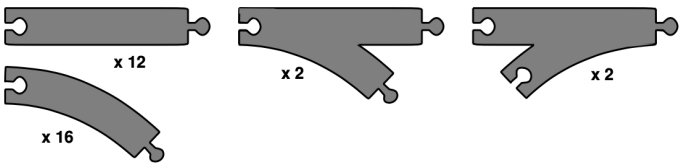
HOCHSCHULE LANDSHUT

# Artificial Intelligence I

*Lab 3 - Winter Semester 2023 / 2024*
https://moodle.haw-landshut.de/course/view.php?id=10282

**1.** A basic wooden railway set contains the pieces shown in the figure below. The task is to connect these pieces into a railway that has no overlapping tracks and no loose ends where a train could run off onto the floor.




(a) Suppose that the pieces fit together exactly with no slack. Give a precise formulation of the task as a search problem.

(b) Identify a suitable uninformed search algorithm for this task and explain your choice.

(c) Explain why removing any one of the "fork" pieces makes the problem unsolvable.

**2.** We can represent the 8-puzzle as a search problem by denoting each state by $x_{00}x_{10}x_{20}x_{01}x_{11}x_{21}x_{02}x_{12}x_{22}$ where $x_{ij}$ is the value at column $i$ and row $j$, $i, j \in \{0, 1, 2\}$, $x_{ij} \in \{0, \ldots, 8\}$. If the space is blank, the value is zero. The actions are: move blank space *up, down, left* or *right* (wherever possible). Each move has a cost of one. We will generate successors in the following order: **up, right, down, left**.

Consider the *Manhattan distance* - for every square the horizontal and vertical distances to that square's location in the goal state are added together; this value is then summed over all squares - as the heuristic function for A*-search. Note that the *Manhattan distance* of the two states

| | | |
|---|---|---|
| 2 | 5 | |
| 1 | 4 | 8 |
| 7 | 3 | 6 |

and

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

is calculated as

$$h_2(s) = 1 + 1 + 1 + 1 + 2 + 0 + 3 + 1 = 10.$$

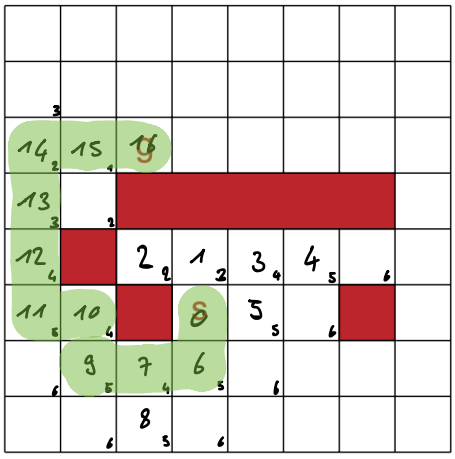
Generate the search tree of A*-search with the *Manhattan distance* heuristic

- starting from $250148736$
- with goal state $123456780$
- stating the values of $g$, $h$ and $f$ at each node in the tree

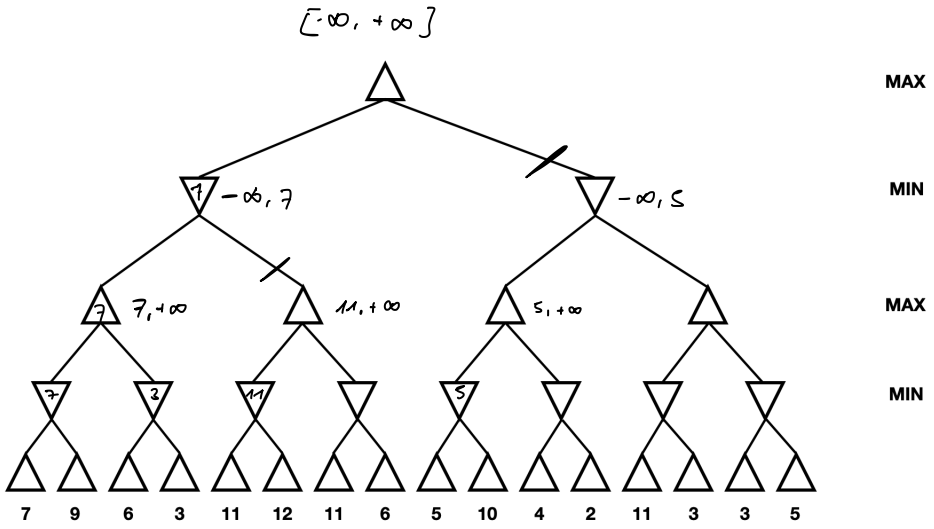
for the first five steps of the A*-algorithm. If there is a tie in the heuristic value, choose the state that comes earlier in lexicographical order. Mark the selected path in the tree.

**3.** Prove the following statement or give a counterexample: Uniform-cost search is a special case of A*-search.

**4.** On the grid below, number the nodes in order in which they are taken off the frontier for an $A^*$-search for the same graph. Manhattan distance should be used as the heuristic function. That is, $h(n)$ for any node $n$ is the Manhattan distance from $n$ to $g$. The Manhattan distance between two points is the distance in the $x$-direction plus the distance in the $y$-direction. It corresponds to the distance traveled along city streets arranged in a grid. For example, the Manhattan distance between $g$ and $s$ is $4$. What is the path that is found by the $A^*$-search?



**5.** The search tree for a two-player game is given in the figure below with the ratings of all leaf nodes. Use *minimax search with $\alpha$-$\beta$ pruning* from left to right. Cross out all nodes that are not visited and give the optimal resulting rating for each inner node. Mark the chosen path.




**6.** Explain why the following assertion is true: For every game tree, the utility obtained by MAX using minimax decisions against a suboptimal MIN will never be lower than the utility obtained by playing against an optimal MIN.

# KI Praktikum 03

### 1. Wooden Railways

- a) Find a connected combination of 32 rail track pieces, categorized into 4 different types, where no overlaps or lose ends occur. *(alternative: Find a combination of all rail tracks where the track forms a unknot like railway.)*
- b) For the problem, I would eighter use Depth-First-Search or Breadth-Frist-Search. The complexity of the problem has a fixed branching factor of 4 and a max depth of 32. Due to the limmited depth of 32 is would use the DFS.
  - Options: Depth first search Breadth first search Iterative deepening search Bidirectional search
- c) The problem with removing one of the fork pieces is, that every fork piece adds a new track but you have to have a second piece to be able to close it again. So you can only remove even numbers of forks.

### 3. *Uniform-Cost-Search == A*-Search*?

How I understood it, Uniform-Cost-Search (cheapest-cost-search) is just a special case of the A*-Search that has a easy heuristic, that sorts the frontier just based on the cheapest path cost.

### 6. Explain why the UTILITY of a MAX player using MINIMAX against a unoptimal MIN player will always be higher or equal than against a optimal MIN player.

Its because the MAX player always predicts all possible plays and the selects the best possible outcome if the min player plays optimaly. If the min player doesn't play optimal the margin is most of the time much higher because the min player makes mistakes which lead to higher utility for the max player.