# B4B36ZUI - Introduction to Artificial Intelligence
# Example exercises

Ondřej Kubíček Karolina Drabent Viliam Lisý

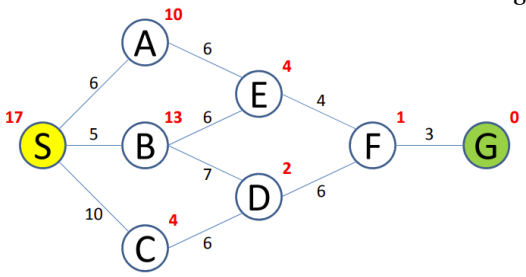
April 8, 2026

## 1 Search

### Exercise 1.1

**ID: 10369**

Consider the following graph with the numbers at the edges denoting the cost for traversing the edge and the numbers at the nodes the value of a heuristic function. S is the initial and G the goal node.




1. Is the denoted heuristic admissible? Explain why!
2. Is the denoted heuristic consistent? Explain why!
3. What value of the cost from the start node does each node have when it is discovered by algorithm A for the first time (first added to the queue)?

### Solution 1.1

(1) Yes, The values of the optimal heuristics are as follows
$h^*(S) = 18, h^*(A) = 13, h^*(B) = 13, h^*(C) = 15, h^*(D) = 9, h^*(E) = 7, h^*(F) = 3, h^*(G) = 0$
For each node $n$ holds $h(n) \leq h^*(n)$, so the heuristic is admissible.
2) No, there is an counterexample in for nodes $B, D$, because $13 = h(B) > h(D) + c(B,D) = 2 + 7 = 9$, which is in direct contradiction to the consistency
3) The A* will expand the nodes in following order and following costs when reached
$g(S) = 0, g(C) = 10, g(A) = 6, g(E) = 12, g(F) = 16, g(B) = 5, g(D) = 12, g(G) = 18$.
Note E and F will be expanded second time from B!.

### Exercise 1.2

**ID: 10161**

Consider a problem, where a robot needs to navigate through a grid-based warehouse of size $X+1 \times Y+1$. The robot needs to navigate through the warehouse to pick up all items from various locations and deliver them to designated drop-off points, which are located in the corners of the grid (coordinates for drop-off points are $(0,0),(X,0),(0,Y),(X,Y)$. At each timestep the robot can either move in 4 directions (up, left, right, down) or pick up one more item or drop all carried items. The amount of items on cell x, y is denoted as $n_{xy}$. Each item can be dropped of at any drop-off point. The agent has to end up in some drop-off position. Position of robot is denoted as $(x_r, y_r)$.
Decide which of the following heuristics are admissible:

- $\sum_{i \in X, j \in Y} n_{ij} \cdot (i + j)$

- $\min\{x_r, X - x_r\} + \min\{y_r, Y - y_r\}$
- $\sum_{i \in X, j \in Y} n_{ij} \cdot (|y_r - i| + |x_r - j|)$
- $\sum_{i \in X, j \in Y} sgn(n_{ij}) \cdot (\min\{i, X - i\} + \min\{j, Y - j\})$

## Solution 1.2

× $\sum_{i \in X, j \in Y} n_{ij} \cdot (i + j)$
✓ $\min\{x_r, X - x_r\} + \min\{y_r, Y - y_r\}$
× $\sum_{i \in X, j \in Y} n_{ij} \cdot (|y_r - i| + |x_r - j|)$
× $\sum_{i \in X, j \in Y} sgn(n_{ij}) \cdot (\min\{i, X - i\} + \min\{j, Y - j\})$

## Exercise 1.3

**ID: 8707**

Which of the following heuristic functions is admissible?

- A heuristic function that always overestimates the actual cost to reach the goal state
- A heuristic function that always underestimates the actual cost to reach the goal state
- A heuristic function that is independent of the state being evaluated
- A heuristic function that has a bounded error from the actual cost to reach the goal state

## Solution 1.3

× A heuristic function that always overestimates the actual cost to reach the goal state
✓ **A heuristic function that always underestimates the actual cost to reach the goal state**
× A heuristic function that is independent of the state being evaluated
× A heuristic function that has a bounded error from the actual cost to reach the goal state

## Exercise 1.4

**ID: 8708**

Which of the following algorithms expands the node that minimizes the heuristic function at each step in the search?

- Depth-first search
- Greedy best-first search
- A* search
- Breadth-first search

## Solution 1.4

× Depth-first search
✓ **Greedy best-first search**
× A* search
× Breadth-first search

# 2 Planning

## Exercise 2.1

**ID: 7891**

Mark all the operations we need to perform on a STRIPS action $a$, so that we transform it to its relaxed variant $a^+$.

