\, f(1,1) = g(w_0 + w_1 + w_2)$

The result of the step-function is “1”, if $g(x) \geq t$ and “0” if $g(x) < t$
Mapping these constrains to our system of equations leads to the following
$I{:}\, w_0 < t$
$II{:}\, w_0 + w_2 \geq t$
$III{:}\, w_0 + w_1 \geq t$
$VI{:}\, w_0 + w_1 + w_2 < t$

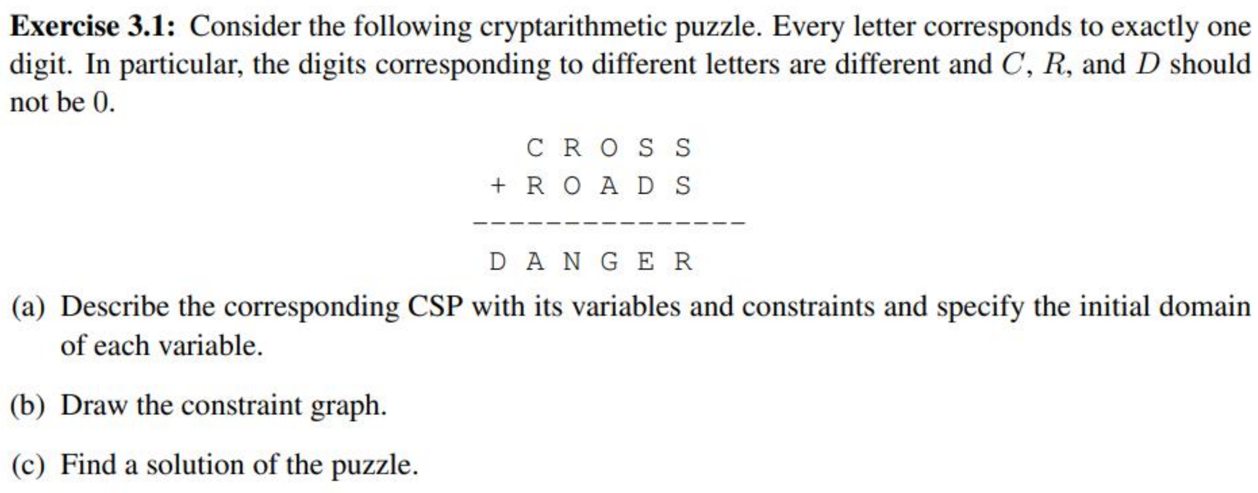
Therefore the following transformations must lead to a valid result

$I, III{:}$
$w_0 < t \leq w_0 + w_1$
$w_0 < w_0 + w_1 \;\; | - w_0$
$w_1 > 0$

$II, IV{:}$
$w_0 + w_1 + w_2 < t \leq w_0 + w_2$
$w_0 + w_1 + w_2 < w_0 + w_2 \;\; | - (w_0 + w_2)$
$w_1 < 0$

But they don’t.

Exercise 3.1



Exercise 3.1: Consider the following cryptarithmetic puzzle. Every letter corresponds to exactly one digit. In particular, the digits corresponding to different letters are different and C, R, and D should not be 0.

```
    C R O S S
  + R O A D S
 ---------------
  D A N G E R
```

(a) Describe the corresponding CSP with its variables and constraints and specify the initial domain of each variable.

(b) Draw the constraint graph.

(c) Find a solution of the puzzle.

a)

