Algorithms

Backtrack Homework 1

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Problem #1: LeetCode 797 - All Paths From Source to Target

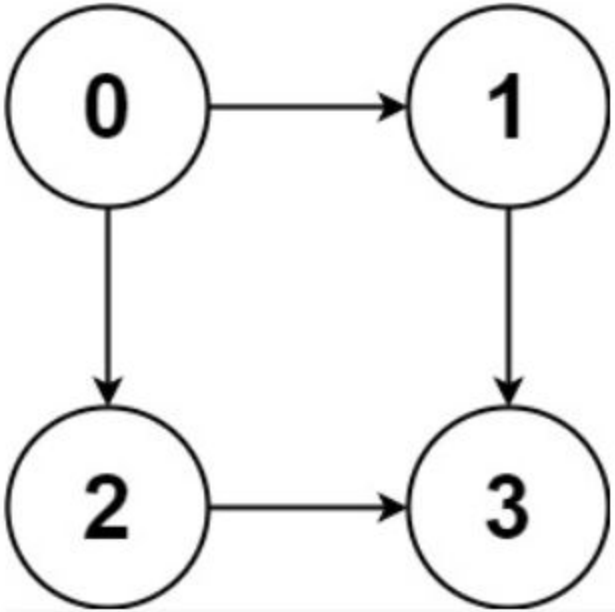
Given a directed acyclic graph (**DAG**) of `n` nodes labeled from `0` to `n - 1`, find all possible paths from node `0` to node `n - 1` and return them in **any order**.

The graph is given as follows: `graph[i]` is a list of all nodes you can visit from node `i` (i.e., there is a directed edge from node `i` to node `graph[i][j]`).

Constraints:

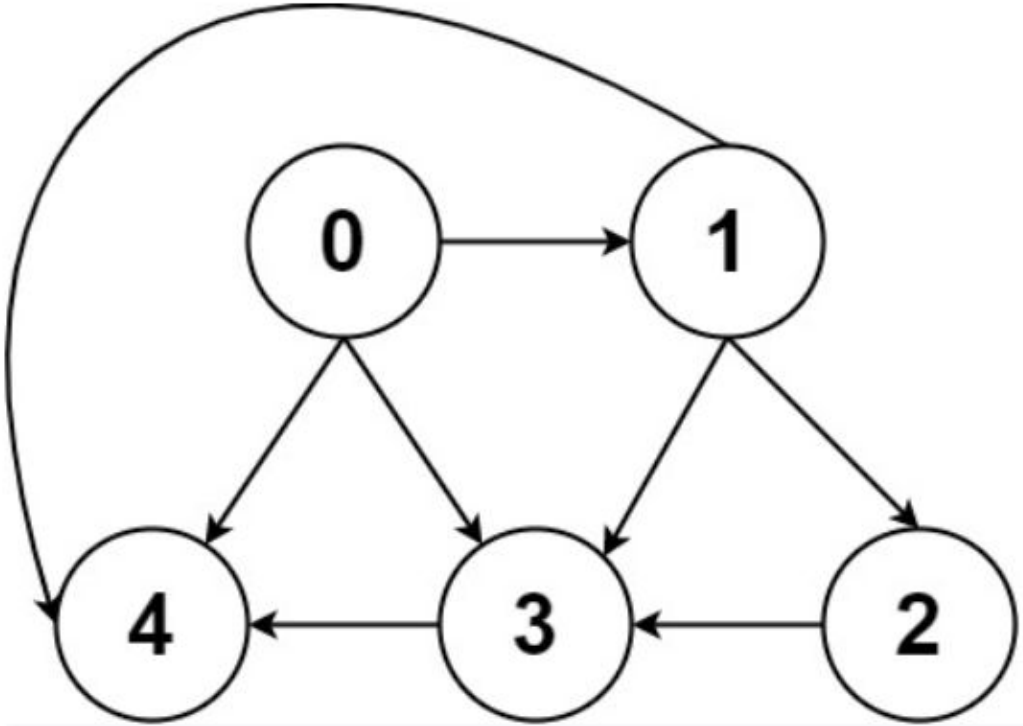
- `n == graph.length`
- `2 <= n <= 15`
- `0 <= graph[i][j] < n`
- `graph[i][j] != i` (i.e., there will be no self-loops).
- All the elements of `graph[i]` are **unique**.
- The input graph is **guaranteed** to be a **DAG**.