- Delete its preconditions
- Delete its cost
- Delete its "add" effects
- Delete its "delete" effects

## Solution 2.1

- × Delete its preconditions
- × Delete its cost
- × Delete its "add" effects
- ✓ **Delete its "delete" effects**

## Exercise 2.2

**ID: 7892**

Which of the following claims about automated planning are true?

- Planning requires complete formal specification of the dynamics of the problem.
- A correct plan can always be found in time polynomial in the size of the planning problem, if it exists.
- One of the very commonly used algorithms in planning is $A^*$.

## Solution 2.2

- ✓ **Planning requires complete formal specification of the dynamics of the problem.**
- × A correct plan can always be found in time polynomial in the size of the planning problem, if it exists.
- ✓ **One of the very commonly used algorithms in planning is $A^*$.**

## Exercise 2.3

**ID: 12175**

Here is STRIPS formulation for egg frying recipe. Atoms: egg_at_fridge, pan_empty, pan_full, heat_on, heat_off, egg_broken, egg_fried Actions: move_egg(X,Y): //fridge to counter or back

- pre: egg_at_X
- add: egg_at_Y
- del: egg_at_X

put_egg_in_pan(X):

- pre: egg_at_X, pan_empty
- add: broken_egg, pan_full
- del: egg_at_X, pan_empty

fry_egg():

- pre: pan_full
- add: heat_on, egg_fried
- del: heat_off

Describe the initial situation:


Now we would like to have the action that puts the egg from the pan on the counter, but only if it is already fried. What should be added? Explain and write all necessary changes.

What are the limitations of this formulation?
How would you improve this formulation?

## Solution 2.3

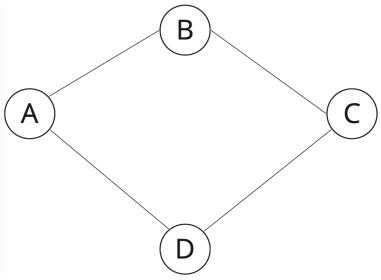
Initial situation:
egg_at_fridge, pan_empty, heat_off
Add plate_empty and plate_full atom.
The action itself:
put_egg_on_a_plate():

- pre: egg_fried, pan_full, plate_empty
- add: plate_full, pan_empty
- del: plate_empty

There are more correct answers than this one.
Limitation: There is only one egg. We represent only one egg's journey to become a fried egg. If someone wanted to represent 2 or more eggs fried at once we it's impossible, if this was supposed to be used in restaurant where eggs are fried sequentially this would not work.
Improvement(anything else that improves the formulation counts too, should include explanation):
Remove pan_empty and pan_full and simply operate on egg_at_pan
Some way of representing more eggs

## Exercise 2.4

**ID: 7864**




Consider the cities A, B, C, D connected with routes as shown in the picture. There is a package in the city A and a car in the city C. The car can move to a neighboring city and load or unload the package.
Using the STRIPS formalism, describe: (a) the initial situation (b) actions of loading and unloading the package, and moving the car

## Solution 2.4

(a) n(A,B), n(B,A), n(B,C), n(C,B), n(C, D), n(D,C), n(A,D), n(D,A) car_at(C), box_at(A), car_empty
(b) move(x,y) pre: car_at(x), n(x,y) add: car_at(y) del: car_at(x)
load(x) pre: car_at(x), box_at(x), car_empty add: car_full del: box_at(x), car_empty
unload(x) pre: car_at(x), car_full add: box_at(x), car_empty del: car_full

# 3 Reinforcement Learning and MDPs

## Exercise 3.1

**ID: 7413**

Assume that the agent in a multi-armed bandit problem observes the following sequence of rewards:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| A | 2 | | | | 2 | 1.5 | | 0.5 |
| B | | | 2 | 2 | | | 2 | |
| C | | 0.5 | | | | | | |

What is the probability distribution of choosing the following action by (a) greedy agent, (b) $\epsilon$-greedy agent with exploration 0.1 a (c) UCB agent with the exploration parameter set to 4. Write all three probability distributions.

