|  |  |  |
| --- | --- | --- |
| **CLASS:** M.Sc. CS |  | **SEM:** I(2022-2023) |
| **SUBJECT:** Analysis of Algorithm and Researching Computing |  | **PAPER:** I |
| **NAME:** Aditya Suresh Mazire |  | **ROLL NO.:** 517 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **INDEX** |  |  |
| **NO** | **DATE** | **TITLE** | **PAGE NO.** | **SIGN** |
|  |  |  |  |  |
| 1 |  | Randomized Selection Algorithm | 2 |  |
| 2 |  | Heap Sort Algorithm | 3 |  |
| 3 |  | Radix Sort Algorithm | 5 |  |
| 4 |  | Bucket Sort Algorithm | 7 |  |
| 5 |  | Floyd - Warshall Algorithm | 9 |  |
| 6 |  | Counting Sort Algorithm | 11 |  |
| 7 |  | Set Covering Problem | 12 |  |
| 8 |  | Subset Sum Problem | 13 |  |

**1)Write a Program for Randomized Selection Algorithm.**

from random import randrange  
def set\_cover(universe, subsets):  
 elements = set(e for s in subsets for e in s)  
 if elements != universe:  
 return None  
 covered = set()  
 cover = []  
 while covered != elements:  
 subset = max(subsets, key=lambda s: len(s - covered))  
 cover.append(subset)  
 covered |= subset  
 return cover  
  
def main():  
 universe = set(range(1, 11))  
 subsets = [set([1, 2, 3, 8, 9, 10]),  
 set([1, 2, 3, 4, 5]),  
 **set([4, 5, 7]),** set([5, 6, 7]),  
 set([6, 7, 8, 9, 10])]  
 cover = set\_cover(universe, subsets)  
 print(cover)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**OutPut:-**

[{1, 2, 3, 8, 9, 10}, {4, 5, 7}, {5, 6, 7}]

Process finished with exit code 0

**2) Write a Program for Heap Sort Algorithm**

def heapify(arr, n, i):  
 largest = i  
 l = 2 \* i + 1  
 r = 2 \* i + 2  
 if l < n and arr[i] < arr[l]:  
 largest = l  
 if r < n and arr[largest] < arr[r]:  
 largest = r  
 if largest != i:  
 arr[i],arr[largest] = arr[largest],arr[i]  
 heapify(arr, n, largest)  
   
def heapSort(arr):  
 n = len(arr)  
 for i in range(n, -1, -1):  
 heapify(arr, n, i)  
 for i in range(n-1, 0, -1):  
 arr[i], arr[0] = arr[0], arr[i]  
 heapify(arr, i, 0)  
  
arr = [ 12, 11, 13, 5, 6, 7]  
heapSort(arr)  
n = len(arr)  
print ("Sorted array is")  
for i in range(n):  
 print ("%d" %arr[i])

**OutPut:-**

Sorted array is

5

6

7

11

12

13

Process finished with exit code 0

**3) Write a Program to perform Radix Sort Algorithm**

def countingSort(arr, exp1):  
 n = len(arr)  
 output = [0] \* (n)  
 count = [0] \* (10)  
 for i in range(0, n):  
 index = (arr[i]/exp1)  
 count[int(index)%10 ] += 1  
 for i in range(1,10):  
 count[i] += count[i-1]  
  
 i = n-1  
 while i>=0:  
 index = (arr[i]/exp1)  
 output[ count[int(index)%10 ] - 1] = arr[i]  
 count[int(index)%10 ] -= 1  
 i -= 1  
  
 i = 0  
 for i in range(0,len(arr)):  
 arr[i] = output[i]  
  
def radixSort(arr):  
 max1 = max(arr)  
 exp = 1  
 while max1/exp > 0:  
 countingSort(arr,exp)  
 exp \*= 10  
   
arr = [ 170, 45, 75, 90, 802, 24, 2, 66]   
radixSort(arr)  
for i in range(len(arr)):  
 print(arr[i])

