fit@hcmus

PROGRAMMING TECHNIQUES

Recursion & Backtracking

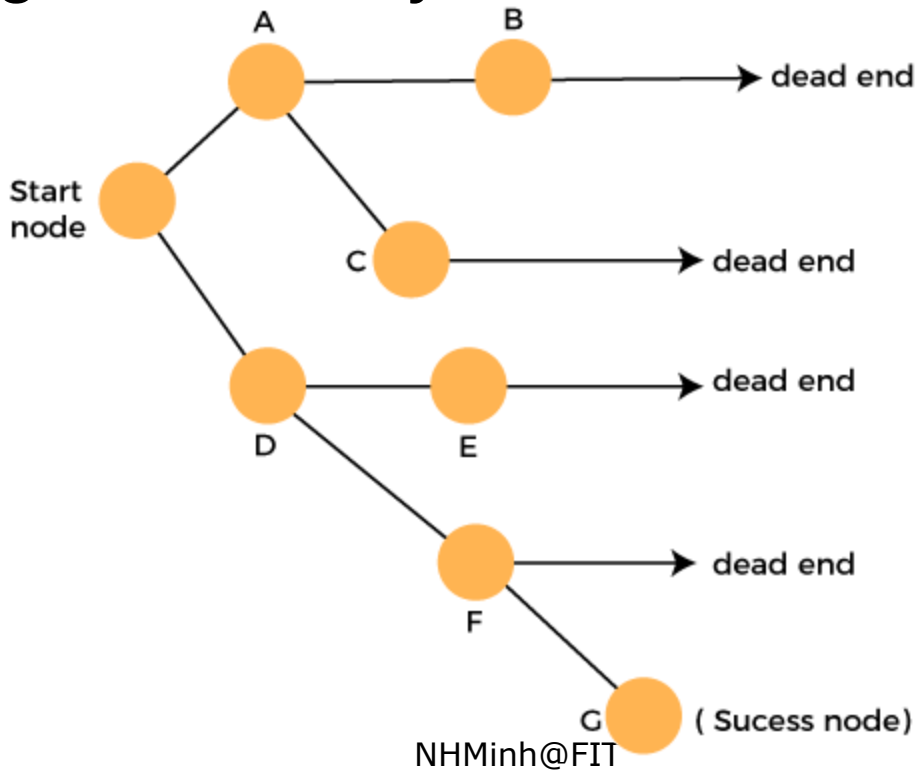
Nguyễn Hải Minh - 04/2024

Outline

- What is Backtracking?
- How does Backtracking work?
- Walk through examples:
  - The N-queen problem
  - All Subsets of an array
- Applications

What is Backtracking?

- Backtracking is a problem-solving algorithmic technique that involves finding a solution incrementally by trying different options and undoing them if they lead to a dead end.



How does Backtracking work?

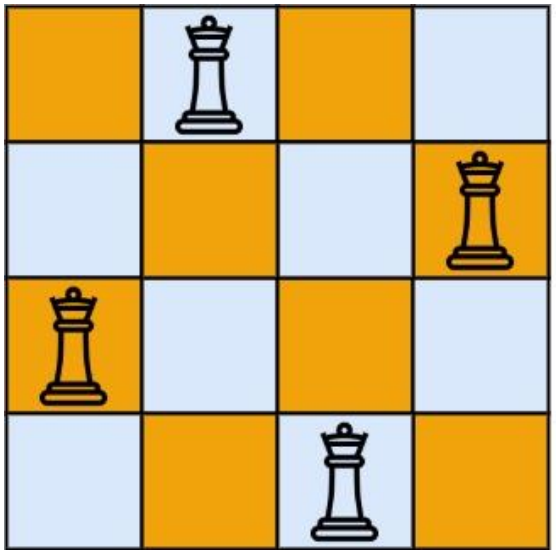
1. **Choose:** make a choice of an initial solution.
2. **Explore:** recursively explore all possible choices of the current solution.
3. **Backtrack:** if a choice doesn't work, undo it and backtrack to the previous choice.

