**LAB DAY-2 05-06-24**

**1.Reverse of a given number using recursion**

def rev(nums):

return nums[::-1]

num=[1,2,3]

print(rev(num))

**OUTPUT:**

[3,2,1]

**2.Perfect number**

def perf(n):

if n< 2:

return False

s= 1

for i in range(2, int(n\*\* 0.5) + 1):

if n% i == 0:

s+= i

if i != n// i:

s += n// i

return s== n

num =6

if perf(num):

print(“it is a perfect number”)

else:

print(“it is not a perfect number”)

**OUTPUT:**

It is a perfect number

**3.** **Write program that demonstrates the usage of these notations by analyzing the time complexity of some example algorithms.**

def linear(arr, target):

#Perform a linear search on the list.

#Time complexity: O(n), Θ(n), Ω(1)

for i in range(len(arr)):

if arr[i] == target:

return i

return -1

def factorial(n):

#Calculate the factorial of n.

#Time complexity: O(n), Θ(n), Ω(n)

if n == 0 or n == 1:

return 1

else:

return n \* factorial(n - 1)

arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

t = 7

nfact = 5

#LINEAR SEARCH

print("Linear Search:")

index = linear(arr, t)

if index != -1:

print(f"Element found at index {index}")

else:

print(f"Element not found")

#FACT

print("\nFactorial Calculation:")

result = factorial(nfact)

print(f"Factorial is {result}")

**OUTPUT:**

Linear Search:

Element found at index 6

Factorial Calculation:

Factorial is 120

**4. Write programs that demonstrate the mathematical analysis of non-recursive and recursive algorithms.**

#Recursive: factorial

def recursive(n):

if n == 0 or n == 1:

return 1

else:

return n \* recursive(n - 1)

n = 5

print(recursive(n))

**OUTPUT:**

**120**

#Nonrecursive: factorial

def iterative(n):

for i in range(1, n + 1):

result \*= i

return result

n = 5

print(iterative(n))

**OUTPUT:**

**120**

**5. Write programs for solving recurrence relations using the Master Theorem, Substitution Method, and Iteration Method will demonstrate how to calculate the time complexity of an example recurrence relation using the specified technique:**

import math

def master (n):

return f"{n}^log(2,2) \* log({n}) = {n \* math.log2(n)}"

def substitution(n):

return f"Guess: T(n) = O(n log n)\nVerification: T(n) = 2T(n/2) + Theta(n) = 2O((n/2) log(n/2)) + Theta(n) = O(n log n)"

def iteration (n):

iterations = ["T(n) = 2T(n/2) + Theta(n)"]

for i in range(1, int(math.log2(n)) + 1):

iterations.append(f"= 2^({i})T(n/2^{i}) + {2\*\*(i-1)}Theta(n)")

iterations.append("= n(1 + log(2) + log(3) + ... + log(n))")

iterations.append(f"= n \* log(n!)")

return "\n".join(iterations)

n = 8

print("Using Master Theorem:")

print(master (n))

print()

print("Using Substitution Method:")

print(substitution (n))

print()

print("Using Iteration Method:")

print(iteration (n))

**OUTPUT:**

**Using Master Theorem:**

**8^log(2,2) \* log(8) = 24.0**

**Using Substitution Method:**

**Guess: T(n) = O(n log n)**

**Verification: T(n) = 2T(n/2) + Theta(n) = 2O((n/2) log(n/2)) + Theta(n) = O(n log n)**

**Using Iteration Method:**

**T(n) = 2T(n/2) + Theta(n)**

**= 2^(1)T(n/2^1) + 1Theta(n)**

**= 2^(2)T(n/2^2) + 2Theta(n)**

**= 2^(3)T(n/2^3) + 4Theta(n)**

**= n(1 + log(2) + log(3) + ... + log(n))**

**= n \* log(n!)**

1. **Intersection of two arrays with unique elements**

def intersection(nums1, nums2):

return list(set(nums1) & set(nums2))

nums1 = [1,2,3,4]

nums2 = [4,3,1,7]

print(intersection(nums1, nums2))

**OUTPUT:**

**[2]**

1. **Intersection of two arrays with duplicate elements**

def intersection(nums1, nums2):

return list(set(nums1) & set(nums2))

nums1 = [1,2,1,4]

nums2 = [4,4,3,1,]

print(intersection(nums1, nums2))

**8.Sorting array in ascending order without built-in functions**

def bubble (arr):

n = len(arr)

for i in range(n):

for j in range(0, n-i-1):

if arr[j] > arr[j+1]:

arr[j], arr[j+1] = arr[j+1], arr[j]

num = [64, 34, 25, 12, 22, 11, 90]

bubble (num)

print("Sorted list is:", num)

**OUTPUT:**

**Sorted list is: [11, 12, 22, 25, 34, 64, 90]**

**9.Sorting array with half odd and half even:**

def sort(num):

odd = sorted([x for x in num if x % 2 != 0])

even = sorted([x for x in num if x % 2 == 0])

result = []

i = j = 0

for k in range(len(nums)):

if k % 2 == 0:

result.append(even[i])

i += 1

else:

result.append(odd[j])

j += 1

return result

n= [8,1,7,3,4]

print(sort(n))

**OUTPUT:**

[1,3,4,7,8]

1. **Sort the array with odd indices having odd numbers and even indices having even numbers :**

def sort(nums):

even= 0

odd= 1

sortednum = [0] \* len(nums)

for num in sorted(nums):

if num % 2 == 0:

sortednum[even] = num

even+= 2

else:

sortednum[odd] = num

odd+= 2

return sortednum

n= [4, 2, 5, 7]

print(sort(n))

**OUTPUT:**

[2,4,5,7]