CO322 – Data Structures and Algorithms

Lab 02 - Algorithm Explanation for the Problems

E/18/180 – Kodituwakku M.K.N.M.

---

1. The Power Sum

In this problem I used a recursive method to solve the problem. So for the recursive method I defined two bases. First I defined the recursive function as **helpSum** this function take three inputs. Those are

- Total
- Power
- Num

In the function first value will be calculated as **val = total – num^{pow}** so by the value of the val variable I defined the base cases such as

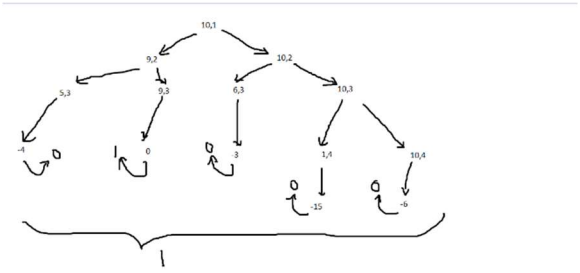
- Val == 0 : then this means that the function has divided into total power sums so then it returns 1
- Val < 0 : this means that there are no power sums to this combination so then function must return 0

If those base cases not satisfied then function will be recursively called as

helpsum(val,pow,num+1) + helpsum(total,pow,num+1)

to check the current combination to check all the combinations

for an example lets take powerSum(10,2)



So this returns 1 because there exist only $1^2+3^2 = 10$

2. Caesar Cipher

In this question first I checked two conditions

- Key = 26 : this means that the string doesn't get encoded
- Key > 26 : then I divided the key by 26 and took the reminder as the key because that is the amount that the string is converted
  If above conditions are not satisfied then continue to next operations.

Then I iterated through the string character by character.
First I checked whether the character is uppercase or lowercase.
Then I checked after adding the key whether the ascii value is greater than z/Z if so then I again made the character as a/A and added the remaining value to the a/A.
After iterating through all the characters I returned the encoded string.

3. Climbing the leaderboard

I started the loop from the top player score and from this I was able to skip doing to 2 for loops from 0 to list size.

Comparing top ranks and when the score of the player was added, then I just continued from that ranked index.

4. Closest Numbers

First I made a function to sort the array and then I sorted the input array. Then I made a new array to store the difference between adjacent numbers and then I looped through the input array and took the difference and store it in the new array and also I stored the minimum difference in a variable after taking the all the difference between adjacent numbers next I looped again the diff array and if the difference is not the minimum value I changed the input array element to max int value if the difference is min difference value then I don't changed the adjacent two values and after looping through all the array then I returned the array.

When printing if the element in the array isn't the max int value then I printed those values.