Stefan Hristov 40284739

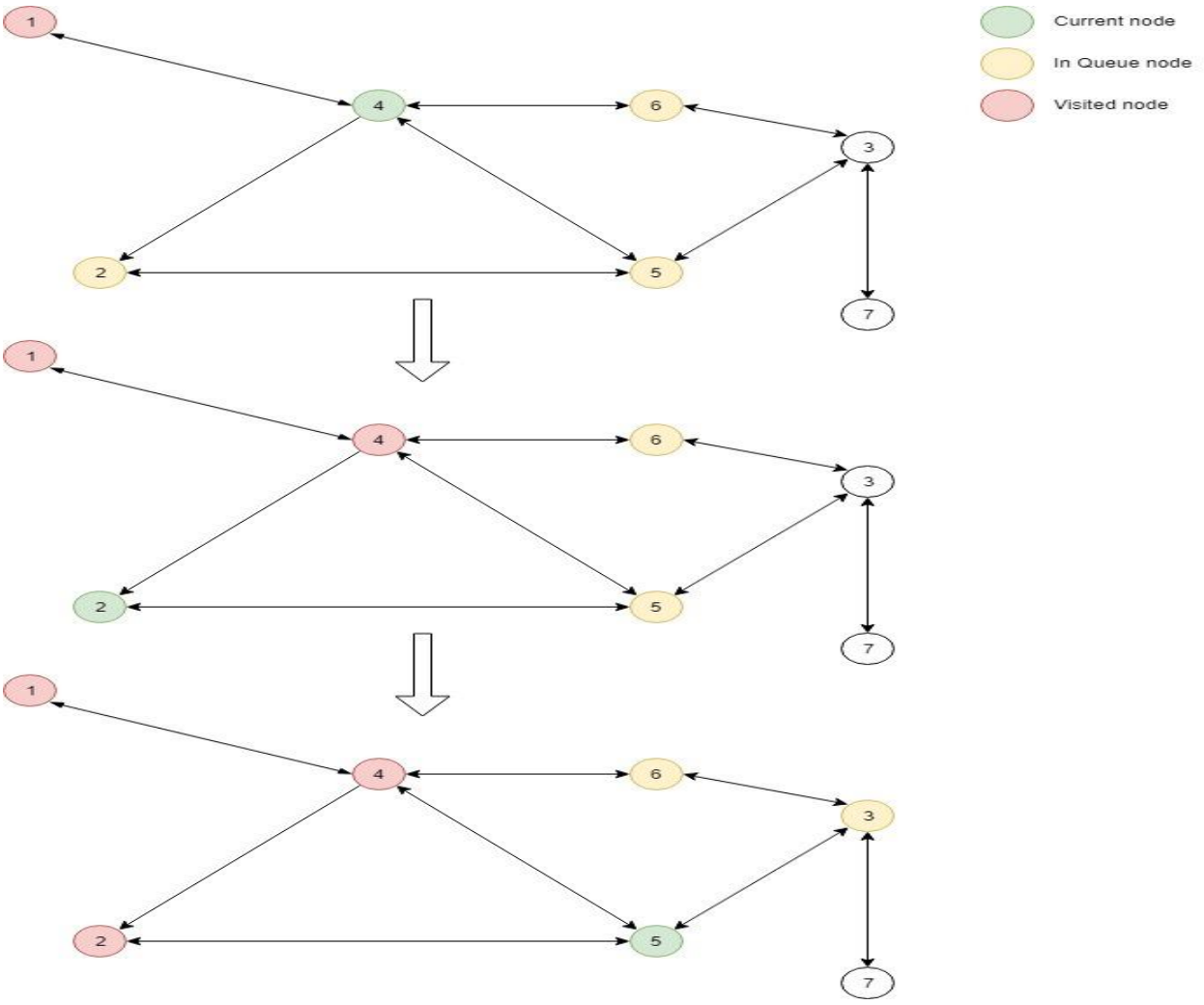
Artificial Intelligence

Chosen 3 algorithms

The task at hand is to find an algorithm which can find a path from a known start to finish point on a graph. This path should be as short as possible. The three algorithms below are **breath-first** search tree, **Dijkstra's** algorithm and **A star (A*)**.

