Assignment-2 Analysis

Task-1:

In this task, there's the algorithm needs to be written to find the optimal play for the coin taking game. Assuming both players play optimally, the algorithm used in the task will determine the optimal play of the day between player1 and player2 where player1 achieves the highest score.

The optimal solution is achieved by using dynamic programming where the table is used to use two piles, pile1 and pile2 in the way such that the different possible solutions for approaching variety of strategies can be shown. That's how you know what the consequences might be if you take the coin off from the pile and this is the way to decide on which pile should you choose to pick the coin from. That's how both players will keep playing until there is no coin in the pile.

It can be quite difficult to follow up with two long arrays or piles. When two players are having two large piles, it might give fewer effective results due to efficiency of strategy logic.

The time complexity for this algorithm or the function in the implementation is O(NM) where N is the number of elements in pile1 and M is the number of elements in pile2.

Task-2:

In this task, another algorithm needs to be designed to find a word in the Snake-words grid given in the file. Grid is list of list which is N*N table and the word needs to be found from the table to get the positions towards finding the word.

Backtracking is used to keep track of the position as you move along the table by iterating through word by word. The array keeps track of the position as moving forward in the table. Whenever the letter of the word is found, it will return the position of the letter in the table in (I, j) format where i is the row so is j the column. Once, the word is found, it returns true and move forward for the next word. As it can move up, down, left, right, diagonally, it will return true if there's any possibility of having the next word in the neighborhood. If there's nothing in the same row next column, it'll backtrack and go to the next row same column. Still, if it can't match the word, it'll backtrack and check for the word by going diagonally. If it doesn't match, that should return false and word not found. These are all the comparison written in the algorithm to keep checking for the words to print the positions of the valid path it can take.

Time complexity for the algorithm is O(KN^2) as N is the length of grid table as there are N^2 number of elements while K is the number of characters in word. As it will move along all the elements of grid table while checking for the word which is having K characters.