# Lab session 2: Searching in "simple" tasks

Pierre-Alexandre Murena

April 12, 2024

GitHub repository: `https://github.com/ppaamm/TUHH-AICourse`

## 1 Depth-First Search for the monkey and banana problem

This section is taken from Lab 1, Section 5. It uses the code shared for Lab 1.

The planning agent implemented in class `PlanningAgent` explores several potential paths to get to the banana. At each time step, the agent simulates the result of all potential actions it can make, which results in a foreseen future environment (variable `env_copy`). The planning is then called recursively from this new state. Such a search is called **depth-first search**.

**Question 1.** Launch an instance of the planning agent. You can observe that the search enters an infinite loop. Can you explain why? Fix the code to make the agent avoid infinite loops.

## 2 The Tower of Hanoi

The Tower of Hanoi is a puzzle made up of three rods and a number $N$ of disks of various diameters. The game starts with all disks stacked on one rod in order of decreasing size (the biggest disk below, the smallest disk above). The goal is to move the stack to one of the other rods, obeying the following rules:

- Only one disk can be moved at a time;
- Only the upper disk of a stack can be moved, to be placed either on top of an existing stack or in an empty space;
- No disk can be placed on top of a smaller disk.

**Question 2.** Using the formalism presented in the lecture, describe the Tower of Hanoi as a search problem (i.e. states, initial state, goal, actions, transition model).

In the file `hanoi_tower.py`, you can find an implementation of Breadth First Search (method `bfs`) and Depth Limited Search (method `dls`).

**Question 3.** Run the BFS for various values of $N < 10$. What is the depth of the found solutions? Does this give any information on the depth of optimal solutions? Make a hypothesis on the depth of the optimal solution for a given $N$.

**Question 4.** Run the DLS for $N = 6$ with a depth limit of 1000. What is the depth of the found solution? Is it coherent or in contradiction with the answer to the previous question?

The problem of the Tower of Hanoi is an example of a problem which is extremely simple to formulate, but complicated to solve in practice. The best solutions use informed search strategy with well chosen heuristics.

## 3 Traveling throughout Wikipedia

In this second exercise, we consider a simple problem:

Given a starting page $P_{\text{start}}$ and an end page $P_{\text{end}}$ on Wikipedia, is there a trajectory on Wikipedia leading from $P_{\text{start}}$ to $P_{\text{end}}$?

For this problem, we use a directed graph of wiki links, in which:

- Each node corresponds to a Wikipedia page;
- An edge from node $n_i$ to node $n_j$ means that there is a link on page $i$ to page $j$.

We use a subgraph of the enwiki-2013 (`https://snap.stanford.edu/data/enwiki-2013.html`), which has a total of 10k nodes. All the methods for loading the dataset are available in the `wikigraph.py` file.

**Question 5.** Get familiar with the code. Create a `Wikigraph` object and explores some potential trajectories.

**Question 6.** What is the length of the optimal path between the pages "out of memory" and "Solar power in Germany"? You can use the search method of your choice. For the implementation of the search method, reuse the implementations used in the previous exercises.