CS-218 Data Structures

Week 2 | Lecture 2

Brute force Approach

- ***Brute Force*** algorithms are straightforward methods of solving a problem that rely on sheer computing power and trying every possibility

- **Example:** You forgot your password
  - **Brute force solution:** *Generate all possible combinations and use the correct one to sign in!*

Backtracking

- Backtracking is like a refined brute force

- At each step, we eliminate choices that are obviously not possible and proceed to recursively check only those that have potential

Backtracking

- When solving problems with recursion, we divide it into sub-problems
  e.g. recursive calls in Fibonnaci, factorial

- What if we don't know what the correct sub-problem is?

- Using Backtracking, we can explore sub-problems until we find the optimal one

Example

- Pseudo-code for finding factorial:

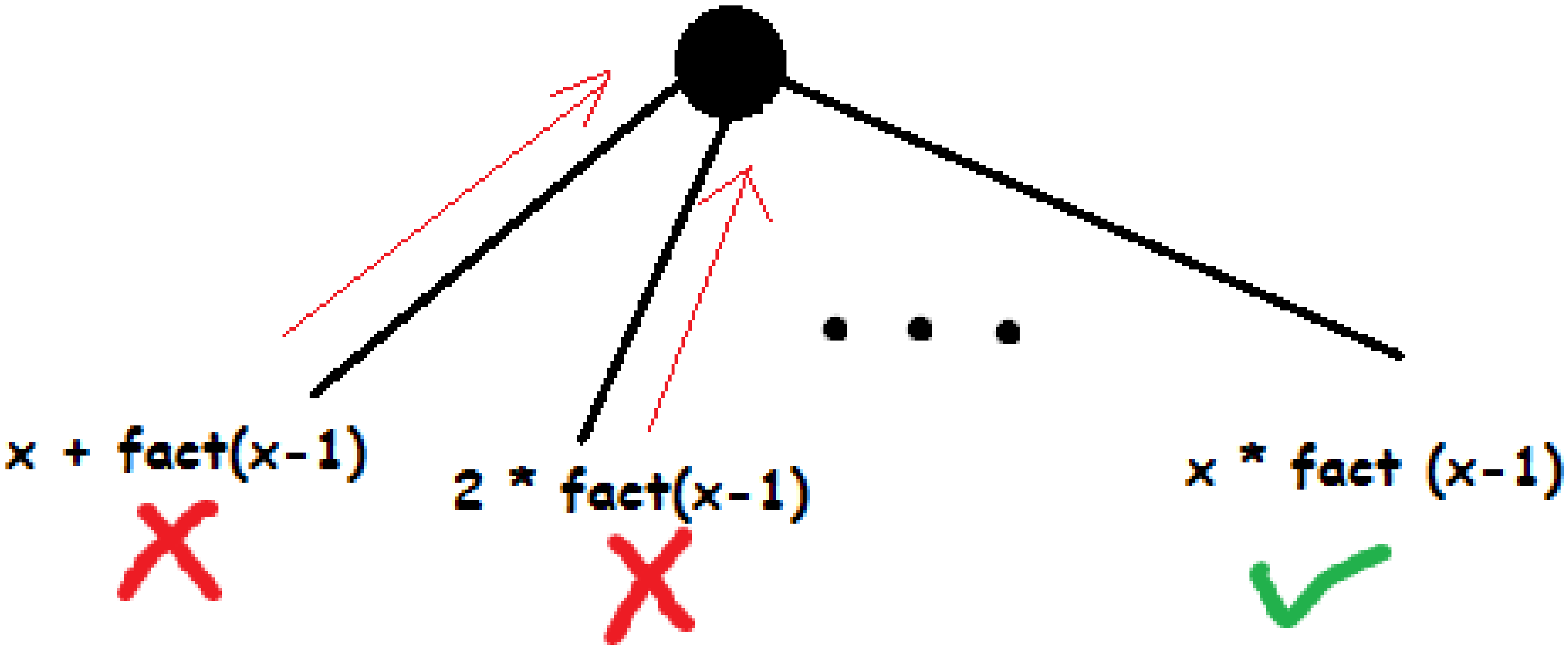
```
function factorial(x)
        if x is 0                 // base case
                return 1

        else return x * factorial(x-1)
        // break into smaller problem(s)
```

Example

- What if we don’t know the recursive case for factorial i.e. **x * factorial(x-1)**

- We can recursively try out different sub-solutions and backtrack if it does not lead us to the correct overall solution