Example 1:




```
Input: graph = [[1,2],[3],[3],[]]
Output: [[0,1,3],[0,2,3]]
Explanation: There are two paths: 0 -> 1 -> 3 and 0 -> 2 -> 3.
```

Example 2:




```
Input: graph = [[4,3,1],[3,2,4],[3],[4],[]]
Output: [[0,4],[0,3,4],[0,1,3,4],[0,1,2,3,4],[0,1,4]]
```

Problem #2: LeetCode 51 - N-Queens

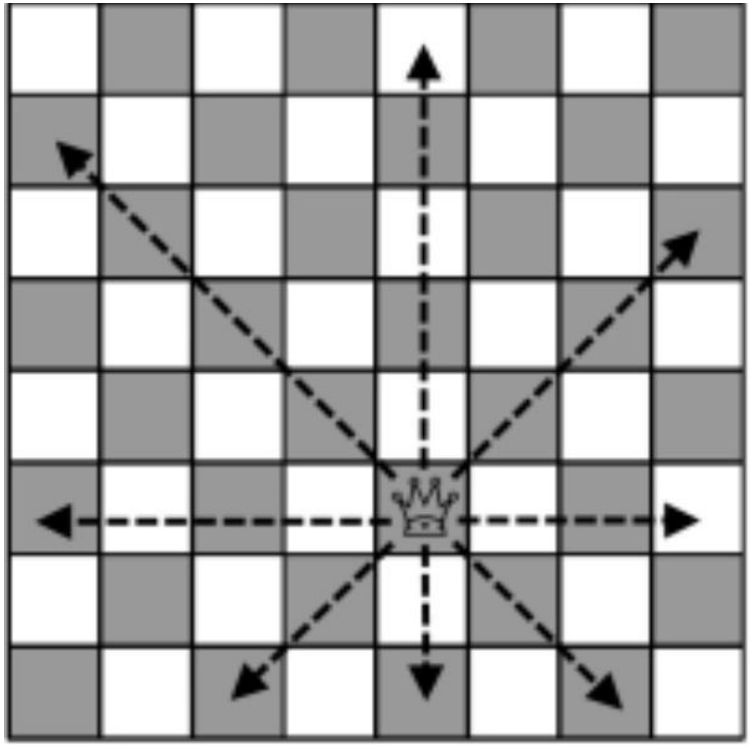
The **n-queens** puzzle is the problem of placing `n` queens on an `n x n` chessboard such that no two queens attack each other.

Given an integer `n`, return *all distinct solutions to the **n-queens puzzle***. You may return the answer in **any order**.

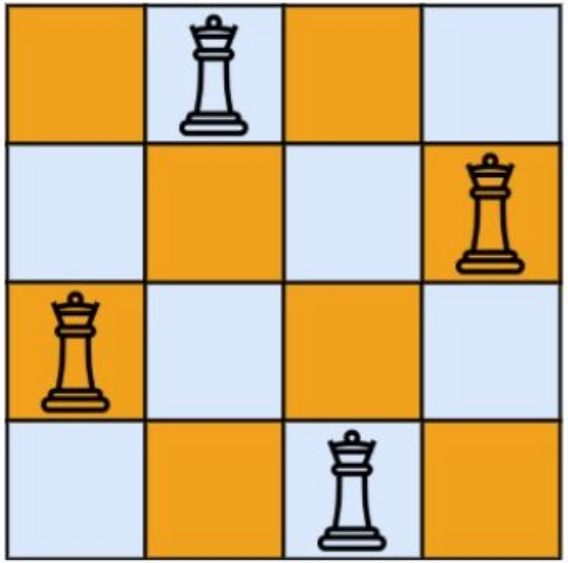
Each solution contains a distinct board configuration of the n-queens' placement, where `'Q'` and `'.'` both indicate a queen and an empty space, respectively.

