

# HEATHER MATARUSE

## ADVANCED ALGORITHMS SUMMATIVE

GITHUB CODE [HERE](#)

DECEMBER 16 , 2021



# Question 1

```
main.py
1 #this is a simple algorithm that takes n number of elements
2 #and returns the sum of them
3 #This program also calculates that time taken to execute a certain number of elements
4 #to calculate their sum using the time function
5
6
7
8
9 # here is my function for calculating the sum_of_numbers
10 def sum_of_n_numbers(n):
11
12     #here is our variable sum of numbers set to zero since we do not have any numbers yet
13     sum_of_numbers = 0
14
15     #here we start by counting from first number |
16     for numbers in range(1,n+1):
17         sum_of_numbers += numbers
18
19     # here we return the sum of the n numbers
20     return sum_of_numbers
21
22 #this is the function that is already in python that I am calling
23
input
then input is 10. The output is 55 || Execution time: 9.470000000001005e-06seconds
then input is 1000. The output is 50005000 || Execution time: 0.000738181000000001seconds
then input is 1000000. The output is 500000500000 || Execution time: 0.006538495seconds
then input is 1000000000. The output is: 500000000500000000 || Execution time: 46.098461454seconds
..Program finished with exit code 0
press ENTER to exit console.
```

## Pseudo-code

- create a function that calculates the sum of the **n** numbers
- Inside our function, we have a variable sum of numbers at zero that tracks the number of numbers
- Then you use a for loop to loop through the numbers you are adding together
- The function should make sure that it returns the sum of the n items

## Time Complexity

- As the algorithm runs it takes  $O(n)$  time because as we can see from the above output the time increases as the number of the numbers increases

## Observations

- For the 10 , 1000, and 100000 elements that running time was less but for the 1000000000 it took a while for the output to be presented as we can see it took 46 seconds

## Question 2

```
main.py
10 #i the created a list where the final grades will be stored
11 n = []
12
13 # I creates a for loop while loop to check
14 # the number the grades is so that if can round it off to give the final grades
15 for grades in original_grades:
16     #if the grades is lower than 38 its considered as fail hence in that case round is not necessary
17     if grades < 38:
18         n.append(grades)
19
20     else:
21
22         #then i find the nearest multiple of 5 is the grades is >= 38
23         rounded_grades = math.ceil(grades/5)*5
24
25         if rounded_grades - grades < 3:
26             n.append(rounded_grades)
27
Input
Student original grade : [73, 38, 67, 33]
Student final grade:    [75, 40, 67, 33]
...Program finished with exit code 0
Press ENTER to exit console.
```

### Pseudocode

- I created a function called gradingStudents that takes in a list of the students' grades.
- After that, I created an empty list called final grade to hold the rounded grades.
- For the input array of the student's grades .
- If the grade is less than 38, it should be added to the final grade.
- If your grade is higher than or equal to 38,
- you then find the nearest multiple of 5 by dividing the grade by 5 and rounding the result to the nearest whole number.
- and then multiply it by 5 to save the result as rounded grade
- Append the rounded grade to the final grade array if the difference between the rounded grade and the original grade is less than 3.
- Otherwise, add the original grade to the final grade.

## Question 3

```
main.py
16 encrypted = encryption(text,key)
17
18
19 # this if the functin that will handle the decryption
20 def decryption(encrypted,key):
21     a = 0
22     decrypted_message = ""
23     #here is where i checked for the modulus
24     if len(encrypted) % 2 == 0 and key % 2 == 0:
25         # here i did floor division
26         b = len(encrypted)//key
27
28         while a < b:
29
30             #here I looped to check the length of the text
31             for i in range(a,len(encrypted),b):
32                 decrypted_message += encrypted[i]
33                 a+=1
34     if len(encrypted) % 2 == 0 and key % 2 != 0:

```

input

```
Plain text : Plain text
Encrypted text: Pittlnea x
Decryption text: Plain text

...Program finished with exit code 0
Press ENTER to exit console.[]
```

### Pseudo-code for decryption

- Create an encryption function that takes two arguments, the text and the key. Within the function, create an empty string variable called encrypted message to store the encrypted message, as well as an integer variable called start (set to 0), which will help you determine where to start and end based on the key. The algorithm here is to loop through the key and then through the text in any gaps in the key. For example, if the key is 2, we will loop through the text in 2 step increments and add the characters to the storing variable (encrypted message). In the end, we return the encrypted message

### Pseudo-code for decryption

- create the function , Check for edges cases if the length of the text is even and the key is even, loop through the length of the word starting at start and take b steps every time. Add the current index of start in the encrypted message character to decrypted\_message and then return decrypted\_message

## Question 4

```
1- def superDigits(x,k):
2-     s = digsum(x)
3-     return sup_digit(str(int(s)*k))
4-
5- def sup_digit(x):
6-     if len(x) <= 1:
7-         return x
8-     else:
9-         return sup_digit( digsum(x) )
10-
11- def digsum(x):
12-     return str(sum((int(i) for i in list(x))))
13-
Input
= 9875, k = 4, concatenated value = 9875987598759875 superdigit is 8
= 148, k = 3, concatenated value = 148148148superdigit is 3
= 56345, k = 6, concatenated value = 563455634556345563455634556345 superdigit is 3

..Program finished with exit code 0
Press ENTER to exit console.
```

### Pseudo-code

- created a function that takes two arguments, n and k which will calculate the super digit
- I also have another function that will calculate the sum
- And another function that checks if the sum is a single number or not
- Lastly I just printed out the output using the test cases

## Question 5

```
main.py
20 #in the shortestReach function we have n which is the total number of nodes in the graph
21 # edges as the total number of edges we have
22 #s which is our starting point to travel in the graph
23 def shortestReach(n,edges,s):
24     my_heap = [(0, s)]
25     my_graph = compose_graph(edges)
26     answer = {}
27     for i in range(1, n + 1):
        input
The Shortest Distance is [24, 3, 15]
...Program finished with exit code 0
Press ENTER to exit console.
```

### Pseudo-code

I wrote a function called compose graph to generate the graph's edges.

I then made a dictionary with the edges as keys and values.

Then I mapped the edges to one another, storing the weights in a tuple because I don't want the value to change.

The function produces a graph, which I will use in the following function.

The next step in the algorithm is to write a function called shortestReach that accepts arguments such as the number of edges, the array of edges, and the starting point, which is s.

In the function where I will use the compose graph function, we will create an answer dictionary to keep track of the shortest reach, with the edges as the key and infinity as the value.

Because we're starting with s, set the value of s in the answer to 0

Lastly, you loop through the dictionary answer with values as the shortest reach and append them to shortest\_reach which return shortest\_reach the array with the shortest reach of each edge starting from one