Breath-first

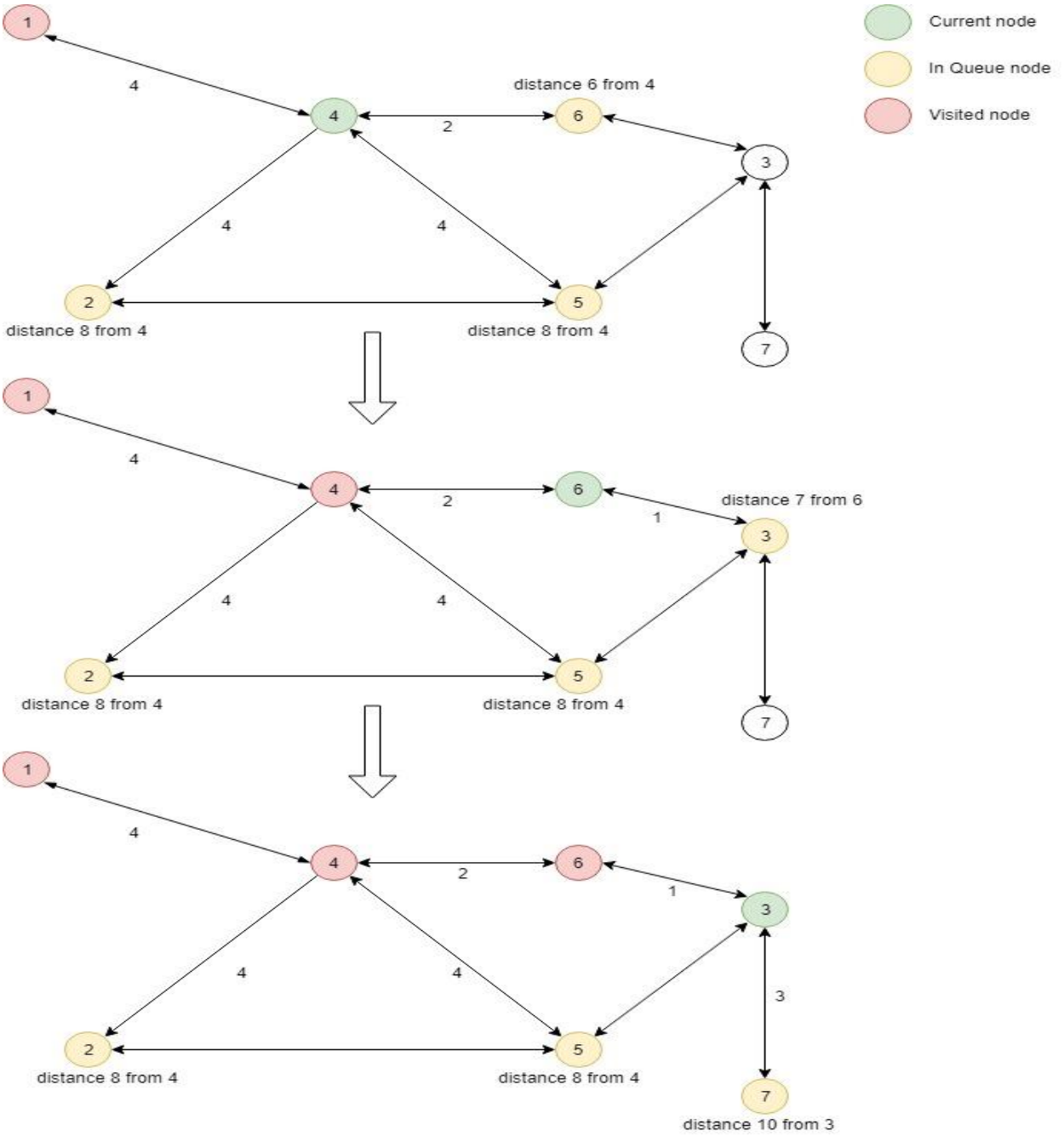
This algorithm requires an empty list for visited nodes and a queue list containing the starting node. It first checks for a direct path to the end node. If such a path is not found, the children nodes (nodes which have a direct path from the current node), if they are not in the visited or queue list, are added to the queue and their parent node is noted in them. The next element in the queue is popped, and the process above is repeated. If a direct path to the goal node is found, the path is traced back and returned. Otherwise, if the queue list becomes empty, it is concluded that there is no path from the start to the end node.



