DFS and BFS uses the same algorithm, except for the frontier method where DFS counts from the last one node appended and BFS from the first node appended. In maze samples I’ve seen that there is no best way to find the right route. I noticed that DFS is better on long routes and the BFS is better to find the best way especially in short maze. Anyway, is there a way to make our two algorithms more intelligent? Of course, yes. When a man arrives to a point where there is a decision to take, the human would take the route that points to the exit of the maze. It’s not always the right way, but if the A point is in the bottom left corner and B point is in the upper right corner, when the human arrives to a point where one route goes to the left and the other one to the right, it would decide the second one, the one which points the B point. Here they come uninformed search and informed search:

-Uninformed Search: these types of algorithms are algorithms that do not know about the problem to solve. They do not care if they are working with a maze, a puzzle or what else. Like DFS and BFS do, they try all the possible actions without knowing that they are working with the maze or the 15 puzzle.

-Informed Search: strategies that use problem-specific knowledge to find solutions in an efficient way.  
An algorithm that is based on an informed search is the Greedy Best-First Search.

-GBFS: Instead of expanding by going to the deepest node (DFS) or the nearest node in all the states (BFS), this algorithm will expand pointing to the nodes that thinks are near to the goal state.   
Since we don’t know where the end is (although we already have a solution) we can estimate where the goal state is by using a heuristic (something before the confirm) function called h(n). This function takes as input our state and returns how close it is. The functions ask itself this question:” Is it better to be to an hypothetic C point which is on the left (far from the exit) or on an hypothetic D point which is the nearest?”  
Since being in D point (which is the nearest one) is better, the algorithm will try to explore to that direction.  
Remember that the heuristic function works without considering the presence of wall or something that could change the path, so, due to this, the function won’t be 100% right.

If the Informed Search are better, why don’t we use only this kind of AI? Actually, since we can give information about our states, it may seem that the AI will find better results but, since GBFS is based on and heuristic function n(h), programmers have to develop a good heuristic function and, the more hard is the problem, the hard will be the function to develop and vice versa.   
In a medium long maze, we can decide to use the BFS to find the fastest path, but this may will explore all the possible states. By using the DFS we are In the hand of the luckiness so the AI may find the best path with exploring the less as possible or the worse path exploring a lot. Using the GBFS the AI will use the logic because its choices are based on hypothetical knowledge so it will not be the best path.

Because we like the idea of heuristic function, we improve this feature and the function starts to consider also how much took to get to that distance between the actual state and the Exit. This is called A\*Search.

-A\*Search: we add to h(n) function which counts the estimated cost to goal state, the g(n) function, which returns the cost to reach the actual node. The best way to choose is the one that has the lowest value of h(n) + g(n). “We are 16 blocks away h(n) from the end and it took me 1 block to get here g(n) so I am at 17. 15 blocks away and took 2 blocks = 17. If I go on the left, I’m 14 blocks away and 3 of cost = 17. Then is starts to improve the distance and I am at 15+4=19. If I follow the other path, I would be at a lower value so let’s try the other one path.” This is the reasoning that an AI would follow, so it may stop to follow a certain route and start another one.  
Why don’t developers use this A\*Search algorithm? Because it is a good solution only if h(n) function doesn’t overestimate the true cost and if h(n) is consistent which means that the next node h(n) value should not be higher than h(n’)+c where N’ is the next node and C the step cost. The most important thing is the heuristic function. The better is the function the better will be the solution.  
A\*Search, due to storing the information, usually occupies more memory so we might look to lighter algorithms or something else.