Example



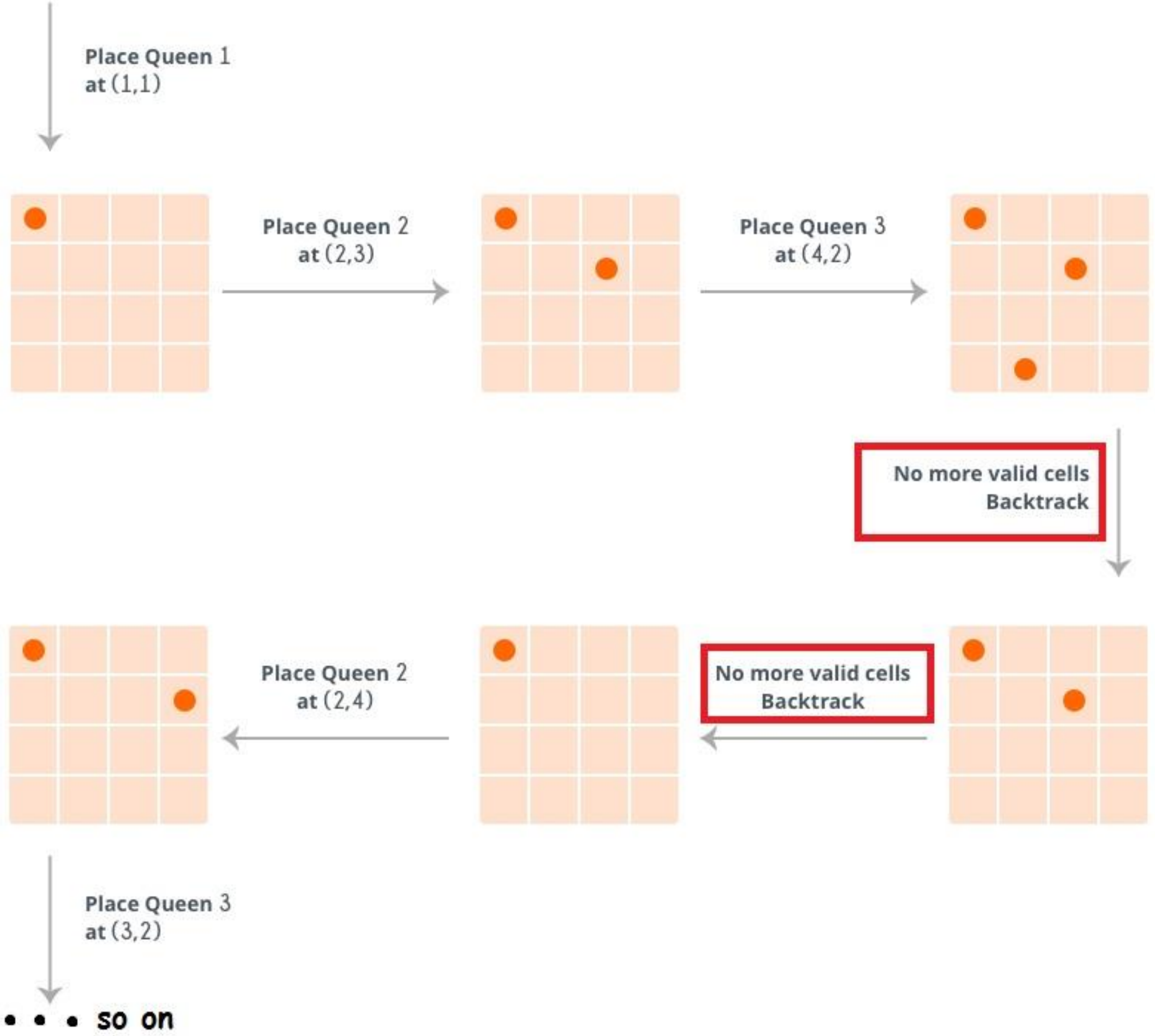
Example

- **N-Queens Problem:**
  Given a grid having N×N cells, we need to place N queens in such a way that no queen is attacked by any other queen. A queen can attack horizontally, vertically and diagonally

- We continue placing queens as long as:
  - The number of unattacked cells is not 0
  - The number of queens to be placed is not 0

Example

Here's how it works for $N = 4$.



Place Queen 1 at (1,1)

Place Queen 2 at (2,3)

Place Queen 3 at (4,2)

No more valid cells Backtrack

No more valid cells Backtrack

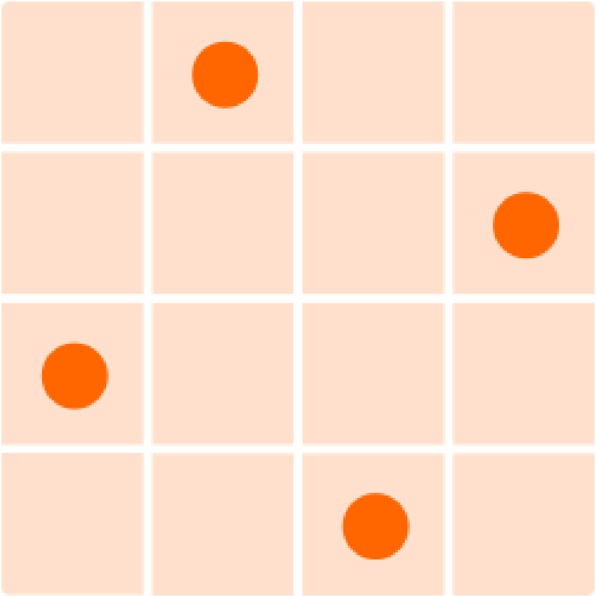
Place Queen 2 at (2,4)

Place Queen 3 at (3,2)

• • • so on

Example

- **Goal state:**



Pruning

- Eliminating choices that do not lead us to solution

- Much of what we did in the previous examples