Variables:
C, R, O, S, A, D, N, G, E, X1, X2, X3, X4, X5

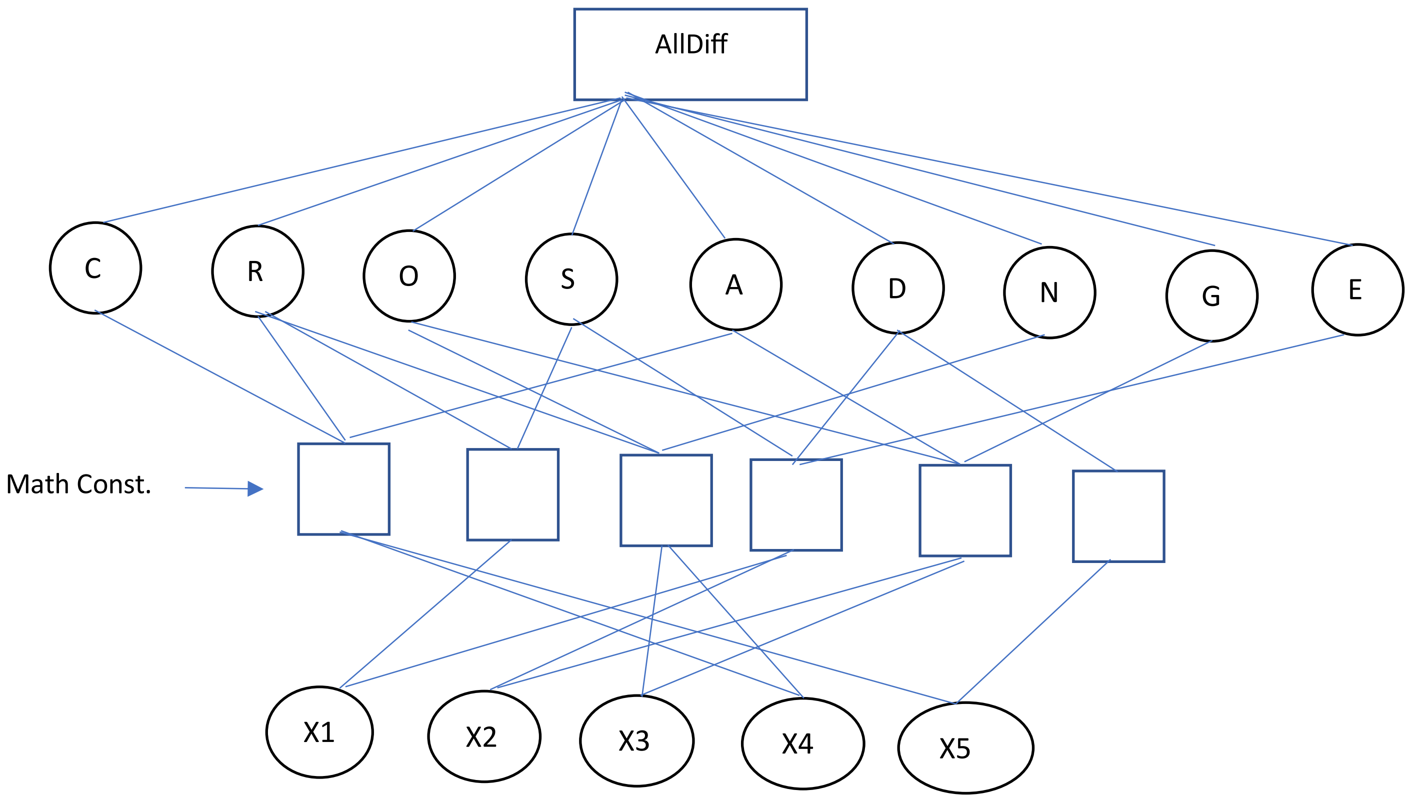
Domain:
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Constraints:
Alldiff(C, R, O, S, A, D, N, G, E)

C != 0
R != 0
D != 0
… because they are the first digit of each number

S + S = R + 10*X1
X1 + S + D = E + 10*X2
X2 + O + A = G + 10*X3
X3 + R + O = N + 10*X4
X4 + C + R = A + 10*X5
X5 = D
… to ensure the addition is correct

b)



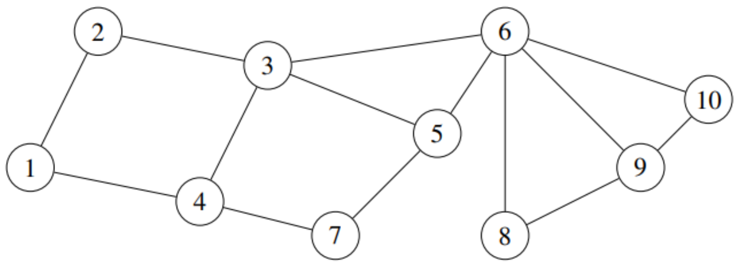
c)

Solution found via following online decoder: https://www.dcode.fr/cryptarithm-solver

CROSS+ROADS=DANGER
96233+62513=158746

Exercise 3.2

Exercise 3.2: Consider the 3-colorability problem for the following graph:



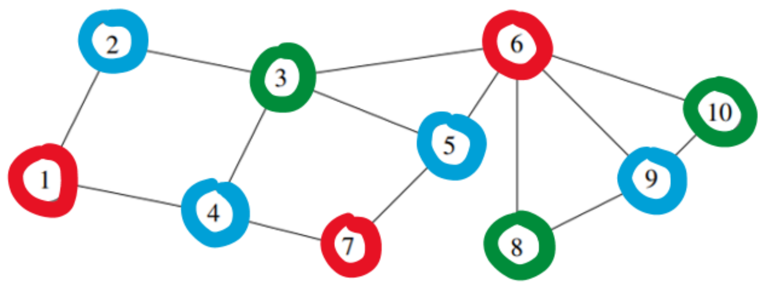
(a) Perform one step of the backtracking search on the following graph. Use the colors red, green, and blue. Select the first node by using the *degree heuristic* and after that, select the nodes according to the *minimum-remaining-values heuristic*. If the remaining values of two or more nodes are equal, apply the *degree heuristic* to these nodes. If you still have multiple options after applying both heuristics, select the node with the smallest number. If the step terminates and colors the entire graph, show the coloring it found. If it necessitates a backtrack, state which node could not be colored and stop (i.e., do not backtrack).

Heuristics:

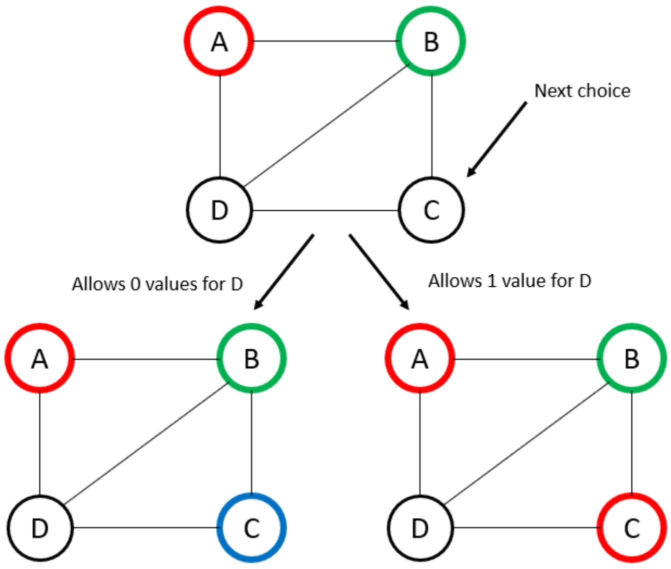
<u>Degree Heuristic (DH)</u>
Selects the node that is involved in the largest number of constraints on other unassigned nodes

<u>Minimum Remaining Values Heuristic (MRVH)</u>
Choose the variable with the fewest legal values

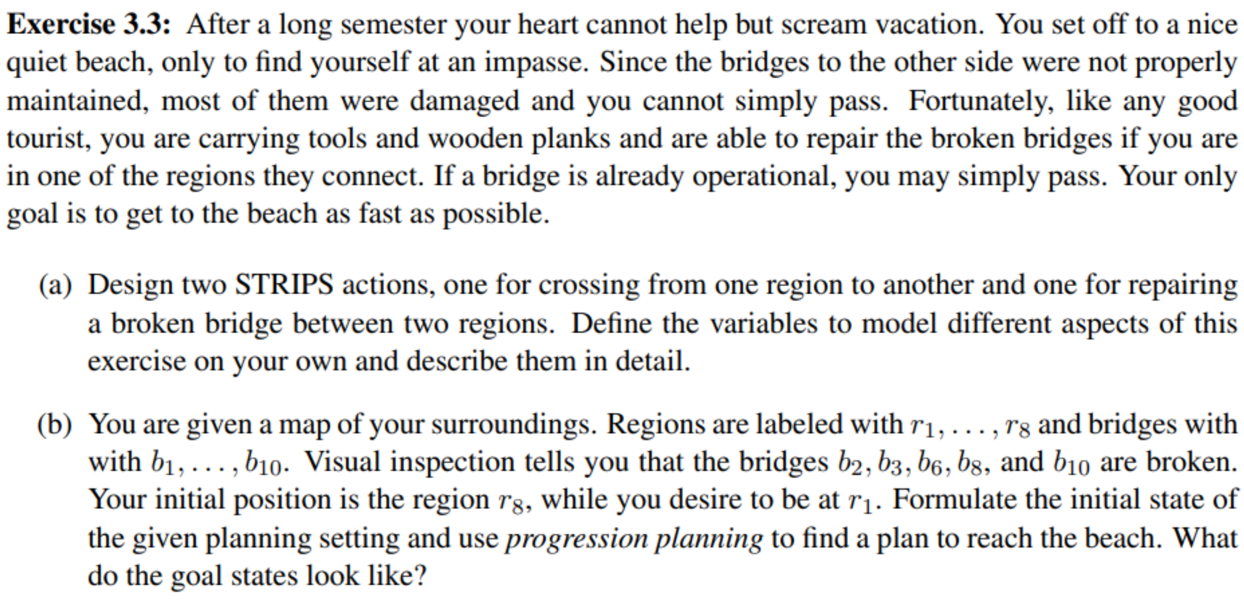
Step	Heuristic	Candidates	Visited Nodes
1	DH	6	6
2	MRVH	3,5,8,9,10	
	DH	3	3,6
3	MRVH	5	3,6,5
4	MRVH	2,4,7,8,9,10	
	DH	1,4,9	
	MIN	1	3,6,5,1
5	MRVH	2	3,6,5,1,2
6	MRVH	4	3,6,5,1,2,4
7	MRVH	7	3,6,5,1,2,4,7
8	MRVH	8,9,10	
	DH	9	3,6,5,1,2,4,7,9
9	MRVH	8	3,6,5,1,2,4,7,9,8
10	MRVH	10	3,6,5,1,2,4,7,9,8,10



