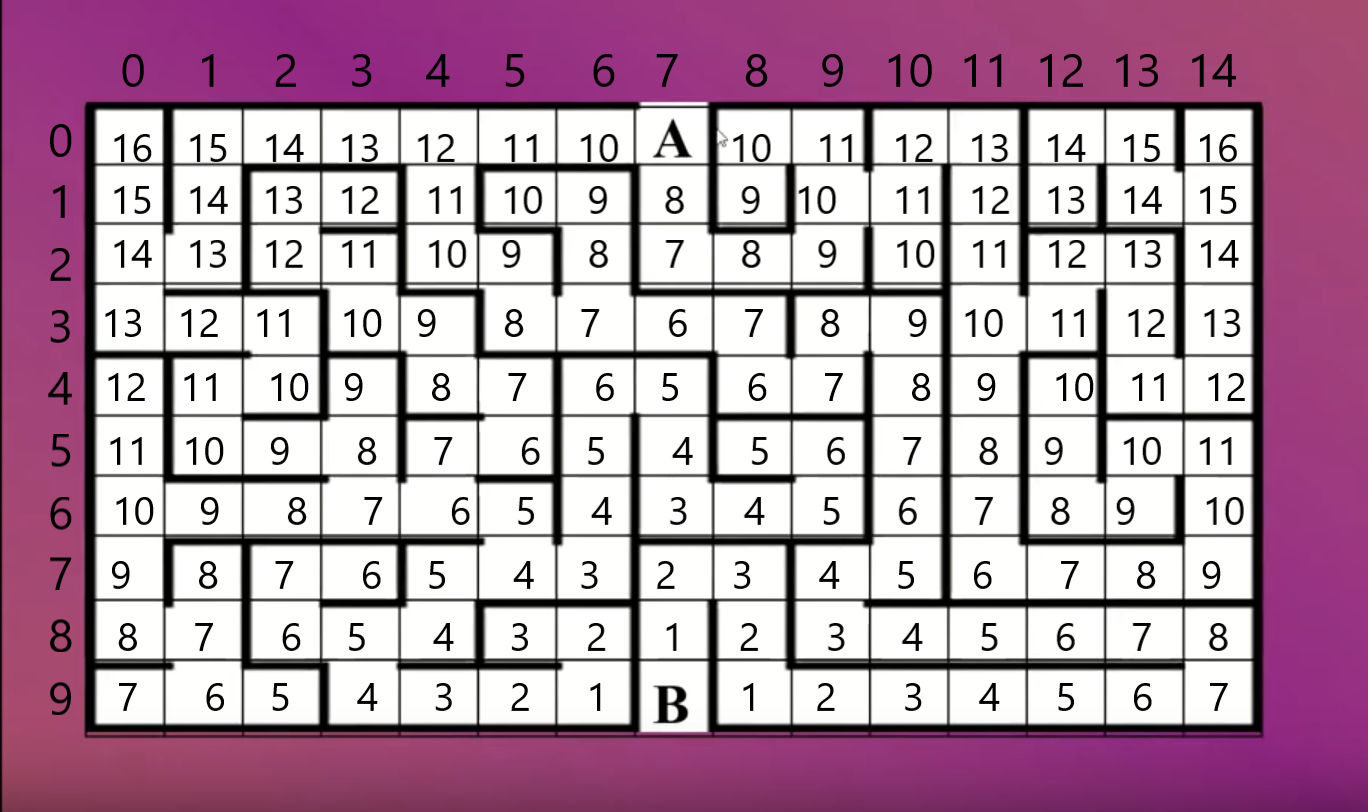
Data on A\* Search

Data are taken from my python program, Xela’s world, a 10 x 15 maze, which is solved by an A\* pathfinding algorithm with a heuristic function: f(n) = g(n) + h(n). For the experiment in which case it would work better, I tried to change only h(n) (heuristic value for each node). So each node has given a heuristic value. It has been calculated by the number of steps taken to reach a determinate node starting from goal node of y,x = 9,7. It means that the nodes directly adjacent to the goal node would have a heuristic value of 1, the next adjacent would have a heuristic value of 2, and so on. Instead, the g(n) function, which will not change in between the different data, would be just the number of steps taken from the starting goal. In other words the opposite of h(n).

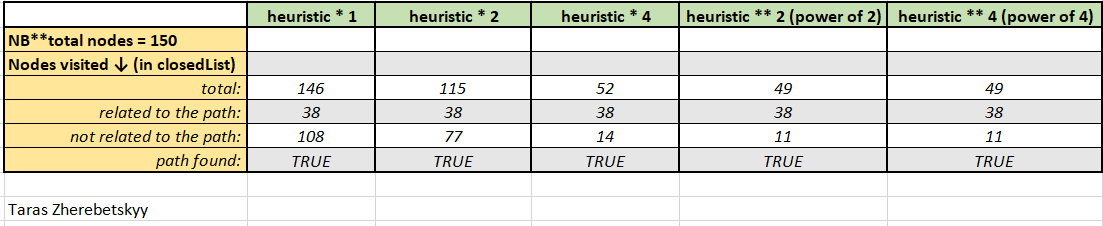
Here there is the graph of the maze with heuristic value for each node:



   This is the basic heuristic function I tried first, and it was not convincing me of its efficiency. What I tried then was tho multiply by a c (constant value) each h(n). the outcome is clear and it confirmed my suspected idea.

I multiplied the heuristic value by 2 for first then 4 and powered by 2 and 4 for the end. The total visited node (nodes appended in the closed list) was almost 3 times greater in the less efficient function compared to the most efficient(heuristic \*\* 2). The visited nodes not related by the shortest path were **10 times more** in the worst case compared to the best case.

Here are the data.



  This means that the algorithm, with the worst heuristic function, visited all the maze before finding the solution. So it would be more like a Dijaski algorithm than an efficient A\* pathfinding algorithm.