The Able-Baker Call center Problem.

Simulate the following scenario assuming that the time between the call is random. There are two technical Support people Able & Baker. Able is more experienced & can provide service faster than Baker. Times are usually a continuous measure, But this time based example is made discrete for ease of explanation. The simulation proceeds in a complex manner because of two servers. When both are free Able gets the call. Assume service distribution other than given on the class PPTs.

// The Simulation ~~proceeds~~ may proceeds in accordance with the following set of steps.

Step1: For caller K, generate an arrival time A_K.

Add it to the previous arrival time T_{k-1} to get the arrival time of caller K as $T_k = T_{k-1} + A_K$.

Step 2: If Able is idle, caller K begins service with Able at the current time T_{now}.

Able's service completion time $T_{fin.A}$ is given by, —

$$T_{fin.A} = T_{now} + T_{serv.A}$$

Here $T_{serv.A}$ is the service time generated from Able's service time distribution.

caller K's time in system, T_{sys}, is given by,—

$$T_{sys} = T_{fin.A} - T_k.$$

Because Able was idle, caller K's delay, T_{wait} can be given by - $T_{wait} = 0$.

If Able is busy, But Baker is idle, caller K begins service with Baker at the current time T_{now}. Baker's service completion time $T_{fin.B}$ is given by

$$T_{fin.B} = T_{now} + T_{serv.B}$$

Here $T_{serv.B}$ is the service time generated from Baker's service time distribution.

Caller K's time in system T_{sys} is given by

$$T_{sys} = T_{finB} - T_k.$$

Because Baker was idle caller K's delay T_{wait} is given by - $T_{wait} = 0$.

Step 3 : If Able & Baker are both busy then calculate the time at which the first one become available, as follows:

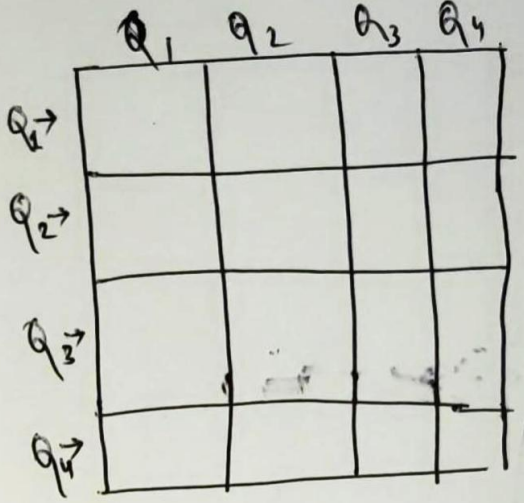
$$T_{beg} = \min(T_{fin.A}, T_{fin.B}).$$

caller K begin service at T_{beg}. when service for caller K begins set $T_{now} = T_{beg}$.

⇒ Then compute $T_{fin.A}$ or $T_{fin.B}$ as in Step2.

N-Queens Problems

We have to place the queens such that they are not under attack i.e. they can't be on the
- → Same row
- → Same column
- → Same diagonal.



We can solve the problem using Backtracking.

Therefore, we have 16 cell & 4 queen.
(As for simplicity we have assumed 4×4 board).

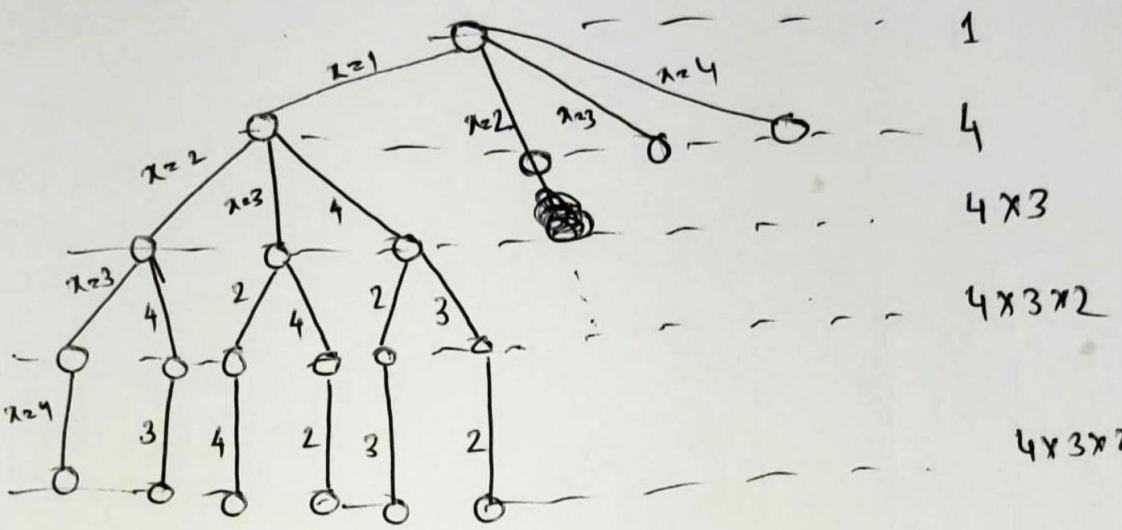
Thus, total no. possibility of placing a queen → $^{16}C_4$.
$= 1820$.

