# DAA LAB:

1. a) Design and implement an algorithm for recursive binary search. Comment on the efficiency and the time complexity of the same.

b) Ramesh wants to arrange a block of unsorted numbers using an approach by dividing it into buckets and sort each of the buckets individually. Design and implement an algorithm to solve it using random numbers. Comment on the efficiency and the time complexity of the same.

Ans: a)

Algorithm:

//Searches for an element in a sorted array by binary search

//Input: A sorted array A[0,1,.......,n-1] of n elements, element to be found and index of first (low) and last (high) element

// Output: If element present in list, returns its position,else returns failure

binary\_search(A,low,high,key) if low<=high

mid ← (low+high)//2

if key=A[mid] then

return mid

else if key < A[mid] then

return binary (A,low,mid-1,key)

else

end if

return binary (A,mid+1,high,key)

return failure if search is not a success and low > high

Program:

def search(low,high,L,key): if(low<=high):

mid = (low+high)//2 if key==L[mid]:

return mid elif key<L[mid]:

return search(low,mid-1,L,key)

else:

return search(mid+1,high,L,key) return None

import time

k = [i for i in range(300000)] key = 300000

low = 0

high = len(k)-1 start = time.time()

match = search(low,high,k,key) end = time.time()

if match==None: print('Unsuccessful search')

else:

print('Key Found at '+ str(match+1))

print ("Time taken: " + str(end-start) + ' seconds') Time complexity : O(log n)

b) Algorithm:

//Input: Array A[0,1,2,.......,n-1] of orderable elements, with each element less than 1

//Output: Array of sorted elements in non-decreasing order

bucket\_sort(A)

n ← length (A)

for i ← 0 to n-1 do

Insert A[i] into list B[n\*A[i]] for i ← 0 to n-1 do

Sort B with insertion sort

Concatenate the list B[0],B[1],B[2],.........,B[n-1] together in order Program:

def insertion\_sort(A):

for i in range(1,len(A)): r = A[i]

j = i-1

while j>=0 and A[j]>r: A[j+1]=A[j]

j=j-1 A[j+1]=r

return A

def bucket\_sort(A): B=[]

for i in range (10): B.append([])

n = len(A)

for i in range(n): B[int(n\*A[i])].append(A[i])

for i in range(n):

B[i] = insertion\_sort(B[i]) result=[]

for i in range(10): result.extend(B[i])

return result

A=[.78,.17,.37,.26,.72,.94,.21,.12,.23,.68]

A = bucket\_sort(A) print(A)

Time complexity: Average - O(n+k)

Worst - O(n^2)

1. a) Given f(n)= 7n+5, Write a program to prove that f(n)= O(n) and find the n0 value. Plot a graph for f(n) and c\*g(n) where c is a constant and for varying n values (10 to 30).

b) Given a set of men and women design and implement Gale- Shapeley algorithm to determine the stable set of marriages among them. Comment on the efficiency and the time complexity of the same. Assumptions: Men propose first according to their preference list. Women can choose a better partner based on the preference.

Men’s preference list Women’s preference list

A V W X V A B C

B W V X W B C A

C V W X X C A B

Ans:

1. Program:

print("Your function is 7n+5")

print ("g(n) = n ") print("Assuming c as 8") for i in range (30):

a1 = 7\*i+5 a2 = 8\*i

if (a2>=a1): n0 = i break

print("Value of n0: ", n0)

print ("Value\t\tF(n)\t\tc\*G(n)") for i in range (10,31):

print (i,"\t\t",7\*i+5,"\t\t",8\*i)

1. Algorithm:

//Input: A set of n men and women with their preferences

//Output: A set S of stable matches

∈ ∈

Initially all m M and w W are free

While there is a man m who is free and hasn’t proposed to every woman Choose such a man m

Let w be the highest ranked woman in m’s preference list to whom m has not yet proposed

If w is free then

(m,w) become engaged

else

w is currently engaged to m’

If w prefers m to m’ then (m,w) become engaged m’ becomes free

endif

else endif

m remains free

End while

Return the set S of engaged pairs Program:

# class Man:

def init (self,name,preference): self.name=name self.status=False self.preference=preference self.partner=""

self.proposed=[] def hopeleft(self):

if len(self.proposed)==len(Women): #THERE WONT BE ANY HOPE IF the GUY HAS already PROPOSED TO EVERYONE.

return False return True

class Woman:

def init (self,name,preference): self.name=name self.status=False self.preference=preference self.partner=""

def objectfromname(w):

for i in Women+Men: #TAKE THE OBJECT NAME AS a STRING and RETURN THE OBJECT be it a Man or WOman

if i.name==w: return i

def loveyoumore(m,w):

if w.preference.index(m.name)<w.preference.index(w.partner): #Check if this guy's better than ur partner.

