BACKTRACKING

→ It is one of the problem solving / algorithm design strategy.

→ Backtracking is an algorithm technique to solve a pblm by an incremental way.

→ It was a recursive approach to solve the problem.

→ We can say that backtracking is used to find all the possible combinations to solve an optimization problem.

In backtracking, the problem can be categorized into 3 categories

1) decision problem :- Here we find whether there is any feasible solution.

2) Optimization problem :- Here we find whether there exists any best solution.

3) Enumeration problem :- Here we find all possible feasible solutions.

It incrementally builds candidates to the solutions, and abandons a candidate ("backtracks") as soon as it determines that the candidate cannot possibly be completed to a valid solution.

Applications of backtracking.

1) n-Queen's problems.
2) Graph colouring
3) 0/1 Knapsack problem
4) Hamiltonian cycles.
5) Sum of Subsets.

Working of backtracking with an example.

3 students → 2 boys.
→ 1 girl

4 3 chairs are there.

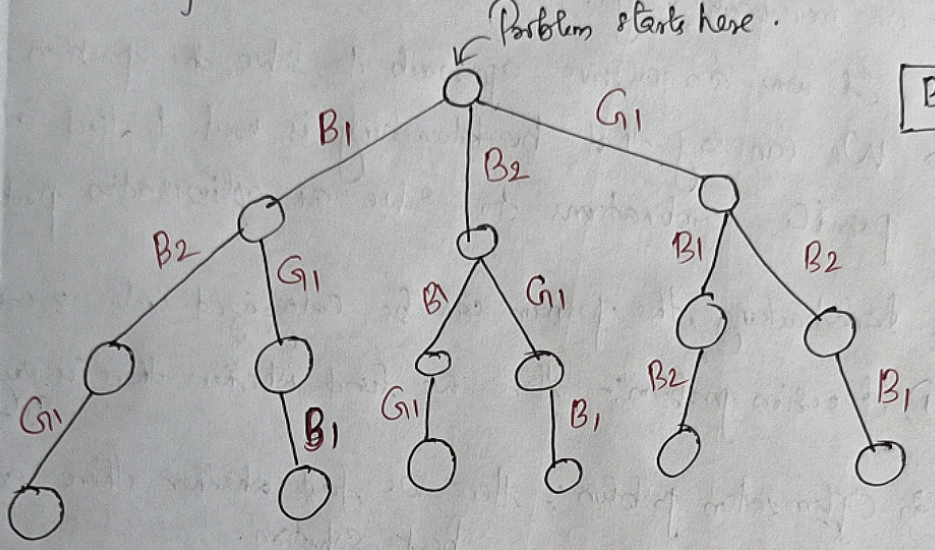
☆ We have to arrange them in these chairs.

pblm :- How many ways we can arrange them

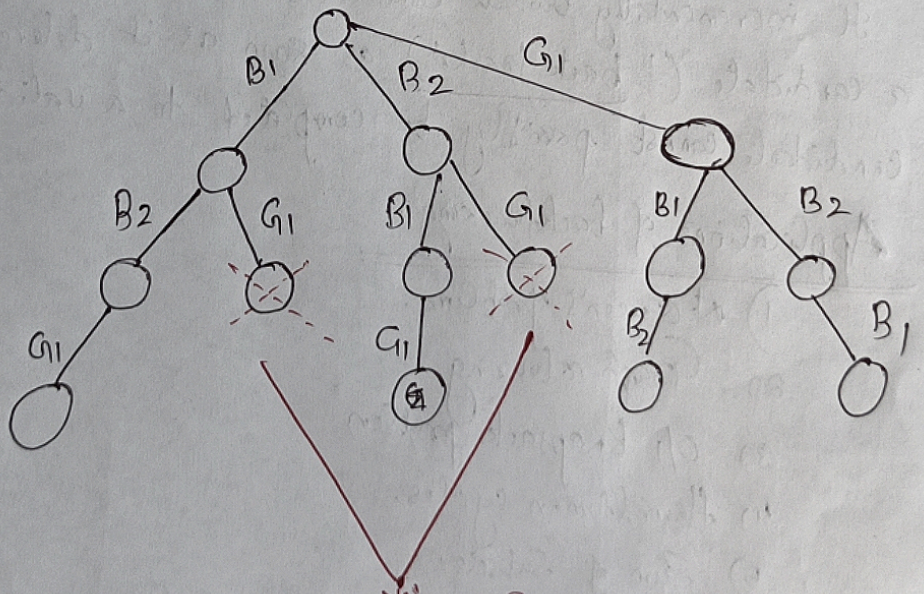
soln :- 3! ways = 3×2×1 = 6 ways.

