Assignment: Backtracking & Greedy Algorithms

1. **Implement a backtracking algorithm**

   Given a collection of amount values (A) and a target sum (S), find all unique combinations in A where the amount values sum up to S.
   Each amount value may be used only once in the combination.
   The solution set should not contain duplicate combinations.
   Amounts will be positive numbers.

   Example: A = [11,1,3,2,6,1,5]; Target Sum = 8
   Result = [3, 5]
   [2, 6]
   [1, 2, 5]
   [1, 1, 6]

   a. Describe a backtracking algorithm to solve this problem.
   b. Implement the solution in a function **amount(A, S)**. Name your file **Amount.py**
   c. What is the time complexity of your implementation, you may find time complexity in detailed or state whether it is linear/polynomial/exponential. etc.?

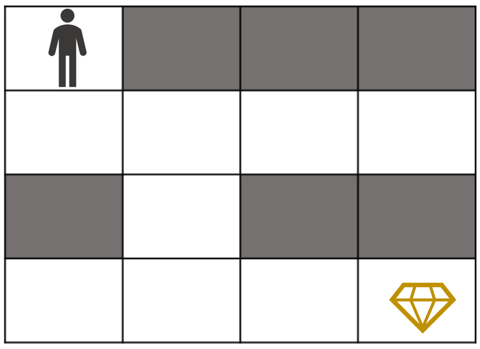
2. **Implement a Greedy algorithm**

   You are a pet store owner and you own few dogs. Each dog has a specific hunger level given by array hunger_level [1..n] (ith dog has hunger level of hunger_level [i]). You have couple of dog biscuits of size given by biscuit_size [1…m]. Your goal to satisfy maximum number of hungry dogs. You need to find the number of dogs we can satisfy.
   If a dog has hunger hunger_level[i], it can be satisfied only by taking a biscuit of size biscuit_size [j] >= hunger_level [i] (i.e biscuit size should be greater than or equal to hunger level to satisfy a dog.)
   Conditions:
   You cannot give same biscuit to two dogs.
   Each dog can get only one biscuit.
   Example 1:
   Input: hunger_level[1,2,3], biscuit_size[1,1]
   Output: 1
   Explanation: Only one dog with hunger level of 1 can be satisfied with one cookie of size 1.
   Example 2:
   Input: hunger_level[2, 1], biscuit_size[1,3,2]
   Output: 2
   Explanation: Two dogs can be satisfied. The biscuit sizes are big enough to satisfy the hunger level of both the dogs.

   a. Describe a greedy algorithm to solve this problem
   b. Write an algorithm implementing the approach. Your function signature should be **feedDog(hunger_level, biscuit_size)**; hunger_level, biscuit_size both are one dimention arrays . Name your file **FeedDog.py**
   c. Analyse the time complexity of the approach.

----------------------(Ungraded question: you can try this question if time permits)---------------------

You are given a puzzle in the form of a nxn matrix. Your location is in the start of a matrix at top left corner (location [0][0]) location and there is treasure location at the destination of the matrix at the bottom right corner of the matrix (location [n-1][n-1]). You can only move left/right or up/down in the puzzle. Your goals is to find out if you can reach the treasure or not.
Matrix is marked with 1 where there is a path and with 0 where is a wall.
For example, this matrix represents below shown puzzle.
Matrix: [[1, 0, 0,0],[1, 1, 1, 1], [0, 1, 0, 0], [1, 1, 1,1]]



a. Describe a backtracking algorithm to solve the puzzle, you goal is to return True if you can reach the destination or return False otherwise.
b. Write the pseudocode for the algorithm that you described in a. Your function signature should be **reachTreasure (puzzle)** and it should return True/False
c. Implement the pseudocode to solve the problem. Name your file **Puzzle.py**