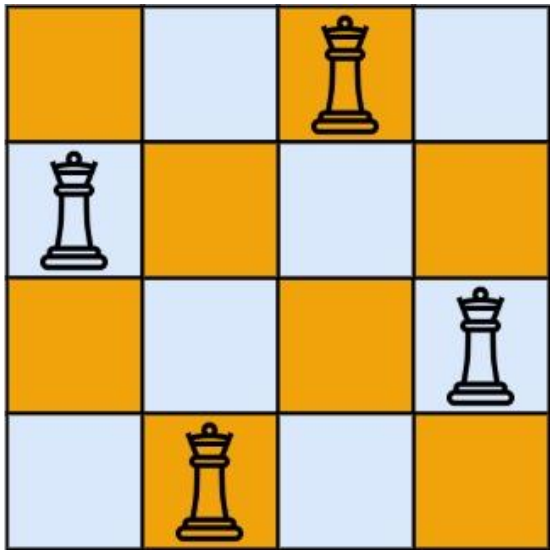
→ *Return fail if no choice remains*

EXAMPLE 1:
THE N-QUEEN PROBLEM

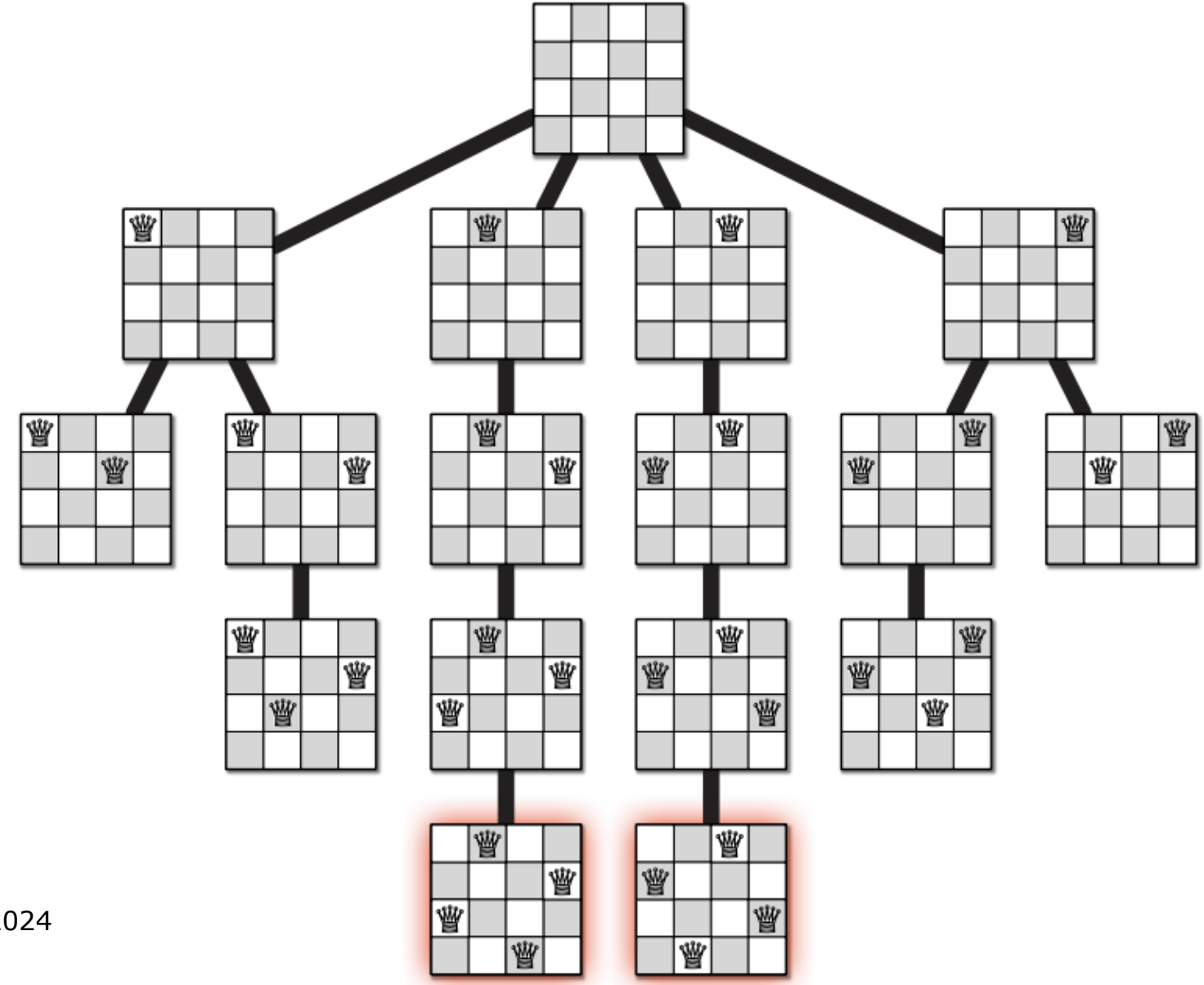
The N-Queen Problem

- Place N queens on an N x N chess board so that no pair of queens attacking each other.
- Example with N=4:



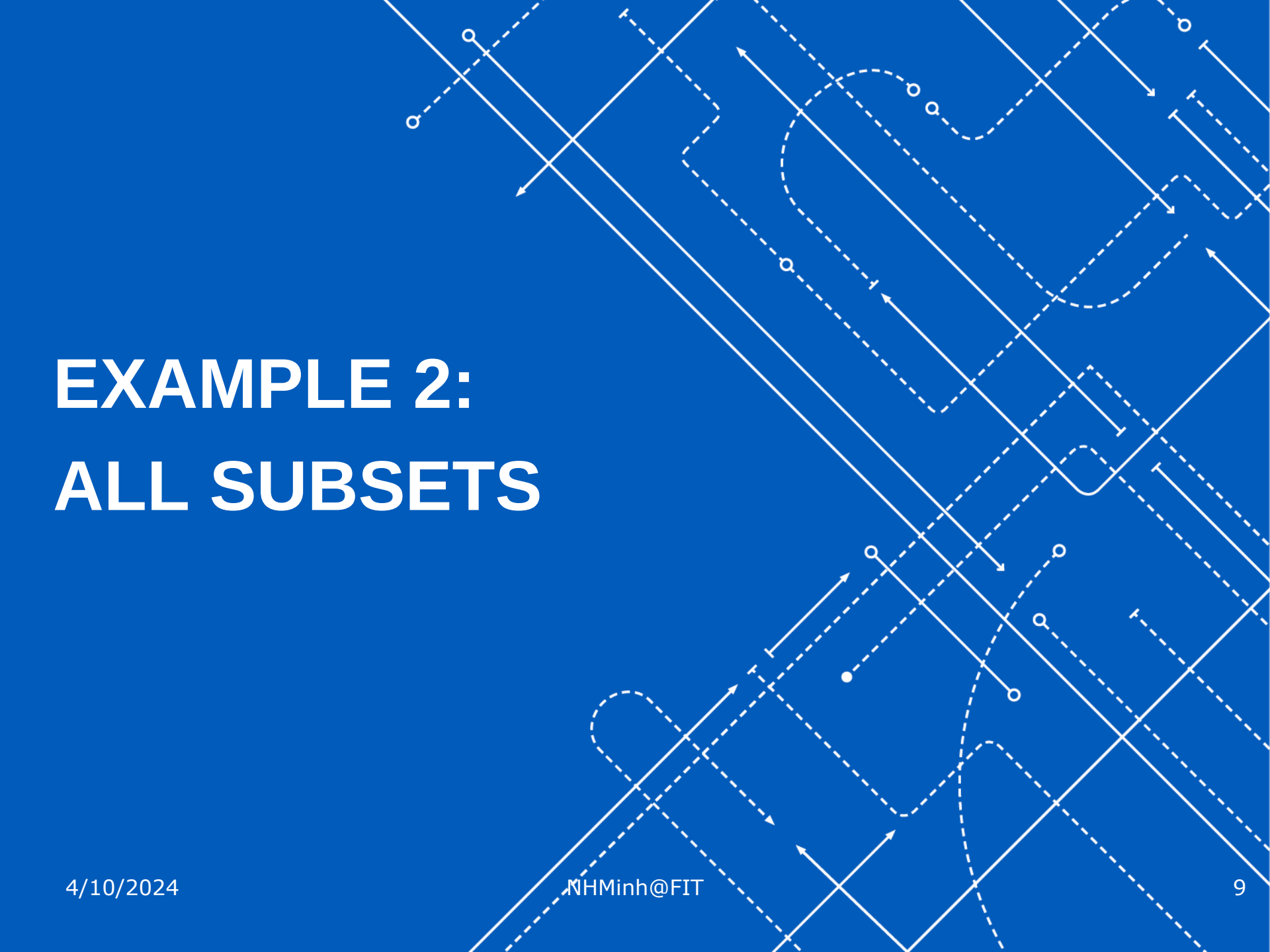


N-Queen – Backtracking Tree



N-Queen Backtracking

```
bool PutQueenAtCol(int board[N][N], int col)
{
   if (col >= N)
      return true;
   for (int i = 0; i < N; i++) {
      if (IsSafe(board, i, col))//is it safe to put the queen here?
      {
         board[i][col] = 1;
         //can remaining queens be put to other cols?
         if (PutQueenAtCol(board, col + 1))
            return true;
         //if NOT, backtrack
         board[i][col] = 0;
      }
   }
   return false;
}
```