**OutPut:-**

2

24

45

66

75

90

170

802

Process finished with exit code 0

**4) Write a Program to Perform Bucket Sort Algorithm**

def insertionSort(b):  
 for i in range(1, len(b)):  
 up = b[i]  
 j = i - 1  
 while j >=0 and b[j] > up:  
 b[j + 1] = b[j]  
 j -= 1  
 b[j + 1] = up  
 return b  
def bucketSort(x):  
 arr = []  
 slot\_num = 10  
 for i in range(slot\_num):  
 arr.append([])  
 for j in x:  
 index\_b = int(slot\_num \* j)  
 arr[index\_b].append(j)  
 for i in range(slot\_num):  
 arr[i] = insertionSort(arr[i])  
 k = 0  
 for i in range(slot\_num):  
 for j in range(len(arr[i])):  
 x[k] = arr[i][j]  
 k += 1  
 return x  
   
x = [0.897, 0.565, 0.656,0.1234, 0.665, 0.3434]  
print("Sorted Array is")   
print(bucketSort(x))

**OutPut:-**

Sorted Array is

[0.1234, 0.3434, 0.565, 0.656, 0.665, 0.897]

Process finished with exit code 0

**5) Write a Program to Perform Folyd-Warshall algorithm**

V = 4  
INF = 99999  
def floydWarshall(graph):  
 dist = list(map(lambda i :list(map(lambda j : j , i)) , graph))  
 for k in range(V):  
 for i in range(V):  
 for j in range(V):  
 dist[i][j] = min(dist[i][j] ,dist[i][k]+ dist[k][j])  
 printSolution(dist)  
   
def printSolution(dist):  
 print ("Following matrix shows the shortest distances\ between every pair of vertices")  
 for i in range(V):  
 for j in range(V):  
 if(dist[i][j] == INF):  
 print ("%7s" %("INF"))  
 else:  
 print ("%7d\t" %(dist[i][j]))  
 if j == V-1:  
 print ()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 graph = [[0,5,INF,10],  
 [INF,0,3,INF],  
 [INF, INF, 0, 1],  
 [INF, INF, INF, 0] ]  
  
 floydWarshall(graph);

**OutPut:-**

Following matrix shows the shortest distances between every pair of vertices

0 5 8 9

INF 0 3 4

INF INF 0 1

INF INF INF 0

Process finished with exit code 0

**6) Write a Program for Counting Sort Algorithm in python**

def countSort(arr):  
 output = [0 for i in range(256)]  
 count = [0 for i in range(256)]  
 ans = ["" for \_ in arr]  
 for i in arr:  
 count[ord(i)] += 1  
 for i in range(256):  
 count[i] += count[i-1]  
  
 for i in range(len(arr)):  
 output[count[ord(arr[i])]-1] = arr[i]  
 count[ord(arr[i])] -= 1  
  
 for i in range(len(arr)):  
 ans[i] = output[i]  
 return ans  
if \_\_name\_\_ == "\_\_main\_\_":  
 arr = "geeksforgeeks"  
 ans = countSort(arr)  
 print ("Sorted character array is %s" %("".join(ans)) )

**OutPut:-**

Sorted character array is eeeefggkkorss

Process finished with exit code 0

**7) Write a program for Set Covering Problem**

from random import randrange  
def set\_cover(universe, subsets):  
 elements = set(e for s in subsets for e in s)  
 if elements != universe:  
 return None  
 covered = set()  
 cover = []  
 while covered != elements:  
 subset = max(subsets, key=lambda s: len(s - covered))  
 cover.append(subset)  
 covered |= subset  
 return cover  
  
def main():  
 universe = set(range(1, 11))  
 subsets = [set([1, 2, 3, 8, 9, 10]),  
 set([1, 2, 3, 4, 5]),  
 set([4, 5, 7]),  
 set([5, 6, 7]),  
 set([6, 7, 8, 9, 10])]  
 cover = set\_cover(universe, subsets)  
 print(cover)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**OutPut:-**

[{1, 2, 3, 8, 9, 10}, {4, 5, 7}, {5, 6, 7}]

Process finished with exit code 0

**8) Write a Program for found a subset with given sum**

def isSubsetSum(set,n, sum) :  
 if (sum == 0) :  
 return True  
 if (n == 0 and sum != 0) :  
 return False  
 if (set[n - 1] > sum) :  
 return isSubsetSum(set, n - 1, sum);  
 return isSubsetSum(set, n-1, sum) or isSubsetSum(set, n-1, sum-set[n-1])  
  
set = [3, 34, 4, 12, 5, 2]   
sum = 9  
n = len(set)   
if (isSubsetSum(set, n, sum) == True) :  
 print("Found a subset with given sum")  
else :  
 print("No subset with given sum")

**OutPut:-**

Found a subset with given sum

Process finished with exit code 0