return True return False

def thereissomeone():

for i in Men: #There's a man left who is SINGLE and hasn't proposed to every woman

if i.hopeleft() and i.status==False: return i

Men = [] Women = []

print ("Enter the number of men and women") n=int(input())

for i in range(n):

print ("Enter name of man " + str(i+1)) name=input()

print ("Enter preference list of man " + str(i+1)) k = list(map(str,input().split()))

print(k)

a = Man(name,k) Men.append(a)

for i in range(n):

print ("Enter name of woman " + str(i+1)) name=input()

print ("Enter preference list of woman " + str(i+1)) k = list(map(str,input().split()))

print(k)

a = Woman(name,k) Women.append(a)

while thereissomeone():

i=thereissomeone() #Choose the single man who hasnt proposed to every girl

wname=i.preference[len(i.proposed):] #UNPROPOSED WOMEN list wname=wname[0] #MOST PREFERRED WOMAN in UNPROPOSED.

w=objectfromname(wname) #You have her name as a string get her as an Object, to use operations.

i.proposed.append(w.name) #This guy just proposed to woman w OMG

if w.status==False: #if the girl is single then mingle i.partner=w.name

w.partner=i.name i.status=True w.status=True

else:

if loveyoumore(i,w): k=objectfromname(w.partner) #GET HER EX k.status=False #AXE HER EX, DUMP HIM. k.partner="" #MAKE HIM SINGLE i.partner=w.name #GET ENGAGED w.partner=i.name

i.status=True #CHANGE RELATIONSHIP STATUS

w.status=True for i in Men:

print(i.name,i.partner)

1. a) Design and implement recursive DFS algorithm to determine the traversal of a graph for

a given source node. Comment on the efficiency and the time complexity of the same.

b) In a shop, a set of jackets numbered according to their sizes are present where half of them are sorted and the rest are unsorted. Shopkeeper now takes the jacket from the unsorted part and put it into the sorted part until all the jackets are sorted according to their sizes. Design and implement an algorithm to solve it using random numbers. Comment on the efficiency and the time complexity of the same.

Ans:

a. Algorithm:

//Input : A graph G (V,E) and a source node s

//Output: DFS traversal of the given graph G

DFS(u):

Mark u as “Explored” and add u to R For each edge (u,v) incident to u

if v is not marked “Explored” then Recursively invoke DFS(v)

Endif

Endfor Program:

def dfs(v): visited[v]=1 print(v) R=[v]

while len(R)!=0: v=R.pop()

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

if ([v,j] in edgelist or [j,v] in edgelist) and visited[j]!=1: R.append(j)

dfs(j)

edgelist=[] visited=[]

n=int(input("Enter the number of vertices: ")) for i in range (n+1):

visited.append(0) print(visited)

edges = int(input("Enter the number of edges: ")) for i in range (edges):

edgelist.append([])

print ("Input edges ") for i in range(edges):

k = list(map(int,input().split(" "))) edgelist[i] = k

print ("Enter initial vertex ")

j = int(input()) dfs(j)

b) Algorithm:

//Input: An array A[0,1,2,......,n-1] of orderable elements

//Output: Array A[0,1,2,.......,n-1] sorted in non-decreasing order

Insertion\_sort(A)

for i ← 1 to n-1

R ← A[i] J ← i-1

While j>=0 and A[j]>R do

A[j+1] ← A[j]

j ← j-1 A[j+1] ← R

Program:

import random,time def sort(L):

N=len(L)

for i in range(1,N): r=L[i]

j=i-1

while j>=0 and L[j]>r: L[j+1]=L[j]

j-=1

L[j+1]=r

return L

alist=[random.randint(0,2000) for i in range(2000)] start=time.time()

sort(alist)

end=time.time() print(end-start,"Seconds") print(alist)

1. a) Given f(n)= 3n2+4n+3, Write a program to prove that f(n)= Ω(n) and find the n0 value. Plot a graph for f(n) and c\*g(n) where c is a constant and for varying n values (10 to 30).

b) A GPS navigation system needs an approach to discover the reachable areas in a given geographical region from a given source area. Design and implement an algorithm to find which nodes can be reached from a given source node for the following graph. Comment on the efficiency and the time complexity of the same.

Ans.