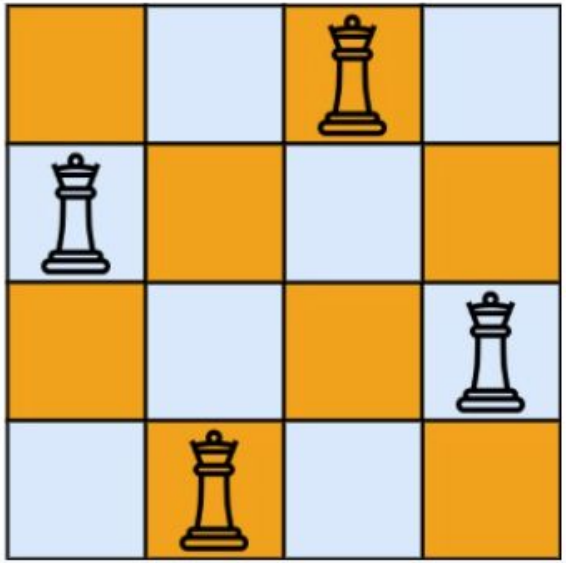
1 <= n <= 9

Queen Move



- In chess, a queen can move and attack any cell in these directions
  - Its row
  - Its col
  - Its normal diagonal
  - Its anti-diagonal





Input: n = 4
Output: [[".Q..","...Q","Q...","..Q."],["..Q.","Q...","...Q",".Q.."]]
Explanation: There exist two distinct solutions to the 4-queens puzzle as shown above

Example 2:

Input: n = 1
Output: [["Q"]]

Coding tips

- Say we code **backtrack(row)**
  - Where we try to put a queen in current row and move to next row
  - Then after the end, we put all queen
- To put a queen, it must be in a place no other queen (from what we put so far) can attack it
  - So all directions are clear!
- There are N rows and columns in a grid
- There are 2N-1 normal and anti-diagonals
  - Can you find trivial formula for them?
- So with boolean array (or sets),
  we can trivially mark visit status

Coding tips

Every square has value (row + col).
Anti-diagonals share the same values

	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	8
2	2	3	4	5	6	7	8	9
3	3	4	5	6	7	8	9	10
4	4	5	6	7	8	9	10	11
5	5	6	7	8	9	10	11	12
6	6	7	8	9	10	11	12	13
7	7	8	9	10	11	12	13	14

Every square has value (row - col). Diagonals
share the same values

	0	1	2	3	4	5	6	7
0	0	-1	-2	-3	-4	-5	-6	-7
1	1	0	-1	-2	-3	-4	-5	-6
2	2	1	0	-1	-2	-3	-4	-5
3	3	2	1	0	-1	-2	-3	-4
4	4	3	2	1	0	-1	-2	-3
5	5	4	3	2	1	0	-1	-2
6	6	5	4	3	2	1	0	-1
7	7	6	5	4	3	2	1	0

- row+col
- row-col
  - -ve

Problem #3: LeetCode 37 - Sudoku Solver

Write a program to solve a Sudoku puzzle by filling the empty cells.

A sudoku solution must satisfy **all of the following rules**:

1. Each of the digits `1-9` must occur exactly once in each row.
2. Each of the digits `1-9` must occur exactly once in each column.
3. Each of the digits `1-9` must occur exactly once in each of the 9 `3x3` sub-boxes of the grid.

The `'.'` character indicates empty cells.