Stefan Hristov 40284739

Dijkstra's

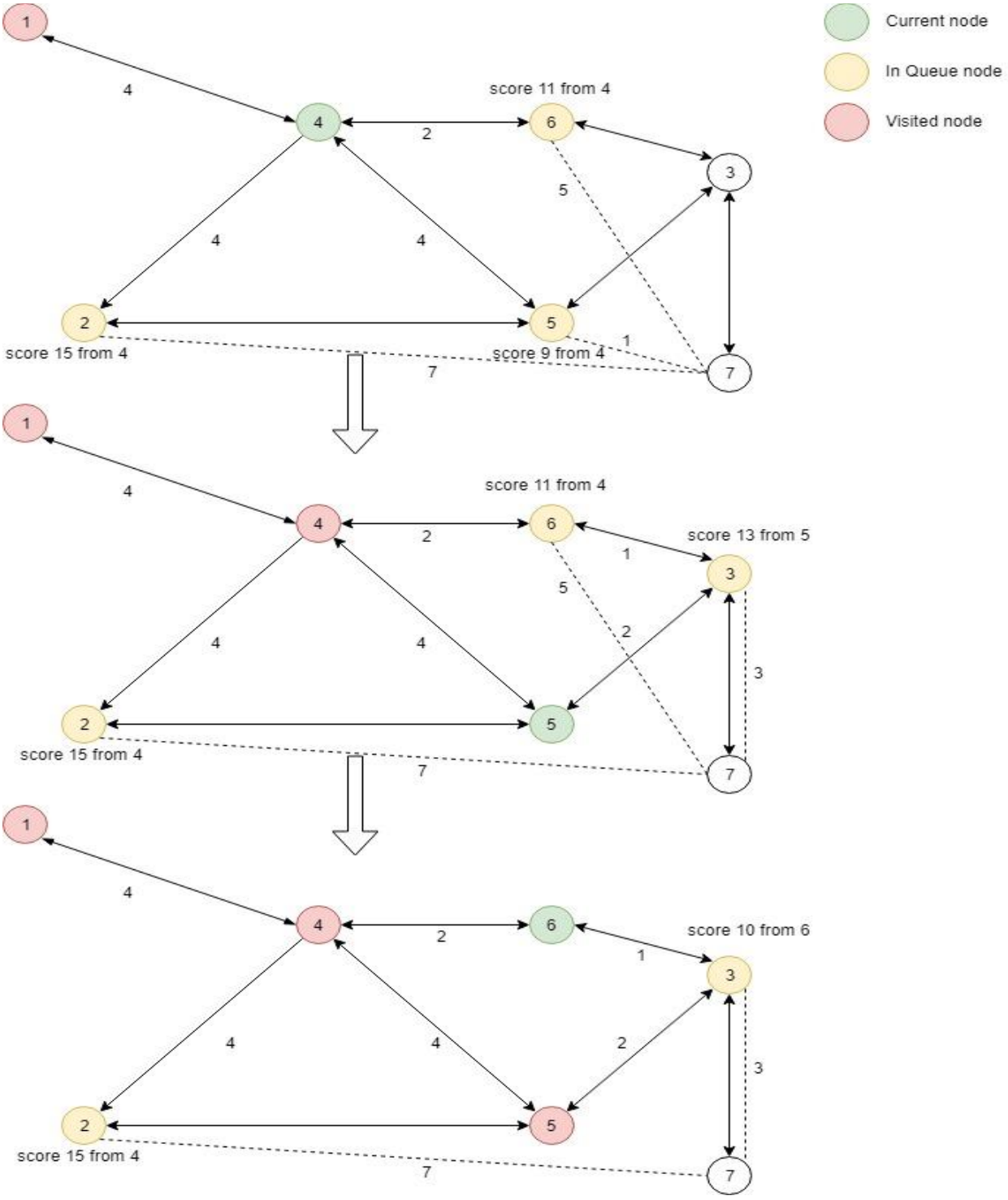
This algorithm is very similar to breath-first search with a few improvements. Firstly, the paths to each node are weighted. For this graph, distance is used. When a node is under consideration to be added to the queue, if it exists in the visited or queue list, then it compares the distance traveled. If the distance is shorter, the node is updated with the new shorter distance and the new parent node. Another improvement is that the queue is a priority queue. The elements are arranged so the one with the shortest distance is on top. Even if the end node is found, if there is a node with a shorter distance in the queue, it will be chosen as the next current node. As the example below demonstrates, even when we have found the goal, we will expand the next node in the queue which has lower distance traveled.



Stefan Hristov 40284739

A*

This algorithm is again very similar to the previous, but it adds a heuristic evaluation to the Dijkstra's approach. Instead of considering only the distance to a node, a heuristic evaluation is used. This evaluation is the distance traveled added to the Euclidean distance from the goal.



Stefan Hristov 40284739

Evaluation

Breath-first algorithm is a simple to implement algorithm which performs relatively well when the solution is not deep in the tree. The time and space complexity are both b^{d+1}. Overall that is not great for a large tree. In the context of the coursework, the shortest route is also preferred. This algorithm doesn't guarantee this.

Dijkstra's algorithm is an improvement to breath-first search which makes it a great algorithm used to this day. It is exhaustive, since it will check every possible solution which might exists. For the current context it is a good choice since it guarantees the best route and from the scale of the context it is a viable option.

The algorithm chosen is A*. This algorithm performs magnificent in producing fast results which are really good results. Not always the best since the heuristic could be over-estimated, but for the most cases a good fast result. For the context of the coursework, this algorithm was used because it provides fast solution for graphs with over 1000 caves but also guarantees the shortest route. Since the heuristic for this context is chosen to be distance, it can be accurately measured from the graph. Triangle theory states that no side of a triangle can be longer than the sum of the remaining 2. Therefore, the heuristic of nodes in the queue will always be considerate that might contain the shortest distance physically possible and will be expanded to confirm if they are. This way, A* guarantees the shortest route as well as avoids checking nodes which are not possible to lead to shorter path than the current.