## Solution 3.1

(a) 0.0; 1.0; 0.0 (b) 0.03333333333333333; 0.9333333333333333; 0.03333333333333333 (c) 0.0; 0.0; 1.0

## Exercise 3.2

**ID: 12177**

There is a hydra that protects a treasure and she has N heads. You want the treasure but you also want to live. Available actions for you are CUT, SING and RUN. Cutting a head off with your sword can regrow 2 heads with 50% probability. Singing a song, as out of tune as possible normally just makes Hydra a little irritated. But there is a small chance Hydra dies because she can't stand it. The more heads she has, the more ears she has so, she's more sensitive to your bad singing and the chance for her dying grows. Last option is that you can also run away. This works out 80% of the times and 20% Hydra is able to eat you because you turned your back unprotected.
What probability function would be most suitable for the SING action.
Draw an MDP representing this situation. Denote the states, actions, probability of the transitions and rewards.
Could this be also represented as multiarmed(or multiheaded ;) ) bandits problem? Why or why not?

## Solution 3.2

Drawing to be added! Let us say the probability of not surviving the SING with one head is $\epsilon$. One suitable probability function would for N heads would be following: Hydra dies with probability $\epsilon$ from the first head + the probability it survived the first head times probability of dying from second + probability of surviving 2 heads times the probability of dying from third head .. etc until N. Formula for this is $1-(1-\epsilon)^N$, which is 1 - (surviving with N heads)
No, it can't be represented as multiarmed bandits. Each action changes the state.

## Exercise 3.3

**ID: 7317**

Assume a simple deterministic gridworld, where a robot can move only left (L) or right (R); there is no cost for the move and the robot gets a reward of 4 for leaving the grid (e.g. it goes left in the left-most tile or right in right-most); discount factor 1; and a policy for all states $\pi(L|s) = \frac{1}{4}$ ; $\pi(R|s) = \frac{3}{4}$.
Assume the initial value function is

| | | | |
|---|---|---|---|
| 2 | 1 | 0 | 1 |

.
What is the value function after two steps of policy evaluation of policy $\pi$?

## Solution 3.3

Synchronous

| | | | |
|---|---|---|---|
| 1.75 | 0.5 | 1 | 3 |

| | | | |
|---|---|---|---|
| 1.375 | 1.1875 | 2.375 | 3.25 |

Asynchronous version (going from left) produces really wild results ($\frac{631}{256}$ in the right-most tile)

## Exercise 3.4

**ID: 12179**

Assume a following deterministic 1D gridworld with following Q-Values. The agent can go left or right in each tile. Going off the grid rewards him with reward 10, reward for other action is 0.

| 1 | 2 | 3 | 4 |
|---|---|---|---|

You are also provided following Q-value function, first table provides values for action < (moving left) and second for action > (moving right).

| 1 | -4 | 4 | -1 |
|---|---|---|---|

| 2 | -4 | 1 | 2 |
|---|---|---|---|

(a) Assume following trajectory 2<1>2>3<2<1< END (which is a sequence of states and actions). Compute the probability that Q-Learning with epsilon-greedy samples this trajectory if $\epsilon = 0.6$.
(b) Write down the formula for the Q-Learning value update using exponential moving average.
(c) Assume the same trajectory as in (a), write down the update for each step in the trajectory. Use $\alpha = 0.1$
(d) After your update, what is the policy the agent will use in the test phase.

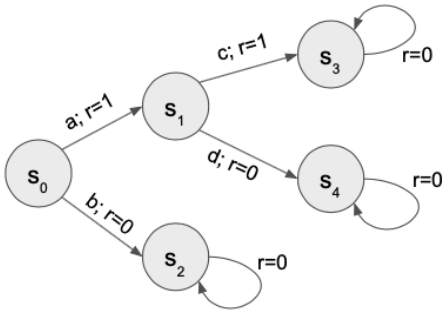
## Solution 3.4

(a) The probability of sampling the whole trajectory is a product of each step, which is a product of first sampling the action and then the probability that the action transitions to the next state. Here the probability of the transition is deterministic, so it boils down to only probability of sampling the actions. The greedy action is an action which provides maximum Q value. The probability of sampling greedy action is $\pi(a,s) = 1-\epsilon+\epsilon\frac{1}{|A|}$ and the probability of any other action is$\pi(a,s) = \epsilon\frac{1}{|A|}$ (b) $Q(s,a) = Q(s,a) + \alpha(R + max_{a'}Q(s',a') - Q(s,a))$, where $s, a, s'$ are current state, action taken and the next state respectively.

## Exercise 3.5

**ID: 7319**

Compute exact values of the **optimal state** and **state-action** value functions in all non-absorbing states of the following MDP, assuming the discount factor of 0.5.