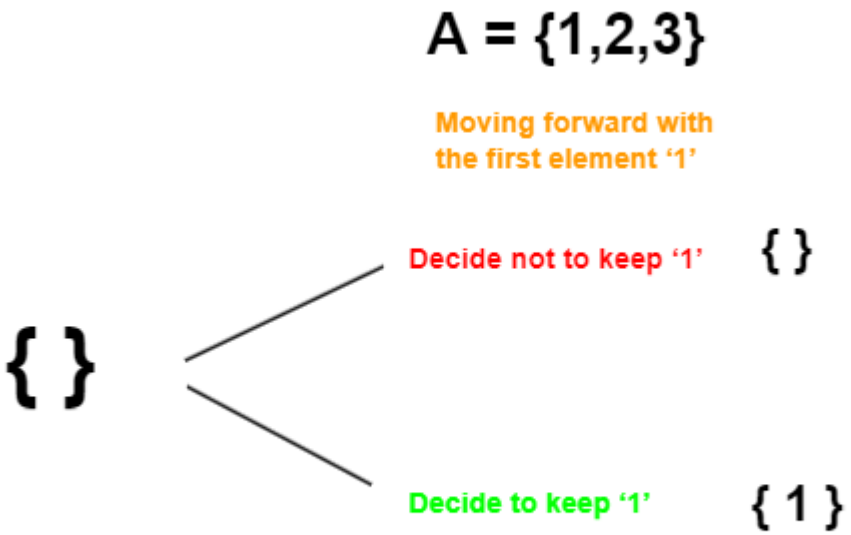
EXAMPLE 2: ALL SUBSETS

All Subsets of an Array

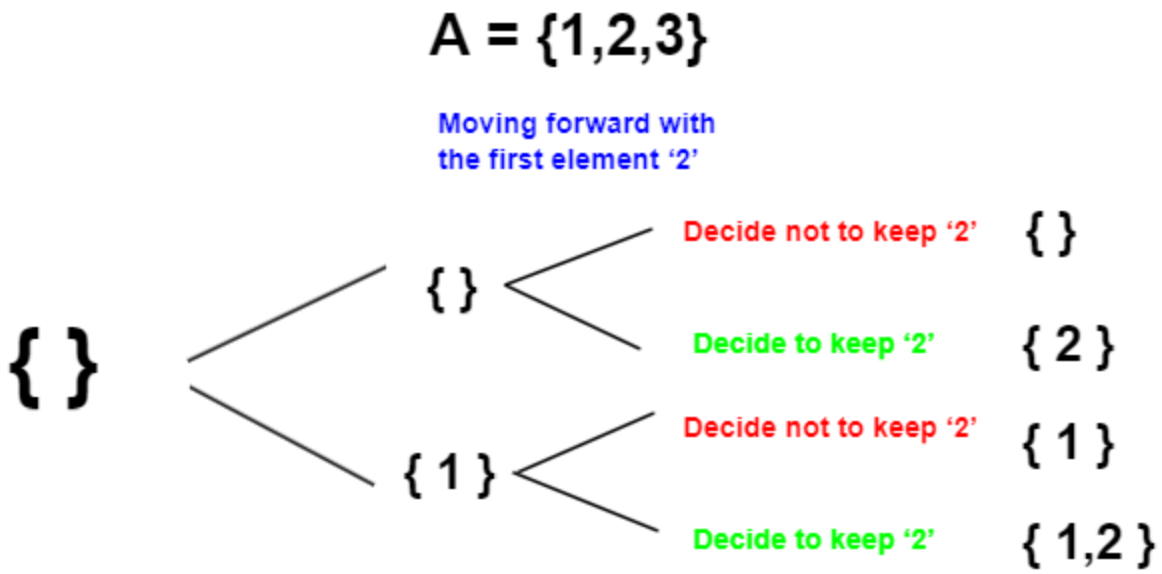
- Given an array **A** of **N** integers, print all subsets of A
- Example:
  - Input: **A = {1, 2, 3}**
  - Output: **{}, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}**
- How many subsets are there?
  - → 2^N
  - Proof?