Constraints:

- `board.length == 9`
- `board[i].length == 9`
- `board[i][j]` is a digit or `'.'`.
- It is **guaranteed** that the input board has only one solution.

Example

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Problem #4: LeetCode 46 - Permutations

Given an array `nums` of distinct integers, return *all the possible permutations*. You can return the answer in **any order**.

Constraints:

- `1 <= nums.length <= 6`
- `-10 <= nums[i] <= 10`
- All the integers of `nums` are **unique**.

Example 1:

```
Input: nums = [1,2,3]
Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

Example 2:

```
Input: nums = [0,1]
Output: [[0,1],[1,0]]
```

Example 3:

```
Input: nums = [1]
Output: [[1]]
```

Hint

- Let **permute_bt**(idx): be a function that generates all permutations starting from this idx (and previous values are fixed)
  - So for list [**2, 1**, 5, 9]
  - permute_bt(2) is generating all permutations from idx = 2
    - which are: [**2, 1**, 5, 9] and [**2, 1**, 9, 5]
- Think recursively: what not how

Problem #5: LeetCode 47 - Permutations II

Given a collection of numbers, `nums`, that might contain duplicates, return *all possible unique permutations* ***in any order***.

- 1 <= nums.length <= 8
- -10 <= nums[i] <= 10
- Hint: use **frequency** array

Example 1:

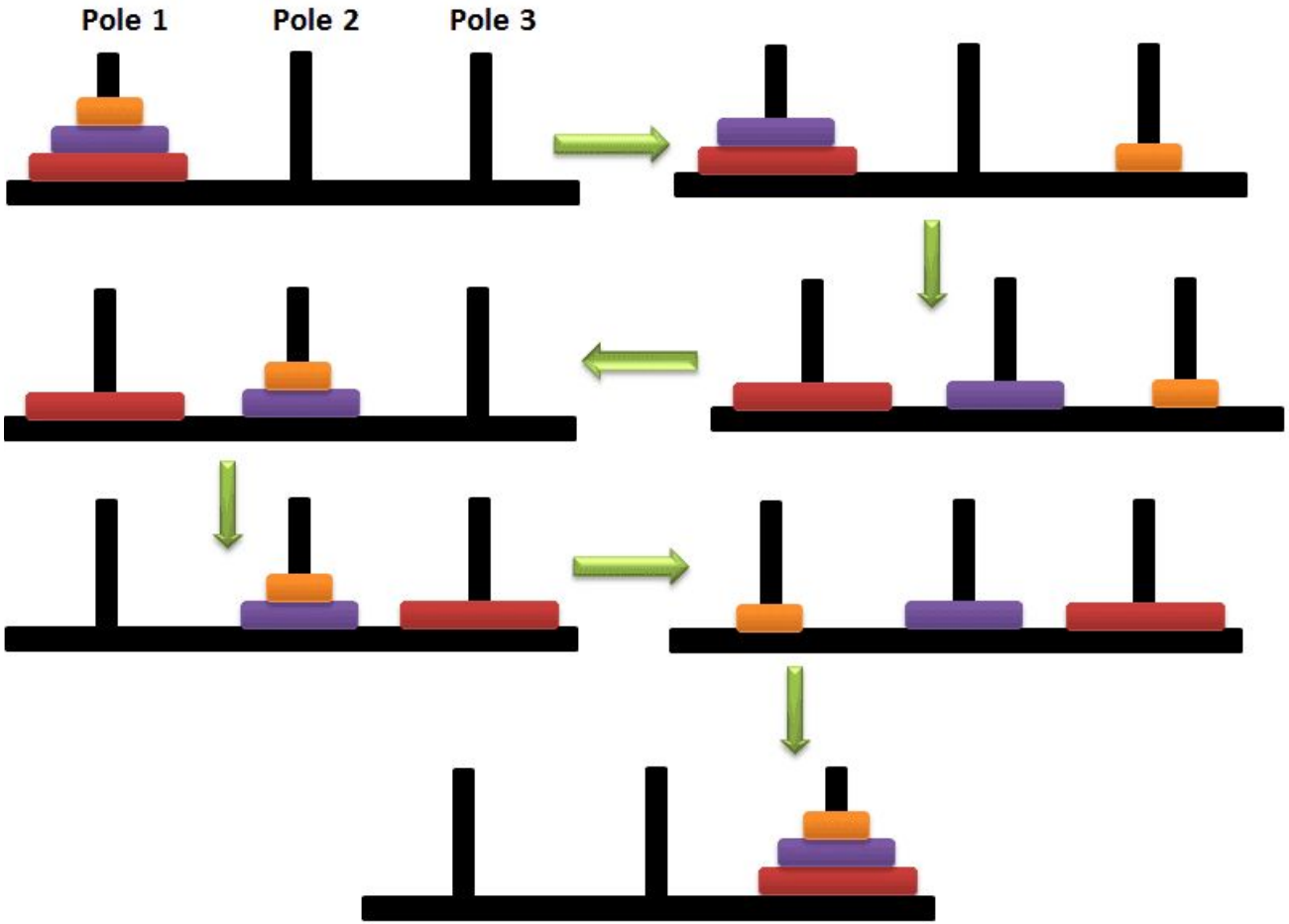
```
Input: nums = [1,1,2]
Output:
[[1,1,2],
 [1,2,1],
 [2,1,1]]
```

Example 2:

