Name: Chin Wen Yuan

Student ID: 29975239

Description for Assignment 2 FIT2004

Task 1 : Coin Taking Game (best_score(pile1,pile2))

First, a memoization table is built with size (length of pile1+1) *(length of pile2+1), addition of 1 is to save the result for the base case (when both piles is empty, which optimal value both players can take will be 0). In this table, each space saves a tuple (optimal coin value that player1 is able to take, optimal coin value that player 2 is able to take) for each round of coin taking. Other than that, I also build the first row and column of the table as that from left to right and up to bottom, where even indexed-number element is adding it self and the even indexed-number element before it and vice versa for the odd indexed-number elements. The sum of each tuple is equal to the maximum total coin value at that position. For other elements, the optimal value for player is computed as the sum of coins at the point deduct the minimum among the left and above (for player1), for player 2 just simply get the total value of coins at the position and deduct what player 1 gets.

For each space in the memoization table, the optimal value of coins for both of the players is calculated and recorded. This means that the time complexity here is O(NM) where N is the number of elements in pile1 and M is the number of elements in pile2.

After that, I use backtracking to get the sequence of coin taking. Starting from the last element in the list (far right and bottom element) I compare the optimal value that player 1 can take for the element to the left and above. I will choose the smaller one (as player 1 would want player 2 to take less in order to win) and if the left one is chosen, it indicates that a coin is taken from pile2, if the above one is chosen, it indicates that a coin is taken from pile1. The time complexity here is O(N+M) where N+M is the total number of coins.

The space complexity is NM for the memoization table and N+M for the sequence table, resulting in O(NM) auxiliary space complexity.

The function will return the maximum value player 1 can get and the sequence for the coin taking.

Task 2 : Snake-words (is_in(grid,word))

First the function finds the first alphabet from the grid that matches the first letter in the word, then the first match is put in to another function to find if there is a matching next alphabet for the next letter. The function search through all the 8 positions beside the match (includes diagonal), if a match is found, it will recurse to find if next match exist. A counter is used to keep track of the number of recursion, if counter is equal to the length of word, it means that I have successfully found my word, if one of the recursion returns False, it means that I was unable to find a match in the grid and I will ignore that chain of recursion and only continue the ones that I am still able to find a match.

The time complexity to find the first matching alphabet will be O(N^2) and the time complexity for recursion is O(K) where K is the number of characters in word (length of word). This will result in time complexity of O(K N^2).

The space complexity is O(K) where K is the length of word, it is used to store the position of alphabet if a chain of position that matches the word is found.