Lab 3: Solve the Maze using A*

(Submission instructions can be found at the end of the document)

Deadline: 16 December 2022 at 16:00

Purpose of the task: Most humans naturally look for the shortest route between two points. Itsaves time, energy, and often headaches to find the speediest and most efficient path from point A to point B. However, that skill is no longer specific to living creatures.

Your Task: Use the A* algorithm to find the shortest path between two points (start, goal) in a maze, i.e., **solve the maze**.

The A* search algorithm is an extension of Dijkstra's algorithm useful for finding the lowest cost path between two nodes (aka vertices) of a graph. The path may traverse any number of nodes connected by edges (aka arcs) with each edge having an associated cost. The algorithm uses a heuristic which associates an estimate of the lowest cost path from this node to the goal node, such that this estimate is never greater than the actual cost.

There are many algorithms to generate mazes. However, in this lab you should use the **maze generator code** provided in **Learn/Uppgifter/Labs/ Lab 3**

A* search function:

$f(n)=g(n)+h(n)$

- $f(n)$ = total estimated cost of path through node n
- $g(n)$ = cost so far to reach node n
- $h(n)$ = estimated cost from n to goal. This is the heuristic part of the cost function

Now that you have a maze, start thinking about your algorithm. You will need to specify a start and goal state, you will need to calculate path-cost from node to node and you will have to calculate the cost from node to goal.

To help you, follow the detailed explanation provided in this tutorial: https://www.redblobgames.com/pathfinding/a-star/introduction.html Which walks you through Breadth First Search, Dijkstra's Algorithm and eventually the A* Algorithm. By incrementally implementing different search algorithms you may find it easier to arrive at the desired A* search solution required in this lab.

Other useful links:

- A* implementation example on a different maze: https://www.laurentluce.com/posts/solving-mazes-using-python-simple-recursivity-and-a-search/
- Theory and example: geeksforgeeks.org/a-search-algorithm/
- Theory: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

Submission instructions: Record a video in which you explain your code, run it and display the solution (i.e. the solution path) and upload it to your YouTube channel. Submit the working video link **and** the "Jupyter Notebook" containing your code solution in Learn under Assignments.