To reduce the size of the problem we will consider that
- → each queen is assigned to a row/column.
- → (no two queen can be on the same row)
- → (we will not place two queen on the same row).

⇒ Let's say we initially allowing the diagonal placement. & lets walk through the solution space called "State-Space Tree".

(In the class I've discussed in detail that how to generate the state space tree).

(State Space Tree for N Queen):



Total Nodes = $1 + 4 * 4\times3 + 4\times3\times2 + 4\times3\times2\times1$.

$$= 1 + \sum_{i=0}^{3}\left[\prod_{j=0}^{i}(4-j)\right]$$

⇒ No. of Nodes when we have avoided same row & col^n

for a N no. queen, Maximum no. of nodes can be given as, –

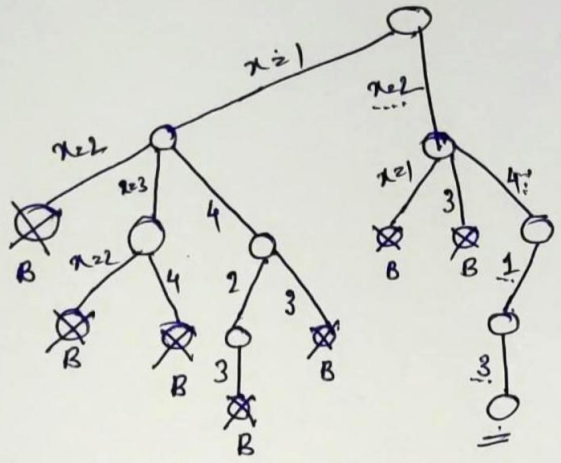
$$1 + \sum_{i=0}^{3}\left[\prod_{j=0}^{i}(N-j)\right]$$

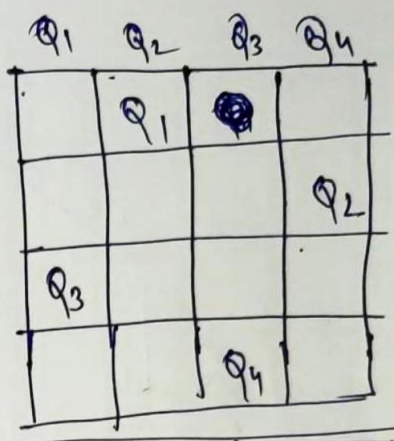
⇒ we will reduce the problem further by applying the bounding function.

⇒ Bounding function prevents/restricts the from placing the queen diagonally.

(In the class I've discussed the formation of tree the final tree is ~~#~~ will be like following).

State Space tree with bounding function





S_1 =	2	4	1	3

S_2 =	3	1	4	2

Ans :- 2,4,1,3.

the one answer is - 2,4,1,3
& another one is 3,1,4,2.

Algorithm

1) Start in the left most column
2) if all queens are placed
   return true
3). Try all rows in the current column
   Do the 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 any solution.
   b) If placing the queen in [row, column] leads to a solution
      → then return True.

c) If placing queen doesn't leads to a solution
→ then unmark this [row, column] (Back track) & go to step(a) to try other rows.

3) If all rows have been tried & nothing worked,
→ return false to trigger back tracking.