```
Input: nums = [1,2,3]
Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

Problem #6: Tower of Hanoi

- This is a **popular** math problem solved by recursion (not backtracking)
- We are given 3 rods (poles) and N disks. We want to move disks from rod A (source) to rod B (destination). We can use rod C (auxiliary)
  - Disks have different size
- You must follow these rules:
  - Only one disk can be moved at a time.
  - Move from top of a stack to another top of stack
  - No disk may be placed on top of a smaller disk.
- Your tasks
  - Simulate the process (with the minimum number of steps)
  - Given N, provide a simple formula for the total number of steps



N = 3

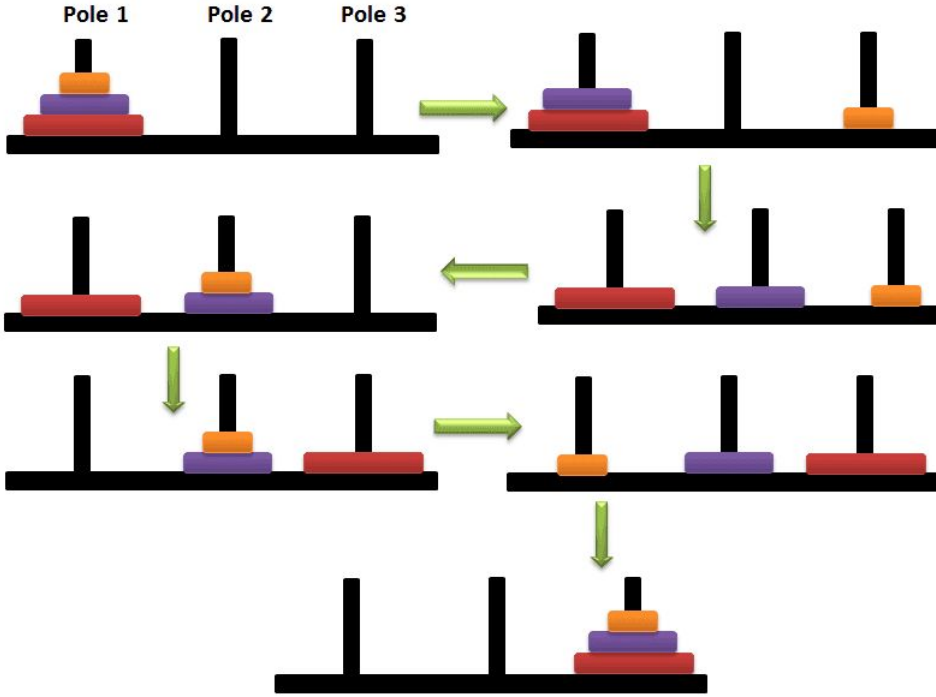
src

Your task

- Define: **towerOfHanoi**(n, from_rod, to_rod, aux_rod)
  - Initially: from_rod = ‘A’, to_rod = ‘B’, aux_rod = ‘C’
    - Names for the 3 rods
- The function should print all the steps to move N disks from from_rod (A) to to_rod B
- Hint: Do tracing to N = 2 and N = 3
  - But think what not how. Code is 5 lines
- Note: internet has many **gif** visualizations.
- For n = 2, print:
  - Move disk 1 from rod A to rod B
  - Move disk 2 from rod A to rod C
  - Move disk 1 from rod B to rod C

For n = 3

- Move disk 1 from rod A to rod C
- Move disk 2 from rod A to rod B
- Move disk 1 from rod C to rod B
- Move disk 3 from rod A to rod C
- Move disk 1 from rod B to rod A
- Move disk 2 from rod B to rod C
- Move disk 1 from rod A to rod C

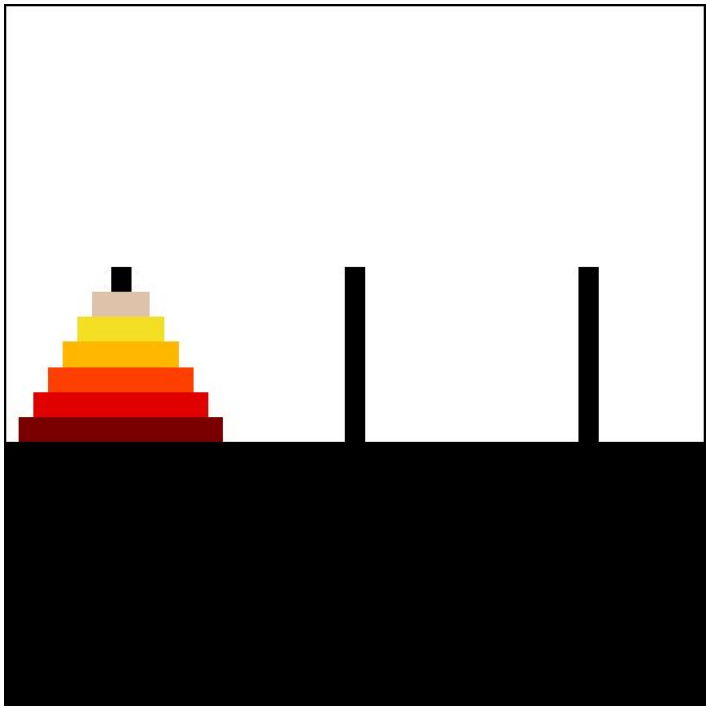


For n = 4

- Move disk 1 from rod A to rod B
- Move disk 2 from rod A to rod C
- Move disk 1 from rod B to rod C
- Move disk 3 from rod A to rod B
- Move disk 1 from rod C to rod A
- Move disk 2 from rod C to rod B
- Move disk 1 from rod A to rod B
- Move disk 4 from rod A to rod C
- Move disk 1 from rod B to rod C
- Move disk 2 from rod B to rod A
- Move disk 1 from rod C to rod A
- Move disk 3 from rod B to rod C
- Move disk 1 from rod A to rod B
- Move disk 2 from rod A to rod C
- Move disk 1 from rod B to rod C

- Given N, find a formula for the steps
  - N = 2 ⇒ 3
  - N = 3 ⇒ 7
  - N = 4 ⇒ 15
  - N = 5 ⇒ 31
- Can you justify mathematically the formula?

N = 4: Animation



src

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."