(b) Give a partially colored graph and select one additional node to color so that the *least-constraining-value heuristic* forces you to choose one particular color for that node. Try to keep your example as simple as possible.

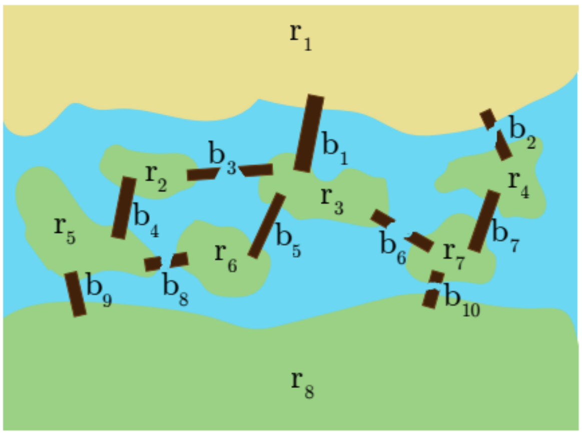


Exercise 3.3


Exercise 3.3: After a long semester your heart cannot help but scream vacation. You set off to a nice quiet beach, only to find yourself at an impasse. Since the bridges to the other side were not properly maintained, most of them were damaged and you cannot simply pass. Fortunately, like any good tourist, you are carrying tools and wooden planks and are able to repair the broken bridges if you are in one of the regions they connect. If a bridge is already operational, you may simply pass. Your only goal is to get to the beach as fast as possible.

(a) Design two STRIPS actions, one for crossing from one region to another and one for repairing a broken bridge between two regions. Define the variables to model different aspects of this exercise on your own and describe them in detail.

(b) You are given a map of your surroundings. Regions are labeled with $r_1, \ldots, r_8$ and bridges with with $b_1, \ldots, b_{10}$. Visual inspection tells you that the bridges b_2, b_3, b_6, b_8, and b_{10} are broken. Your initial position is the region r_8, while you desire to be at r_1. Formulate the initial state of the given planning setting and use *progression planning* to find a plan to reach the beach. What do the goal states look like?



a)

$\boldsymbol{Action}(Cross(r1, r2, b))$

$\quad \boldsymbol{Precond}: at(r1) \wedge connected(r1, b) \wedge connected(r2, b) \wedge (r1 \mathrel{!}= r2) \wedge !\, broken(b)$

$\quad \boldsymbol{Effect}: at(r2) \wedge !\, at(r1)$

$\boldsymbol{Action}(Repair(r, b))$

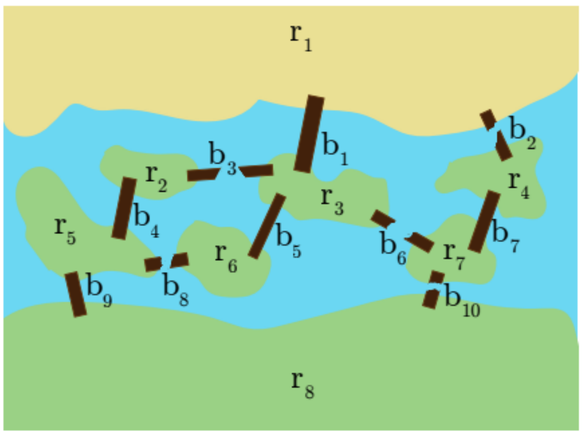
$\quad \boldsymbol{Precond}: at(r) \wedge connected(r, b) \wedge broken(b)$