## Solution 3.5

In order to compute the values for states $s_0, s_1$ we need to compute the absorbing states first, which contain only a single action so $V(s_i) = Q(s_i, -)$
The value in absorbing state is computed as infinite sum where each time $t$, the agent gets reward $r$, but weighted by the discount factor $\gamma^t$. Such an infinite sum has following form
$V(s_2) = 1/(1-\gamma)r = 0$
$V(s_3) = 0$
$V(s_4) = 0$
Now we can compute the Q values and out of those the V values. The optimal policy is such a policy that takes action which maximizes the value
$Q(s_1, c) = r + \gamma V(s_3) = 1 + 0.5 \cdot 0 = 1$
$Q(s_1, d) = 0$
$V(s_1) = \max_{A \in c,d} Q(s_1, A) = \max\{1, 0\} = 1$
$Q(s_0, a) = 1 + 0.5 \cdot 1 = 1.5$

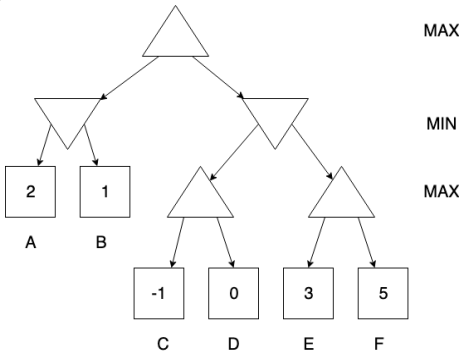
$Q(s_0, b) = 0$
$V(s_0) = \max\{1.5, 0\} = 1.5$

# 4 Games

## Exercise 4.1

**ID: 7460**

Consider the following two-player game:




1. Compute the value of the game.
2. Decide which leafs will be pruned-out by the alpha-beta pruning algorithm,

## Solution 4.1

The value of the game can be computed with alpha-beta
The algorithm explores left part of the game with $\alpha = -\infty, \beta = \infty$.
First there is an update $\beta = 2$ and then $\beta = 1$. This value is then propagated up, where the update is $\alpha = 1$.
Right part of the tree is explored with $\alpha = 1, \beta = \infty$.
First the left subtree is evaluated with the resulting value 0, which is propagated up for update $\beta = 0$.
Now $\alpha > \beta$. Therefore, we prune E a F.
1. Value of the game is 1
2. Nodes E a F

## Exercise 4.2

**ID: 7467**

Consider standard alpha-beta pruning algorithm with modified initial search window set to $[w, x]$. The algorithm returns a value $y > x$. Decide which of the statements are true.

- The exact value of the game is not known, but we can find it with repeated search with interval $[x, \infty]$
- The exact value of the game is not known, but we can find it with repeated search with interval $[y, \infty]$
- The value of the game is known and equal to $y$.
- The exact value of the game is not known, but we can find it with repeated search with interval $[x, y]$

## Solution 4.2

- ✓ **The exact value of the game is not known, but we can find it with repeated search with interval** $[x, \infty]$
- ✓ **The exact value of the game is not known, but we can find it with repeated search with interval** $[y, \infty]$
- × The value of the game is known and equal to $y$.
- × The exact value of the game is not known, but we can find it with repeated search with interval $[x, y]$

## Exercise 4.3

**ID: 7464**

Decide which of the following statements are true.

- Alpha-beta pruning always visits fewer nodes of the game tree compared to minimax.
- Negascout can visit some of the nodes in the game tree multiple times.
- Alpha-beta pruning searches the game tree in the breadth-first-search manner.

### Solution 4.3

- × Alpha-beta pruning always visits fewer nodes of the game tree compared to minimax.
- ✓ **Negascout can visit some of the nodes in the game tree multiple times.**
- × Alpha-beta pruning searches the game tree in the breadth-first-search manner.

### Exercise 4.4

**ID: 7465**

Decide which of the following statements are true.

- Negascout visits nodes in the game tree at most twice.
- Negamax visits exactly the same nodes in the game tree as minimax.
- Alpha-beta pruning can visit fewer different nodes in the game tree compared to Negascout.

### Solution 4.4

- × Negascout visits nodes in the game tree at most twice.
- ✓ **Negamax visits exactly the same nodes in the game tree as minimax.**
- × Alpha-beta pruning can visit fewer different nodes in the game tree compared to Negascout.

### Exercise 4.5

**ID: 12173**

Explain what does it mean that a game is "solved" (in the context of Zero-sum games). Based on your definition, which games out of these are solved

- Connect 4
- Chess
- Checkers
- Tic-tac-toe (gomoku on 3x3 board)

### Solution 4.5

- ✓ **Connect 4**
- × Chess
- ✓ **Checkers**
- ✓ **Tic-tac-toe (gomoku on 3x3 board)**