1. I approached the challenge by employing the technique that involved two utility functions, namely find and union." which efficiently managed disjoint sets. After initializing a list to represent parent cities, I organized the roads in ascending order based on their associated maintenance costs. While progressing through these shorted roads, I incremented an accumulating total when the cities at the ends belonged to separate sets. Subsequently, I utilized the union function to merge these sets. The input details where extracted from 'input 1_2.txt, and the resulting output, signifying the minimized maintenance cost achieved by thoughtfully selecting a subset of roads, was recorded within output1_2.txt.

2. I used dynamic programming to calculate the number of unique ways Freddy can ~~achieves~~ achieve this. The count-ways() function recursively determines the count while utilizing an array dp to store previously computed results, minimizing redundancy. The input is read from "input2_4.txt", and the calculated result is written to 'output2_4.txt.' This approach

optimizes the solution by avoiding repetitive calculations.

3. I defined a list dp of length X+1 to store the minimum number of coins required to make up each amount from 0 to X. I initialize dp to 0 and all other elements to a large number. INF. Then, for each coins denomination C[i], I iterated over all amounts from C[i] to X and update dp[j] as the minimum of dp[j] and dp[j-C[i]]+1, since I can either use the coin C[i] or not to make up the amount j. Finally, I returned dp[X] as the minimum number of coins required to make up the target amount, or -1 if dp[X] is still INF.