

1) True or False?

a) False

An agent does not need complete information about the environment in order to behave rationally. Rationality is defined relative to the information available to the agent. Even with partial perception, an agent can still choose the action that maximizes expected performance based on what it has observed and what it knows. Example: A vacuum cleaner robot may not know the exact layout of a house but can still act rationally by selecting actions that best achieve its cleaning goal given its limited sensors.

b) True

A simple reflex agent bases its decisions only on the current percept and ignores history. However, some environments require memory or state inference for rational behavior. Example: Consider a maze where two distinct locations produce identical sensor readings. The correct action depends on the previous path taken. Since a reflex agent cannot distinguish between these states based solely on the current percept, it cannot act rationally in such an environment.

c) True

Such environments can be constructed trivially. Example: Imagine an environment with only one state and a performance measure that assigns the same score to every action. Since no action can be better or worse than another, any possible agent behaves rationally in this setting.

d) False

Agent functions map percept sequences to actions. The set of possible agent functions is uncountably infinite, while the set of computer programs is only countably infinite. Therefore, there are more agent functions than programs, which means some agent functions cannot be implemented by any program. Additionally, some functions are not computable at all because they would require solving undecidable problems (e.g., variants of the Halting Problem).

e) False

Rationality does not imply guaranteed success in every single round, especially in games involving probability and hidden information. A rational agent selects actions that maximize expected utility, but outcomes can still be unfavorable due to chance. Example: A rational poker agent may go all-in with a statistically strong hand and still lose due to random card draws. Rational play maximizes long-term winnings, but it does not eliminate the possibility of short term losses.

2) Describing Environment Properties of Agents

a) Task Environment Selection

I have chosen autonomous underwater exploration as my task environment. This is done via autonomous underwater vehicles (AUVs), which are unmanned, untethered vehicles used to conduct underwater research. They are equipped with several different sensors and can be programmed to fulfill various tasks.

b) Observable

AUVs are mainly used to explore the sea floor or specific sites and has to rely on its sensors, like cameras and sonar. Because only the environment directly observed by the AUV is known, it is only partially observable.

c) Agents

In a mission is usually only a single AUV involved, at least in the immediate vicinity. It is therefore a single agent setting.

d) Deterministic Nature

The environment is stochastic as every action taken by the agent is subject to random chance. As an example could a changing current change the position of the AUV or a fish swimming past it be picked up by its sensors.

e) Episodic vs. Sequential

It is a sequential environment because every action taken by the AUV has an impact on its future actions, for example has every change in position, rotation or velocity an impact on its further navigation.

f) Dynamic Characteristic

The environment is dynamic, as the AUV has to navigate the sea, which changes regardless of the action or inaction of the vehicle.

g) Discreteness

As the environment of underwater exploration is the continuously changing sea, it is a continuous environment.

5) Binary tree traversals

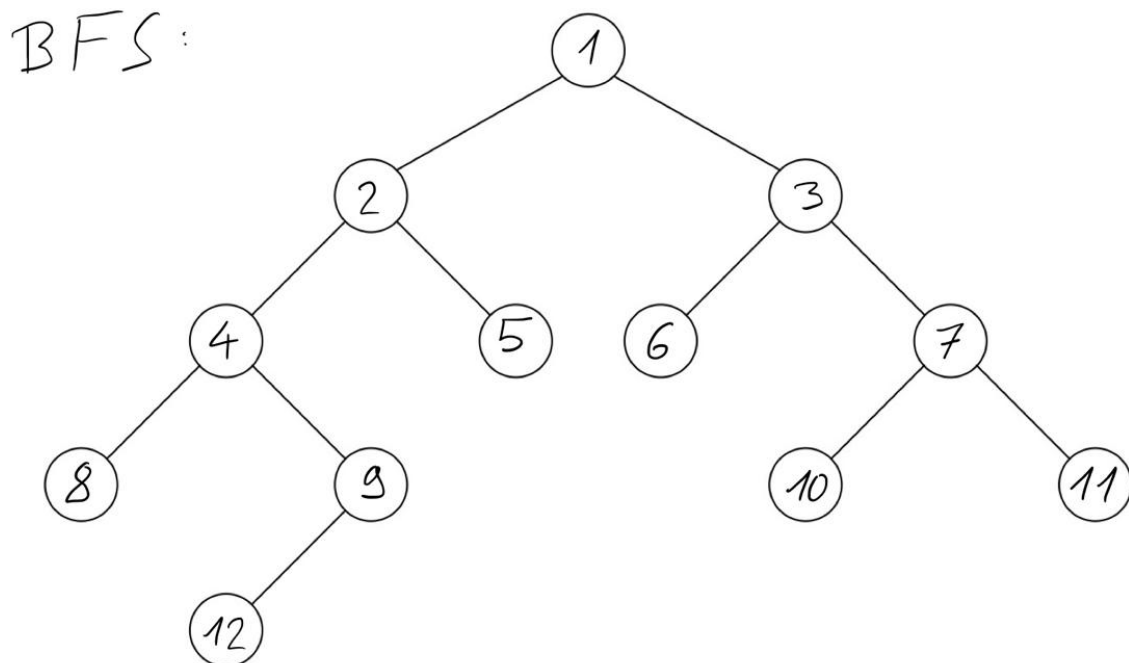


Figure 1: A binary tree.