$\quad \boldsymbol{Effect}: !\, broken(b)$

$at(r) \ldots Player\ is\ in\ room$ r
$connected(r, b) \ldots Room$ is connected to bridge "b"
$broken(b) \ldots Bridge$ "b" $is\ broken$

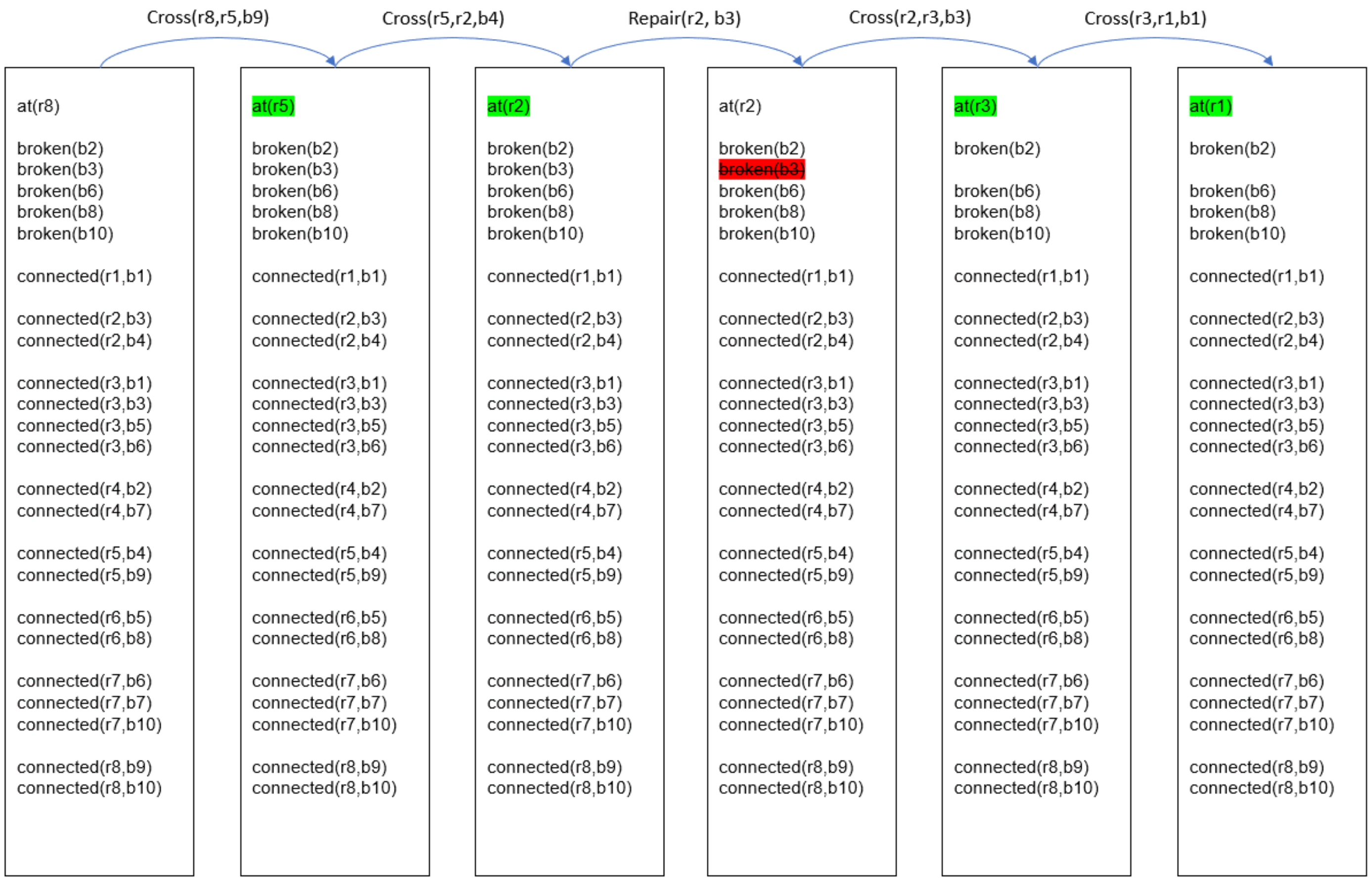
Samll error:
Should also include a notBroken(b) statement, since strips cannot contain negative attributes like !broken(b)

(b) You are given a map of your surroundings. Regions are labeled with $r_1, \ldots, r_8$ and bridges with with $b_1, \ldots, b_{10}$. Visual inspection tells you that the bridges b_2, b_3, b_6, b_8, and b_{10} are broken. Your initial position is the region r_8, while you desire to be at r_1. Formulate the initial state of the given planning setting and use *progression planning* to find a plan to reach the beach. What do the goal states look like?



