DESIGN & ANALYSIS OF ALGORITHMS (5)

techworldthink • March 10, 2022

15 Write the Backtracking algorithm for N-Queen Problem.

Backtracking is an algorithmic-technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time (by time, here, is referred to the time elapsed till reaching any level of the search tree).

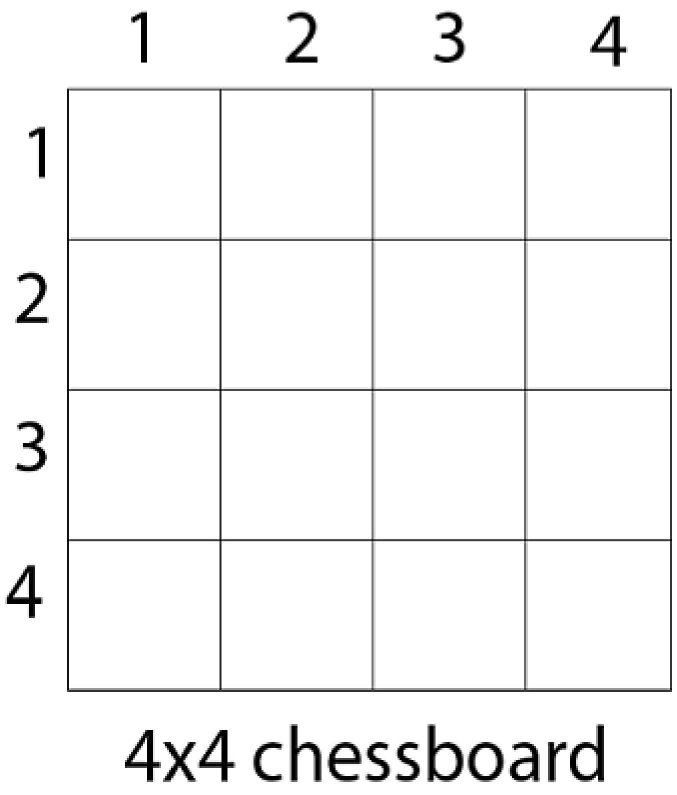
Backtracking can be defined as a general algorithmic technique that considers searching every possible combination in order to solve a computational problem.

N-Queens Problem

N - Queens problem is to place n - queens in such a manner on an n x n chessboard that no queens attack each other by being in the same row, column or diagonal.

It can be seen that for n =1, the problem has a trivial solution, and no solution exists for n =2 and n =3. So first we will consider the 4 queens problem and then generate it to n - queens problem.

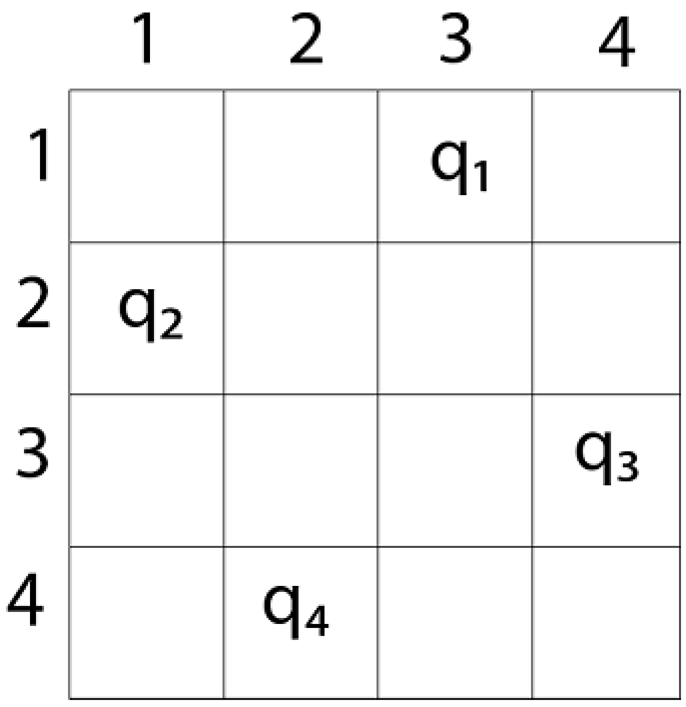
Given a 4 x 4 chessboard and number the rows and column of the chessboard 1 through 4.



N-Queens Problem

Since, we have to place 4 queens such as q1 q2 q3 and q4 on the chessboard, such that no two queens attack each other. In such a conditional each queen must be placed on a different row, i.e., we put queen "i" on row "i."