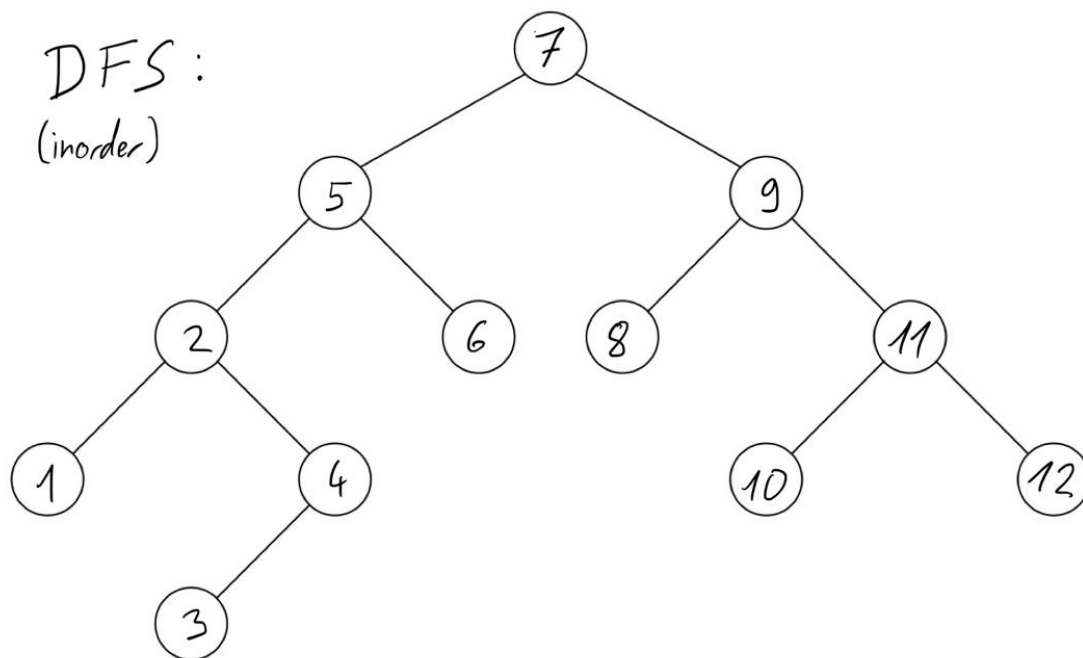


Figure 1: A binary tree.

6) Graph traversals

a) BFS

BFS : 1. Schritt Q = 0
 Queue Q 3, 5, 7
 FIFO 5, 7, 2
 7, 2, 6
 2, 6
 6, 1, 4
 1, 4
 4
 -

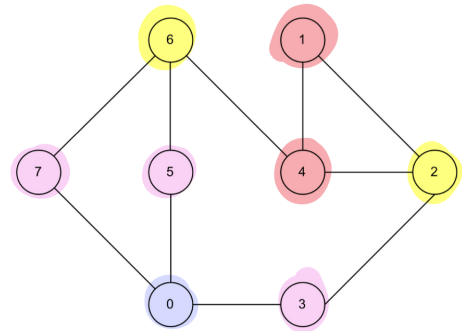


Figure 2: A 2-dimensional graph. Apply BFS and DFS starting at node 0.

b) DFS - lowest value first

DFS S = 0 3 2 1 4 6 5 7
 Stack S
 LIFO

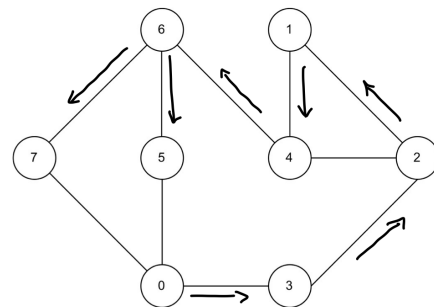


Figure 2: A 2-dimensional graph. Apply BFS and DFS starting at node 0.

b) DFS - highest value first

DFS S = 0 7 6 5 4 2 3 1
 Stack S
 LIFO

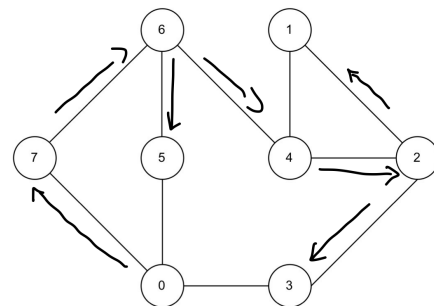


Figure 2: A 2-dimensional graph. Apply BFS and DFS starting at node 0.