All Subsets – Backtracking

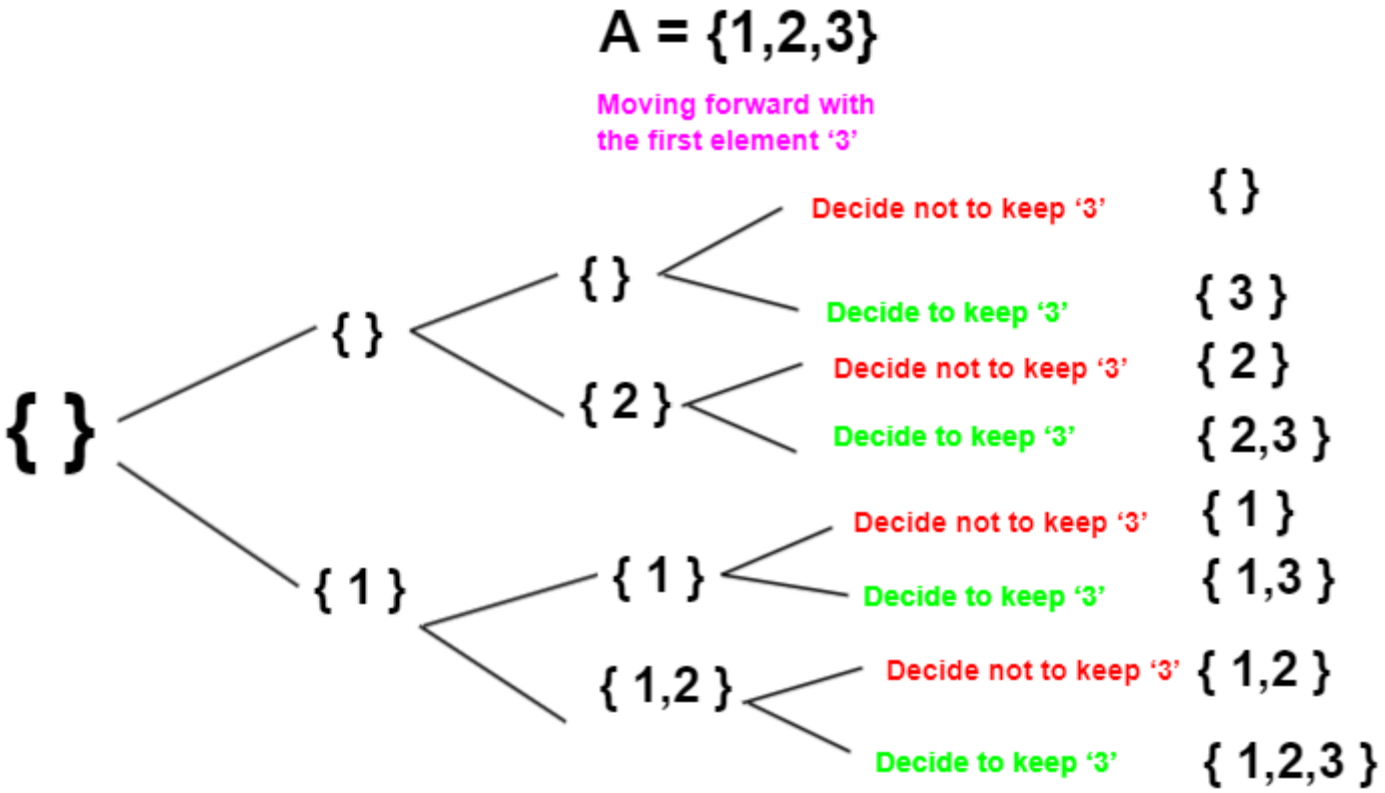
- For each element of the array, we have 2 choices:
  - Not include it to the subset
  - Include it to the subset



All Subsets – Backtracking



All Subsets – Backtracking



All Subsets – Backtracking

```
void Subset(int arr[], int mask[], int n, int k)
{
  //We have iterate through all elements in arr
  if (k == n) {
    for (int i = 0; i < n; i++)
      if (mask[i] == 1)
        cout << arr[i] << " "; //output the choices
    cout << endl;
    return;
  }
  //Else: we have 2 choices:
  mask[k] = 0; //not include arr[k] to the subset
  Subset(arr, mask, n, k + 1);

  mask[k] = 1; //backtrack: include arr[k] to the subset
  Subset(arr, mask, n, k + 1);
}
```

APPLICATIONS

Pros and Cons of Backtracking

- Advantages:
  - Backtracking is the best option for solving tactical problem.
  - Backtracking is effective for constraint satisfaction problem.
  - Backtracking is simple to implement.
  - Backtracking is always accurate.

Pros and Cons of Backtracking

- Disdvantages:
  - Not efficient for solving strategic problem.
  - The runtime is usually slow.
  - Need large amount of space for storing different states.
  - The basic approach detects the conflicts (failures) too late.

Applications of Backtracking

- Solving puzzles: e.g., Sudoku, crossword, N-puzzle, …
- Finding the shortest path through a maze
- Scheduling problems / Constraint Satisfaction problems
- Resource allocation problems
- Network optimizing problems
- …

Exercises

- Subset sum problem
- Remove invalid parentheses
- Print all permutation of a given string
- Partition of a set into K subsets with equal sum
- Combinational sum
- Print all solutions in N-Queen problem
- Print all longest common subsequences of 2 arrays