**Task 1 Using State Space Search for Problem Solving**

**AI-lab 1 - Tower of Hanoi Problem**

Student 1: Joachim Johnson

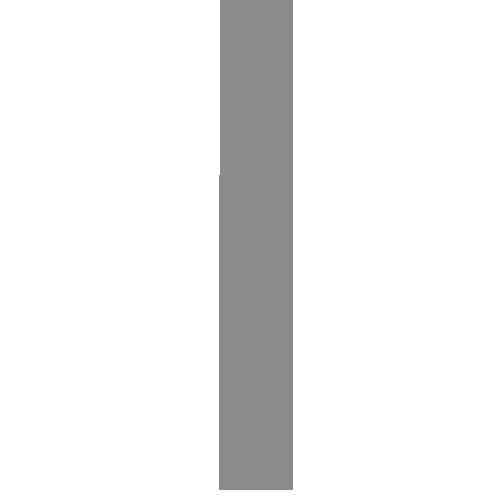
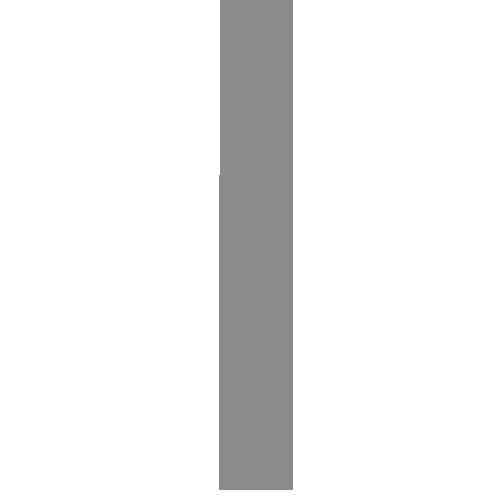
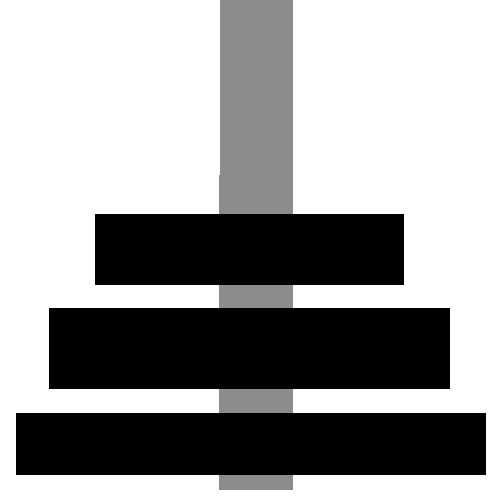
Student 2: Jonas Åsander

Course: Artificial Intelligence ([**DT112G**](https://lms.oru.se/webapps/blackboard/execute/launcher?type=Course&id=_13769_1&url=))

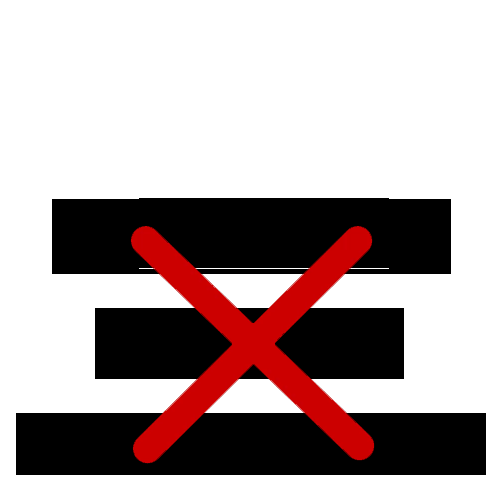
franziska.klugl@oru.se.

**Introduction:**

In this task do we solve the tower of Hanoi problem. Where we have 3 pins with 6 moveble disk in 6 different sizes there you could only move on at a time and only to another pin.



With the rules that it couldn't put a bigger block on a small one.



First we needed to make the structure that could maintain the search function itself

Secondly did we need to do the search algorithm itself.

This should be solved with programing with 2 kinds of search algorithms.

These 2 is deth first and breadth first algorithm and check the differences and outcome of these to see which one is shorter and which one passes less states.

Language is python.

The minimum where 3 pins and 3 disks. We did show the code with 6 disks and 3 pins.

**Data structure design:**

We have 2 global arrays that is the start and goal that's not really used other than start of the program and check if the goal is reacht.

We can explain the system the we use. start: (0.0.0.0.0.0) goal: (2.2.2.2.2.2)

Where the one to the left is the biggest and that to the right is the smallest disk.

and 0 mean it on the 1:e pin and 2 on the 3:e disk. Easy to implement in an array and move around without to move around in different arrays.

As you may see is the pins hard coded to 3 pins in the search algorithm and the number of disks is set by how big those 2 big global arrays.

**Overall processes description:**

We use Main() as start of our project that even keep the program rolling and call other functions. Why we let it rolling in that main is we wanted to keep the functions separated and clean and everything centralised send and receive data from the other def:s and send in data where its needed.

The search algorithms is it own def and is called from the running loop in Main function.

both return a value back after checking. This is practically whats ask after to do.

Whats we did here where 2 kind of search algorithms that we could do in 2 different files making it easy to use and show. Should only use it one at a time.

Main:

Take our frontier value and check the possible new state we can get from our frontier value, then put it our frontier value and put it in the ‘Visited’. We also save a copy of our new value and what value they come from.

After that we take away all of the new value we already visited so we not visit them again.

Then we check if our vistied is the desired goal value, and if it is we print out the way to finish our game.

obs:

in deepsearch we change the smallest first all the way we can change it before we go on the seconds smallest disk.

Newstate:

Check which disk that are movable, and then check the movable disk where u can put them and move them if they can.

then remove all ‘None’ value and return our new value as a array.

disk\_place:

check if u can move the disk, if u can return True, else False

Move\_disk\_one/Two\_step

change the valu of the disk

**Results and Discussion:**

The results of this task is code that can solve the problem with 2 different kind of algorithms.

With the problem did we get a few questions about the algorithms.

Questions relating to depth first and breadth first and how fitting they are in this situation and problem.

1. Do both algorithms produce a plan to solve the problem?

Yes they do.

2. Which one is shorter?

The deeps is shorter often shorter (in our case is was) because of its nature.

3. The algorithms use (“expand”) a number of states to search for the solution. Which one uses/passes less states?

The breath search use less move to get from left to right with all disk

Linkt files:

AI.py