We can represent the solution in the form of State Space tree.

{ A state space tree is the tree of the construction of the solution from partial solution, starting with the root ~~with root with~~ with no component solution }



Next, we are applying a constraint that the girl should not be in between the boys.



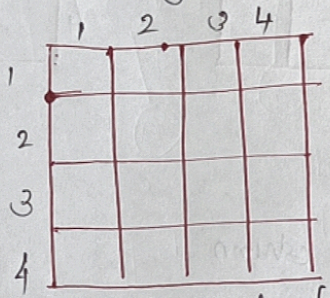
Bounding function. → is needed to kill some live nodes without actually expanding them.

N- Queens Problem

N-Queens problem is to place n-queens in such a manner or an NXN 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 solution exists for n=2 and n=3.

So first we will consider the 4 queens problem.

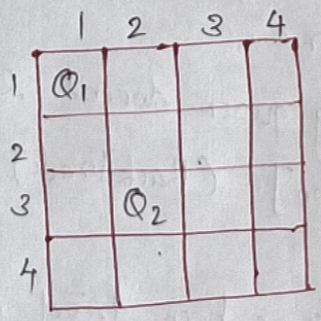
Given a 4X4 chessboard and number the rows and columns of the chessboard as 1 through 4.



4X4 chessboard.

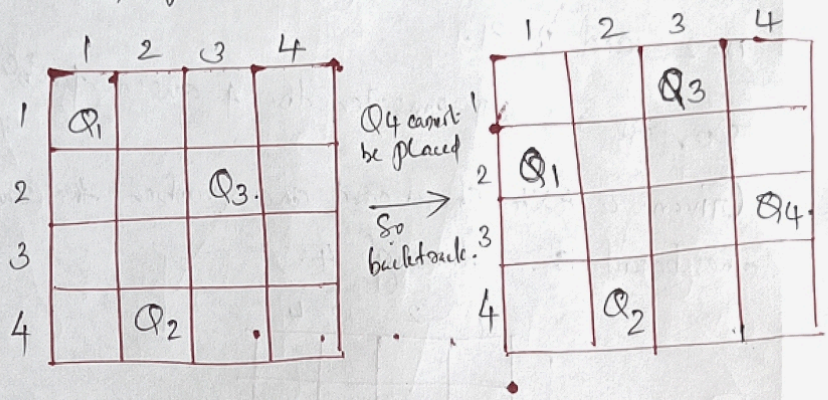
Since we have to place 4 queens such as Q_1, Q_2, Q_3 and Q_4 on the chessboard, such that no two queens attack each other. In such a condition each queen must be placed on a different row. ie, we put queen "i" on row "i".

Now, we place queen Q_1 in the very first acceptable position (1,1). (Whenever we place a ~~colu~~ Queen, increment the column and find a position for the next queen). Next we put queen Q_2 so that both these queens do not attack each other.



We find that if we place Q_2 in column 2 & row 3, ~~then~~ the dead end is encountered. ~~The first acceptable position is for Q_2~~. Then increment the column and when we try to place Q_3, then the dead end is encountered. So we

backtrack one step and replace the last placed queen Q_2. (by incrementing row). Then we obtain the position for Q_2 in the 4th row and now we can place Q_3 in 3rd column 3rd row. These steps are continued until all the queen are placed in the safe position.



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 for every tried row.
   a) If the queen can be placed safely in this row
      then mark this [row, column] as part of the solution &
      recursively check if placing queen here leads to a solution.
   b) If placing the queen in [row, column] leads to a solution
      then return true.
   c) If placing queen doesn't lead to a solution then unmark
      this [row, column] (Backtrack) and go to step (a) to try
      other rows.
4) If all rows have been tried and nothing worked,
   return false to trigger backtracking.