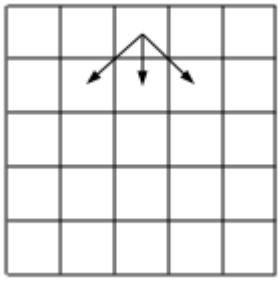
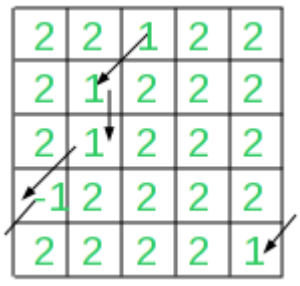
1. What is the idea of clustering? Explain how single-link clustering is similar to Kruskal's greedy MST algorithm. Describe one real-world application of clustering. Include citation(s) for any online reference(s) you use.
    a. Clustering is a type of unsupervised learning which divides items into clusters based on similarities between data points, dividing objects into a pre-set number of categories.
    b. The algorithm used to link objects is essentially the same as Kruskal's algorithm, but the difference is that the order in which clusters are formed matter in single link clustering, while with Kruskal's it doesn't
    c. One use could be for analyzing market data, for example clustering people with similar purchasing habits together and showing people in the same clusters similar advertisements that they might be more susceptible to as a group.
    d. https://www.geeksforgeeks.org/clustering-in-machine-learning/#
    e. https://en.wikipedia.org/wiki/Single-linkage_clustering

2. Mylar the robot has a simple life. Its sole purpose is to navigate from the top row of an NxN grid to the bottom row, while consuming as little fuel as possible. Mylar is able to choose its starting column. At each time step, it can select from one of three possible actions: move straight down, or move diagonally down left or right:



If Mylar ever attempts to step off the left or right edge of the grid, it reappears in the appropriate location on the opposite side.

Complicating Mylar's life is the fact that there may be a different fuel cost associated with traversing each cell in the grid. Here is an example grid along with the optimal sequence of decisions (with a total fuel cost of 5):



Your goal is to develop a dynamic programming algorithm that Mylar can use to solve this problem. You should provide an English language description, clear pseudocode (or

well-documented code), and a performance analysis. The input to the algorithm is the NxN grid of numbers. The output of the algorithm should be the total fuel usage; you do not need to return the path itself.

a. See code file
b. Time complexity = O(nxn)

3. Describe a real-world application of the Knapsack problem. Include citations for any online source(s) you use.

    a. The first real world application that came to mind for me with the knapsack problem was in relation to video games. Many video games have a system in place that limits the amount of weight a player can carry, and items have a specific weight and sell value. A penalty is frequently given to a player for crossing the weight threshold, often in the form of slowed movement speed or weakened attacks. This can lead to frustrating moments of scrolling through all the items you've picked up to search for what is best to get rid of to negate the penalty. The knapsack problem could be implemented to give players advice on which items are best to drop from their inventory when this occurs, minimizing the amount of time the player has to spend on frustrating inventory management.

4. Suppose we have a solution to the *n*-Queens problem instance in which n = 4. Can we extend this solution to find a solution to the problem instance in which n = 5? Can we then use the solution for n = 4 and n = 5 to construct a solution to the instance in which n = 6 and continue this dynamic programming approach to find a solution to any instance in which n > 4? Justify your answer.
    a. Assume that we used a backtracking approach to find the n=4 solution
    b. We can then find n=5 using the same approach and the same partial solution
    c. For n=4 there are only 2 solutions
    d. We can then take those two solutions and work to add the additional queens into those solutions, which works in the case of n=5
    e. This does not extend to n =6 as the solutions here are fully distinct from the earlier n=4 and n=5 solutions
    f. As this does not work for n= 6, it obviously can not work for all n> 4
5. Find at least two instances of the *n*-Queens problem that have no solutions.
    a. With n=2 or n=3, the board is so small that the queens will always attract each other
    b. No matter where we place the 2 queens below the will always be adjacent , or diagonal to each other

c. Because no 3 queens can share a row or column, eventually one will end up attacking the other diagonally
