Analysis of backtracking algorithm for 8-queens problem

Marius Stokkedal, matk20@student.bth.se, DV2619

Background

DFS (depth first search)

DFS uses a stack, LIFO (last in first out). When traversing the nodes of the tree structure which is being searched the search begins in the root node and moves through the nodes as far as possible until a node with no unvisited neighbors is reached. DFS is most suitable when the solution is far away from the start-node.

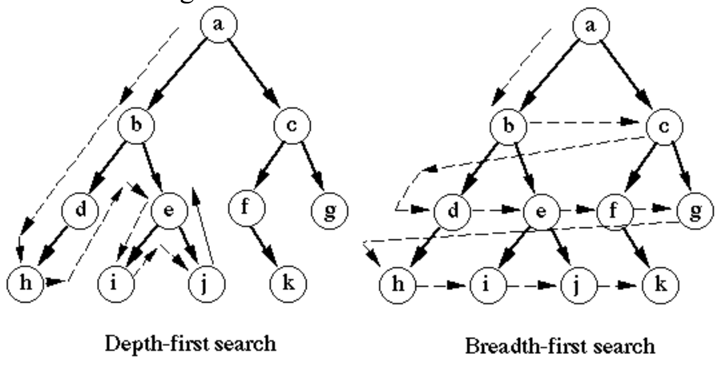
DFS is more suitable for puzzle problems, such as sudoku and chess since a decision is made, all paths based on this decision are explored and if this decision leads to a win the search is completed. When visiting a node, the next node(s) in line is its children. The visited node is added to the stack and is removed when all its children has been visited. DFS is suitable for applications where memory is limited, but not when searching for the shortest path. It uses less space than BFS (breadth first search) because at one time the maximum number of nodes stored is the path from the root node to a leaf node. [1]

BFS (breadth first search)

BFS uses a queue, FIFO (first in first out). When traversing a tree structure using the BFS approach every level is searched before moving further. Since BFS searches every level rather than every path after the root node it is therefore more suitable when searching for the shortest path between nodes in a graph.

A BFS starts at a given node and from there it continiues to all nodes one step away from the start node, then two steps away from the start node and so on until the goal is reached. When the searched node is found it is certain that the current path taken will be the shortest. If a shorter path would exist, the node would have been found earlier on in a different level of the tree. [2] [3]

DFS and BFS is visualized in the figure below.



[Figure 1] [4]

DFS (depth first search) vs BFS (breadth first search)

For a fair comparison, the two algorithms were run ten times on the same computer which was not being used for anything else during the test for minimal impact on the test results. The resulting times is displayed in the table below.

Run	**Time DFS**	**Time BFS**
1	0.077	0.104
2	0.057	0.102
3	0.056	0.100
4	0.055	0.102
5	0.055	0.101
6	0.057	0.103
7	0.090	0.122
8	0.065	0.102
9	0.066	0.102
10	0.069	0.101
Mean:	**0.065**	**0.104**

As the times in the table above shows, the BFS has a mean time for execution which is 60% higher than that of DFS. This reflects what was mentioned earlier in the rapport, a DFS performs better on a puzzle problem; which is a category this problem fits right in to.

Time complexity

When placing the first queen on the board of size n•n there are n possible columns to choose from, and to find all solutions all must be tested. The second queen can not be placed so that she is under attack from another queen, there are therefore n-1 possible columns to choose from and so on for the following queens. When one column is chosen for the second queen there will be $n - 2$ possible positions because one square is occupied by the first queen's sideways movement and another square will be occupied by the queen's diagonal movement. When positioning the third queen there will be two squares occupied by the other two queens and at least one square occupied by the two queens' diagonal movement. Instead of calculating the possibilities as $n \bullet (n-2) \bullet (n-4) \ldots$ we can use the definition of Big-O and calculate $O(n!)$ since $n \bullet (n-2) \bullet (n-3) \ldots \leq n!$ which means that $O(n \bullet (n-2) \bullet (n-3) \ldots) = O(n!)$.

Backtracking

When using an algorithm with backtracking the search will be more effective when searching for puzzle solutions like this one. When an illegal placement is performed it will be discovered directly and that path will not be explored further and thereby avoiding dead ends. If an algorithm like brute force would be used instead all possible placements would be tested which would result in many more paths being explored. The time complexity for brute force is $O(n^n)$ since there are n possible columns with n possible squares for each. In the case with an 8•8-board $n^n = 16777216$ and $n! = 40320$ which means that $O(n^n) > O(n!)$.

Works Cited

[1] MKS075, "geeksforgeeks," 2022. [Online]. Available: https://www.geeksforgeeks.org/difference-between-bfs-and-dfs/. [Accessed 11 09 2022].

[2] H.-P. Störr, "stack overflow," 01 12 2021. [Online]. Available: https://stackoverflow.com/questions/3332947/what-are-the-practical-factors-to-consider-when-choosing-between-depth-first-sea. [Accessed 11 09 2022].

[3] Y. U. Vaity, "Stack overflow," 03 07 2017. [Online]. Available: https://stackoverflow.com/questions/3332947/what-are-the-practical-factors-to-consider-when-choosing-between-depth-first-sea. [Accessed 11 09 2022].

[4] T. NG, "BFS and DFS," 15 02 2014. [Online]. Available: https://medium.com/@tim_ng/bfs-and-dfs-52d3cb642a0e. [Accessed 11 09 2022].