Now, we place queen q1 in the very first acceptable position (1, 1). Next, we put queen q2 so that both these queens do not attack each other. We find that if we place q2 in column 1 and 2, then the dead end is encountered. Thus the first acceptable position for q2 in column 3, i.e. (2, 3) but then no position is left for placing queen 'q3' safely. So we backtrack one step and place the queen 'q2' in (2, 4), the next best possible solution. Then we obtain the position for placing 'q3' which is (3, 2). But later this position also leads to a dead end, and no place is found where 'q4' can be placed safely. Then we have to backtrack till 'q1' and place it to (1, 2) and then all other queens are placed safely by moving q2 to (2, 4), q3 to (3, 1) and q4 to (4, 3). That is, we get the solution (2, 4, 1, 3). This is one possible solution for the 4-queens problem. For another possible solution, the whole method is repeated for all partial solutions. The other solutions for 4 - queens problems is (3, 1, 4, 2) i.e.



N-Queens Problem

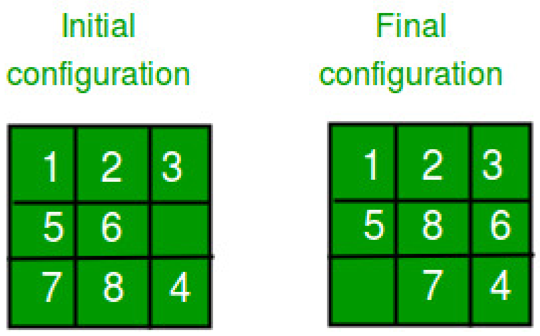
Algorithm

```
1.Start in the leftmost column
2.If all queen are placed
    return true
3.Try all row in the current column
     Do following every for tried row
          a) if the queen can be placed safely in the row then mark this [row,column] as
             part  of the solution & recursively check if placing queen have leads to a
              solution
          b) If placing the queen in [row,column]  leads to a solution then return true
          c) If placing the queen doesn't leads to a solution then unmark this
[row,column]
              (backtrack) and goto step(a) to try another rows
4. If all rows have been tried and nothing worked return false to trigger backtracking
```

16 Explain the 8-puzzle problem and its solution using branch and bound technique.

Given a 3×3 board with 8 tiles (every tile has one number from 1 to 8) and one empty space. The objective is to place the numbers on tiles to match the *final configuration using the empty space. We can slide four adjacent (left, right, above, and below) tiles into the empty space.*

For example,



8puzzle

Branch and Bound

The search for an answer node can often be speeded by using an “intelligent” ranking function, also called an approximate cost function to avoid searching in sub-trees that do not contain an answer node. It is similar to the backtracking technique but uses a BFS-like search.

There are basically three types of nodes involved in Branch and Bound

1. Live node is a node that has been generated but whose children have not yet been generated.

2. E-node is a live node whose children are currently being explored. In other words, an E-node is a node currently being expanded.

3. Dead node is a generated node that is not to be expanded or explored any further. All children of a dead node have already been expanded.

Cost function:

Each node X in the search tree is associated with a cost. The cost function is useful for determining the next E-node. The next E-node is the one with the least cost. The cost function is defined as

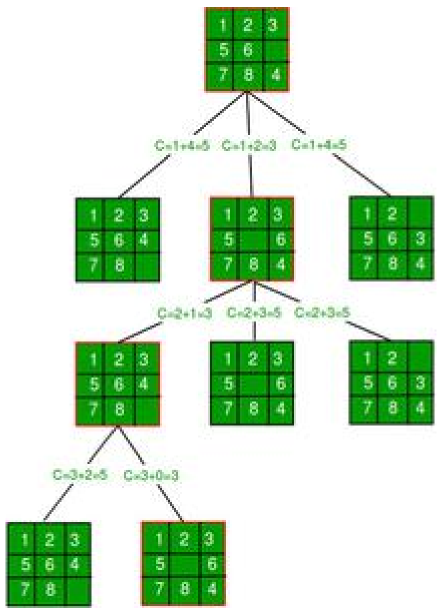
```
   C(X) = g(X) + h(X) where
   g(X) = cost of reaching the current node
          from the root
   h(X) = cost of reaching an answer node from X.
```

The ideal **Cost function for** an **8-puzzle Algorithm :**

We assume that moving one tile in any direction will have a 1 unit cost. Keeping that in mind, we define a cost function for the 8-puzzle algorithm as below:

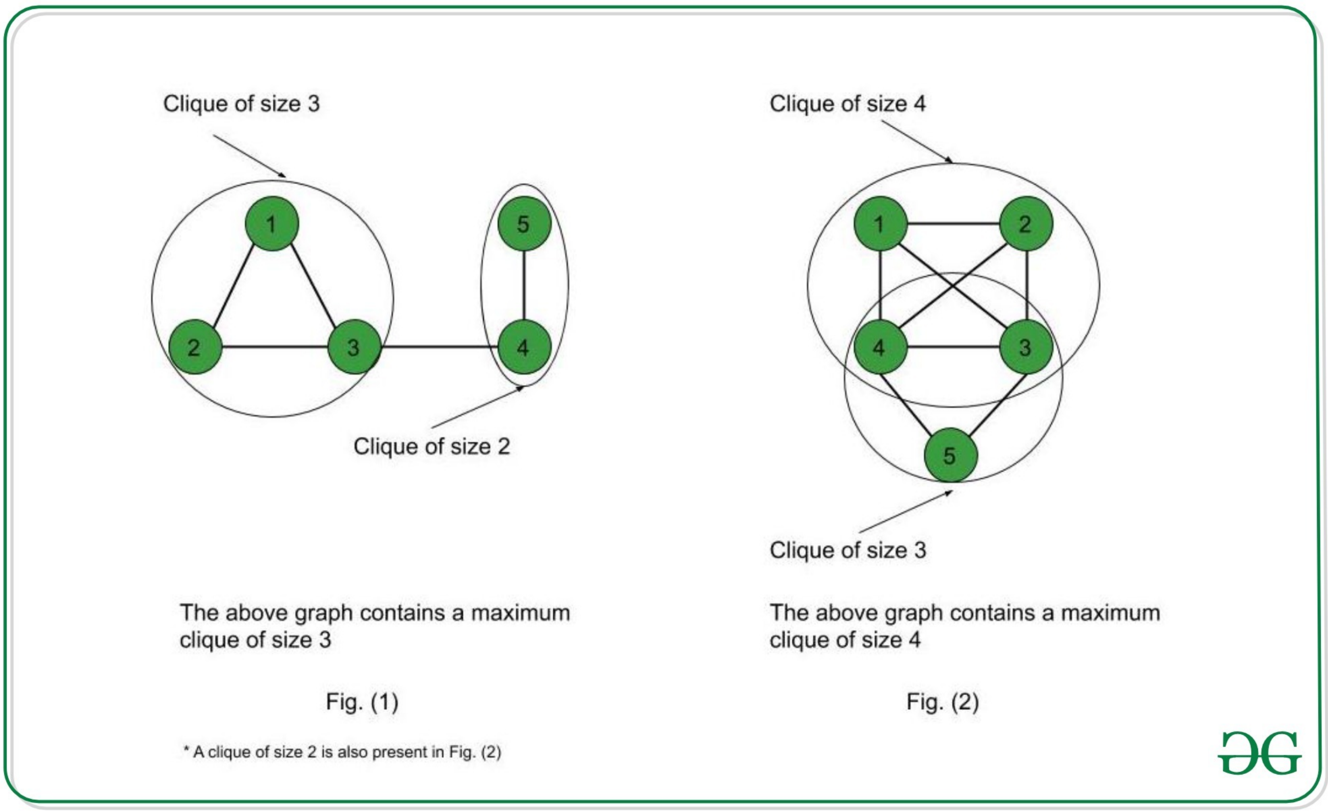
```
   c(x) = f(x) + h(x) where
   f(x) is the length of the path from root to x
        (the number of moves so far) and
   h(x) is the number of non-blank tiles not in
        their goal position (the number of mis-
        -placed tiles). There are at least h(x)
        moves to transform state x to a goal state
```

The below diagram shows the path followed by the above algorithm to reach the final configuration from the given initial configuration of the 8-Puzzle. Note that only nodes having the least value of cost function are expanded.



17 Show that the Clique problem is NP-Complete.

A clique is a subgraph of a graph such that all the vertices in this subgraph are connected with each other that is the subgraph is a complete graph. The Maximal Clique Problem is to find the maximum sized clique of a given graph G, that is a complete graph which is a subgraph of G and contains the maximum number of vertices. This is an optimization problem. Correspondingly, the Clique Decision Problem is to find if a clique of size k exists in the given graph or not.


Clique of size 3
Clique of size 2
Clique of size 4
Clique of size 3
The above graph contains a maximum clique of size 3
Fig. (1)
The above graph contains a maximum clique of size 4
Fig. (2)
* A clique of size 2 is also present in Fig. (2)

To prove that a problem is NP-Complete, we have to show that it belongs to both NP and NP-Hard Classes. (Since NP-Complete problems are NP-Hard problems which also belong to NP)

continue ....   Next PDF- ›