Algorithm for backtracking:- Recursive

```
{
for (each ak that belongs to T(a1, a2 -- ak-1)) do
{
  if (Bk (a1, a2 -- ak) = true) then // feasible sequence
  {
    if ((a1, a2 -- ak) is path to answer node then
        print (a[1], a[2] .. a[k]));
  if (k<n) then
    Backtrack (k+1); // find next set
}}
```

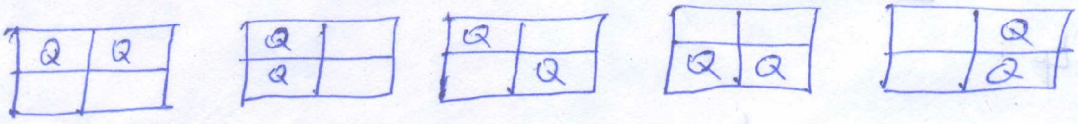
Non-recursive

```
{
k := 1;
while (k ≠ 0) do
{
if (any a[k] that belongs to T(a[1], a[2], --a[k-1] remaining untried]
AND (Bk (a[1], a[2] -- a[k] is true) then
{
  if ((a[1], a[2] --- a[k]) is a path to answer node) then
    write (a[1], a[2], --a[k]); // solution printed
// consider next element of the set
    k := k+1;
  else
    k := k-1; // backtrack to most recent value
}}
```

Consider $n \times n$ chessboard on which we have to place n queens so that no two queens attack each other by being in the same row or in the same column or on the same diagonal.

Examples:-

i) Consider 2×2 board

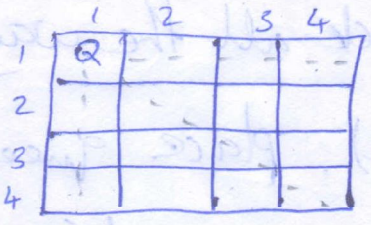


* 2-Queen's problem is not solvable,
* Because 2-queen's can be placed on 2×2 chessboard

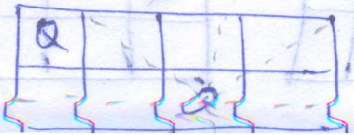
ii) Consider 4×4 board

* Let us take 4-queen's and 4×4 chessboard.

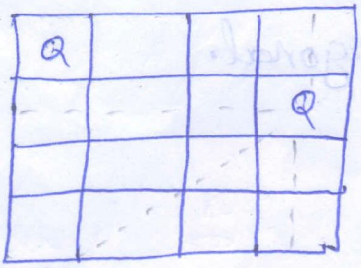
Step 1:
→ now we start with empty chessboard
→ place queen 1 in the 1st possible solution of its row.
→ i.e, on 1st row & 1st column.



Step 2:
→ Then place queen 2 after trying unsuccessful place - 1 (1,2), (2,1), (2,2) & at (2,3) i.e., 2nd row & 3rd column.



Then place 3rd queen cannot be placed in next column, as there is no acceptable position for queens. hence algorithm backtracks & places 2nd queen at (2,4) position.

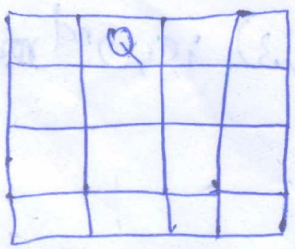


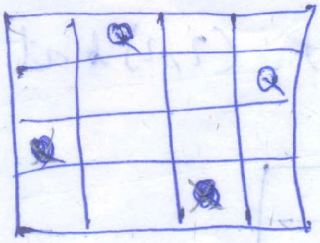
Step: 4

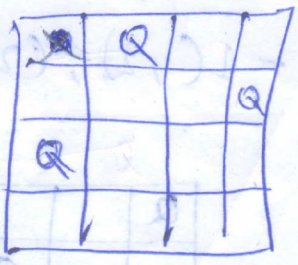
The place 3rd queen at (3,2) but it is again another dead end as next queen (4th queen) cannot be placed at permissible position.

Step: 5

Hence we need to backtrack all the way upto queen 1 & move it to (1,2). place queen 1 at (1,2), queen 2 at (2,4), queen 3 at (3,1) & queen 4 at (4,3)







Algorithm

```
N queen (k,n)
{
for i=1 to n do
{
  if (place (row, column i)) then // This function checks if
  {                                  two queens are on same
                                     diagonal or not.
   board [row] column  → no conflicts so place que
   if (row = n) then → dead end
   Print_board (n) → print the board configuration
   else → try next queen with next position
   queen (row+1, n)
  }}                  ↳ Row by row each queen is
                        placed by satisfying
                        constraints
place (row, column).
{ for i ← 1 to row -1 do
                  ↳ checking for column & diagonal conflicts
  {
   if (board [i] = column ) then
                  ↳ same column by 2 queens
   return 0
   else if (abs (board [i] - column) = abs (i-row))
                  ↳ This formula gives 2 queens are on same diagonal
   return 0                                          then
  }
return 1   → no conflicts hence queen can be placed.
```