Initial state: r8

Goal: r1



at(r8)	at(r5)	at(r2)	at(r2)	at(r3)	at(r1)
broken(b2)	broken(b2)	broken(b2)	broken(b2)	broken(b2)	broken(b2)
broken(b3)	broken(b3)	broken(b3)	~~broken(b3)~~		
broken(b6)	broken(b6)	broken(b6)	broken(b6)	broken(b6)	broken(b6)
broken(b8)	broken(b8)	broken(b8)	broken(b8)	broken(b8)	broken(b8)
broken(b10)	broken(b10)	broken(b10)	broken(b10)	broken(b10)	broken(b10)
connected(r1,b1)	connected(r1,b1)	connected(r1,b1)	connected(r1,b1)	connected(r1,b1)	connected(r1,b1)
connected(r2,b3)	connected(r2,b3)	connected(r2,b3)	connected(r2,b3)	connected(r2,b3)	connected(r2,b3)
connected(r2,b4)	connected(r2,b4)	connected(r2,b4)	connected(r2,b4)	connected(r2,b4)	connected(r2,b4)
connected(r3,b1)	connected(r3,b1)	connected(r3,b1)	connected(r3,b1)	connected(r3,b1)	connected(r3,b1)
connected(r3,b3)	connected(r3,b3)	connected(r3,b3)	connected(r3,b3)	connected(r3,b3)	connected(r3,b3)
connected(r3,b5)	connected(r3,b5)	connected(r3,b5)	connected(r3,b5)	connected(r3,b5)	connected(r3,b5)
connected(r3,b6)	connected(r3,b6)	connected(r3,b6)	connected(r3,b6)	connected(r3,b6)	connected(r3,b6)
connected(r4,b2)	connected(r4,b2)	connected(r4,b2)	connected(r4,b2)	connected(r4,b2)	connected(r4,b2)
connected(r4,b7)	connected(r4,b7)	connected(r4,b7)	connected(r4,b7)	connected(r4,b7)	connected(r4,b7)
connected(r5,b4)	connected(r5,b4)	connected(r5,b4)	connected(r5,b4)	connected(r5,b4)	connected(r5,b4)
connected(r5,b9)	connected(r5,b9)	connected(r5,b9)	connected(r5,b9)	connected(r5,b9)	connected(r5,b9)
connected(r6,b5)	connected(r6,b5)	connected(r6,b5)	connected(r6,b5)	connected(r6,b5)	connected(r6,b5)
connected(r6,b8)	connected(r6,b8)	connected(r6,b8)	connected(r6,b8)	connected(r6,b8)	connected(r6,b8)
connected(r7,b6)	connected(r7,b6)	connected(r7,b6)	connected(r7,b6)	connected(r7,b6)	connected(r7,b6)
connected(r7,b7)	connected(r7,b7)	connected(r7,b7)	connected(r7,b7)	connected(r7,b7)	connected(r7,b7)
connected(r7,b10)	connected(r7,b10)	connected(r7,b10)	connected(r7,b10)	connected(r7,b10)	connected(r7,b10)
connected(r8,b9)	connected(r8,b9)	connected(r8,b9)	connected(r8,b9)	connected(r8,b9)	connected(r8,b9)
connected(r8,b10)	connected(r8,b10)	connected(r8,b10)	connected(r8,b10)	connected(r8,b10)	connected(r8,b10)

