-Approach:

a frontier that contains the initial state (the initial state is where we start from)

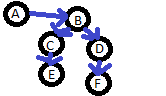
repeat (repeat a process)

if the frontier is empty it means there is no solution

else remove a node from the frontier

and if the node contains the goal state, return the solution

else continue to expand trying to perform all the possible actions



1. Starting with state A
2. It’s not empty so remove it but still considering A state
3. A is not the goal state so it expands and reaches B
4. Same passages of points 2 and 3
5. Now there are now C and D. Let’s consider C
6. Repeat the passages and there is E state
7. It’s our goal state so it will return the solution to reach E

During this process there may occur a problem. Like the 15 puzzle, after moving a tail on the left, the next action allows us to remove it to the right, so is like going from A to B and then do from B to A, entering a loop and slowing down things.

Instead of using approach, we can use Revised Approach:

Still starting from an initial state

Defining an empty explored set

Repeat

if the frontier is empty it means there is no solution

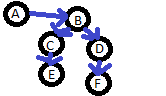
else remove a node from the frontier

and if the node contains the goal state, return the solution

add the node to the explored set

expand the node, add it to the explored nodes if they aren’t already in the frontier or in the explored set

But, how do the AI decide which path follow if there are more nodes connected?



There is “rule” which says: the last thing I added is the first thing I will remove so:

1. Starting with A, removing it from the frontier and adding it to the explored set
2. Now at B, it can go to A, C or D.
3. A is already in the explored set so in the frontier there are only C and D nodes.
4. Since D was the latest add node it will continue with D
5. In the Frontier there Are now C and F while in the Explored Set there are A,B,D nodes
6. F is now empty so there is no result and in frontier there is only C node.
7. It continues with C node and finds E.
8. E is the Goal State so it return the solution

This version of algorithm to find a solution is called: Depth-First Search.

Depth-First Search is an algorithm which mines to the deepest node of our diagram. If F had 100 hundred nodes but the result is E, this algorithms would try all that 100 possibilities before trying to expand C and finding the solution. The opposite of Depth-First Search is the Breadth-First Search where instead of using a stack to expand nodes (first in-last out) it uses a queue (first in-first out) so it would be, in order of exploring, (A,B,C,D [E is already in the frontier at this point, but D’s been added before so gets explored before],E [should stops here because E is the goal state] and F).

Depth-First Search is going to always get a solution. In case of an escape from a maze, it will explore the possible choices before finding the exit, but it will find it. Otherwise, in case of 2 options which are the same right, it will return only the first solution it finds, not the best one.

Immagine che contiene screenshot

Descrizione generata automaticamente

In DFS it may choose to go in the longest direction and explore the deepest possible till it finds the goal state.

In BFS it explore one times in a path and the other time in the other path. (At 8th action it should go straight and up, not only up, my bad).

None is the best, it depends from case to case. In this one the BFS is the best because finds the best way instead of DFS which found the longest route