Exercise 3.4

Exercise 3.4: You have recently noticed that some birds are regular visitors of your balcony, so your want to surprise them by building a small bird house in which you can put some seeds for them. You already bought a wooden kit and left it in your living room and there are some tools, such as a hammer and nails, in your basement. Since you finished your studying early, you have plenty of time and you decide to practice your AI skills by formalizing the process of gathering the essentials for building the bird house as follows:

$$
\begin{aligned}
&\textbf{Action}(\mathit{Take}(r,o)),\\
&\quad \mathsf{Precond}: \mathit{contains}(r,o) \wedge \mathit{in}(r),\\
&\quad \mathsf{Effect}: \mathit{holds}(o) \wedge \neg \mathit{contains}(r,o)\\
&\textbf{Action}(\mathit{Put}(r,o)),\\
&\quad \mathsf{Precond}: \mathit{holds}(o) \wedge \mathit{in}(r),\\
&\quad \mathsf{Effect}: \neg \mathit{holds}(o) \wedge \mathit{contains}(r,o)\\
&\textbf{Action}(\mathit{Move}(r_1,r_2)),\\
&\quad \mathsf{Precond}: \mathit{in}(r_1),\\
&\quad \mathsf{Effect}: \neg \mathit{in}(r_1) \wedge \mathit{in}(r_2)
\end{aligned}
$$

The meaning of the predicates is as follows:
$\mathit{in}(r)$: you are located in the room r,
$\mathit{contains}(r,o)$: the room r contains the object o,
$\mathit{holds}(o)$: you are holding the object o.

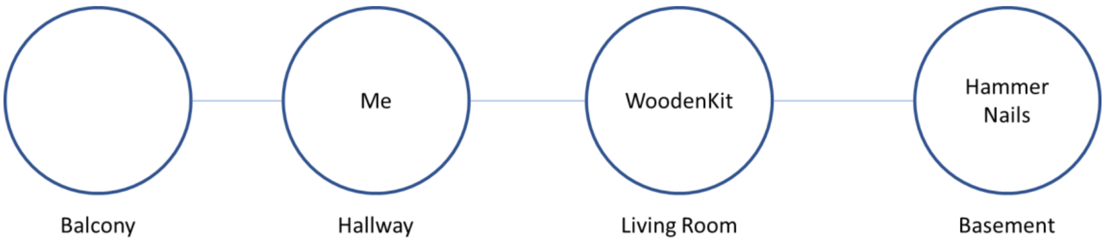
Furthermore, you specify your initial state as:

$$S := \{\mathit{in}(\mathit{hallway}), \mathit{contains}(\mathit{livingRoom}, \mathit{woodenKit}), \mathit{contains}(\mathit{basement}, \mathit{hammer}), \mathit{contains}(\mathit{basement}, \mathit{nails})\}.$$

The desired state is formalized as:

$$G := \{\mathit{in}(\mathit{balcony}), \mathit{contains}(\mathit{balcony}, \mathit{woodenKit}), \mathit{contains}(\mathit{balcony}, \mathit{hammer}), \mathit{contains}(\mathit{balcony}, \mathit{nails})\}.$$

Use the STRIPS state-space search algorithm starting in S (i.e., use *progression planning*) to determine the shortest possible plan that achieves the desired goal state G. You do not have to draw a complete graph. Simply show how the solution is found.



Move(H,LR)
Take(LR,WK)
Move(LR,BSM)
TAKE(BSM,H)
TAKE(BSM,N)
Move(BSM,LR)
Move(LR,HW)
Move(HW,BLC)
PUT(BLC,WK)
PUT(BLC,H)
PUT(BLC,N)

Abbreviations:

BLC … Balcony
HW … Hallway
LR … Living Room
BSM … Basement

WK … Wooden Kit
H … Hammer
N … Nails

Exercise 3.5

Exercise 3.5: Assume there is a lottery with tickets for €5 and there are three possible prizes: €100 with a probability of 0.1%, €50 with probability 0.2%, and €1 otherwise.

(a) What is the expected monetary value of a lottery ticket?

(b) When is it rational to buy a ticket? Give an inequality involving utilities with the following utilities: $U(S_k) = 0$, $U(S_{k+5}) = 5 \cdot U(S_{k+1})$, $U(S_{k+50}) = 35 \cdot U(S_{k+5})$, but there is no information about $U(S_{k+100})$. (S_n denotes the state of possessing n Euros.)

(c) Define $U(S_{k+5})$ and $U(S_{k+100})$ such that a rational agent whose utility function satisfies the equations in Subtask (b) chooses to buy a lottery ticket.

a) Monetary Value

$$(100-5) * 0.1\% + (50-5) * 0.2\% + (1-5) * 99.7\% = -3.803€$$

b) Rational to buy a ticket

$$\begin{aligned}
\mathrm{EU(buy)} &= 0.1\% * \mathrm{U}(\mathrm{S}_{k+100}) + 0.2\% * \mathrm{U}(\mathrm{S}_{k+50}) + 99.7\% * \mathrm{U}(\mathrm{S}_{k+1}) \\
&= 0.1\% * \mathrm{U}(\mathrm{S}_{k+100}) + 0.2\% * 35 * \mathrm{U}(\mathrm{S}_{k+5}) + 99.7\% * \mathrm{U}(\mathrm{S}_{k+1}) \\
&= 0.1\% * \mathrm{U}(\mathrm{S}_{k+100}) + 0.2\% * 35 * 5 * \mathrm{U}(\mathrm{S}_{k+1}) + 99.7\% * \mathrm{U}(\mathrm{S}_{k+1}) \\
&= 0.001 * U(S_{k+100}) + 1.347 * U(S_{k+1})
\end{aligned}$$

$$\mathrm{EU(!\,buy)} = U(S_{k+5}) = 5 * U(S_{k+1})$$

$$\begin{aligned}
EU(buy) &\geq EU(!\,buy) \\
0.001 * U(S_{k+100}) + 1.347 * U(S_{k+1}) &\geq 5 * U(S_{k+1}) \\
0.001 * U(S_{k+100}) &\geq 3.653 * U(S_{k+1}) \\
U(S_{k+100}) &\geq 3653 * U(S_{k+1})
\end{aligned}$$

c) Defining $U(S_{k+5})$ *and* $U(S_{k+100})$

$$U(S_{k+100}) \geq 3653 * U(S_{k+1})$$

$$U(S_{k+100}) \geq 3653 * \frac{1}{5} * U(S_{k+5})$$

$$U(S_{k+100}) \geq 730.6 * U(S_{k+5})$$

$$\begin{aligned}
&U(S_{k+1}) = 1 \\
&\rightarrow U(S_{k+5}) = 5 \\
&\rightarrow U(S_{k+100}) = 730.6 * 5 = 3653
\end{aligned}$$

Exercise 3.6

Exercise 3.6: In 1713, Nicolas Bernoulli investigated a problem, nowadays referred to as the *St. Petersburg paradox*, which works as follows. You have the opportunity to play a game in which a fair coin is tossed repeatedly until it comes up heads. If the first head appears on the n-th toss, you win 2^n Euros.

(a) Show that the expected monetary value of this game is not finite.

(b) Daniel Bernoulli, the cousin of Nicolas, resolved the apparent paradox in 1738 by suggesting that the utility of money is measured on a logarithmic scale, i.e., $U(S_n) = a\log_2 n + b$, where S_n $(n > 0)$ is the state of having n Euros and a, b are constants. What is the expected utility of the game under this assumption? Assume, for simplicity, an initial wealth of 0 Euros and that no stake has to be paid in order to play the game.

a) Infinite monetary value

$$EMV(1) = \frac{1}{2^1} * 2^1$$

$$EMV(2) = \frac{1}{2^1} * 2^1 + \frac{1}{2^2} * 2^2$$

$$EMV(3) = \frac{1}{2^1} * 2^1 + \frac{1}{2^2} * 2^2 + \frac{1}{2^3} * 2^3$$

…

$$EMV(n) = \sum_{i=1}^{n} \frac{1}{2^n} * 2^n$$

$$= \sum_{i=1}^{n} 1$$

$$= n$$

$$\lim_{n\to\infty} n = \infty$$

b) Expected Utility

$$EU(1) = \frac{1}{2^1} * a * ld(2^1) + b$$

$$EU(2) = \frac{1}{2^1} * a * ld(2^1) + b + \frac{1}{2^2} * a * ld(2^2) + b$$

…

$$EU(n) = \sum_{i=1}^{n} \frac{1}{2^i} * a * ld\left(2^i\right) + b$$

$$= \sum_{i=1}^{n} \frac{1}{2^i} * a * i + b$$

$$= \sum_{i=1}^{n} \frac{a * i}{2^i} + \sum_{i=1}^{n} \frac{b}{2^i}$$

$$= a * \sum_{i=1}^{n} \frac{i}{2^i} + b * \sum_{i=1}^{n} \frac{1}{2^i}$$

$$\lim_{n\to\infty} a * \sum_{i=1}^{n} \frac{i}{2^i} + b * \sum_{i=1}^{n} \frac{1}{2^i} = a * 2 + b * 1$$

$$= 2a + b$$