1. Program:

print ("The function is 3n^2 + 4n + 3") print ("G(n) = n ")

print ("Assuming c as 10") for i in range (30):

a1 = 3\*(i\*\*2)+4\*i+3 a2 = 10\*i

if (a1>=a2): n0 = i break

print ("The value of n0: ", n0) print ("n \t\t F(n) \t\t c\*G(n)") for i in range (10,31):

a1 = 3\*(i\*\*2)+4\*i+3 a2 = 10\*i

print (i,"\t\t",a1,"\t\t",a2)

1. Algorithm: BFS (s):

Mark s as ‘Visited’

Initialize R = {s} Define layer L0 = {s} While Li is not empty

For each node u ∈

Li

Consider each edge (u,v) incident to v If v is not marked ‘Visited’ then

Mark v visited

Add v to set R and to layer Li+1

EndIf

EndFor EndWhile

Program: def bfs(v):

visited[v]=1

print(v) L=[v]

while len(L)!=0: v=deleteq(L)

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

if ([v,j] in edgelist) and visited[j]!=1: print(j)

visited[j]=1

L.append(j)

edgelist=[] visited=[]

def deleteq(L): item=L[0] L.remove(item) return item

n=int(input("Enter the number of vertices: ")) for i in range (n+1):

visited.append(0) print(visited)

edges = int(input("Enter the number of edges: "))

for i in range (edges): edgelist.append([]) print ("Input edges ") for i in range(edges):

k = list(map(int,input().split(" "))) edgelist[i] = k

print ("Enter initial vertex ") j = int(input())

bfs(j)

1. a) Given f(n)= 7n2+7n+5, Write a program to prove that f(n)= θ(n2 ) and find the n0 value. Plot a graph for f(n), c1\*g(n)and c2\*g(n) where c1, c2is a constant and for varying n values (10 to 30).

b) Design and implement merge sort algorithm that takes random number input and displays the execution time required. State the design strategy used and time complexity of the same.

Ans:

1. Program:

print ("The function is 7n^2 + 7n + 5") print ("G(n) = n^2 ")

print ("Assuming c1 as 6 and c2 as 8") for i in range (30):

a1 = 7\*(i\*\*2)+7\*i+5 a2 = 6\*(i\*\*2)

a3 = 8\*(i\*\*2)

if a1>=a2 and a3>=a1: n0 = i

break

print ("The value of n0: ", n0)

print ("n \t\t F(n) \t\t c1\*G(n) \t\t c2\*G(n)") for i in range (10,31):

a1 = 7\*(i\*\*2)+7\*i+5 a2 = 6\*(i\*\*2)

a3 = 8\*(i\*\*2)

print (i,"\t\t",a1,"\t\t",a2, "\t\t", a3)

1. Algorithm: Mergesort(A[0,1,.......,n-1])

//Input: An array A of orderable elements

//Output: A sorted in non-decreasing order

if n>1

copy A[0,1,..........,[n/2]-1] to B[0,1,......,[n/2]-1]

copy A[[n/2]-1,.....,n-1] to C[0,1,......,[n/2]-1]

Mergesort (B[0,.....,[n/2]-1])

Mergesort (C[0,.....,[n/2]-1]) Merge(B,C,A)

Merge (B[0,.......,p-1], C[0,......,q-1], A[0,.........,p+q-1] )

//Input: Arrays B[0,....,p-1] and C[0,........,q-1] btoh sorted

//Output: Sorted array A[0,....,p+q-1] of the elements of B and C

I← 0; j← 0; k←0

While i < p and j < q do

If B[i]<=C[j]

A[k] ← B[i]; i ← i+1

Else

Endif Endwhile

If i=p then

A[k] ← C[j]; j ← j+1

Else

Copy C[j,.....,q-1] to A[k,......,p+q-1]

Copy B[i,......,p-1] to A[k,........,p+q-1]

Program:

def mergeSort(alist): print("Splitting ",alist) if len(alist)>1:

mid = len(alist)//2 lefthalf = alist[:mid] righthalf = alist[mid:]

mergeSort(lefthalf) mergeSort(righthalf)

i=0 j=0 k=0

while i < len(lefthalf) and j < len(righthalf):

if lefthalf[i] < righthalf[j]: alist[k]=lefthalf[i] i=i+1

else:

alist[k]=righthalf[j] j=j+1

k=k+1

while i < len(lefthalf): alist[k]=lefthalf[i] i=i+1

k=k+1

while j < len(righthalf): alist[k]=righthalf[j] j=j+1

k=k+1

print("Merging ",alist) import random,time

alist=[random.randint(0,100) for i in range(10)]

start=time.time() mergeSort(alist) end=time.time() print(end-start,"Seconds") print(alist)

1. a) Given f(n)= 4n+3, Write a program to prove that f(n)= O(n) and find the n0 value. Plot a graph for f(n) and c\*g(n) where c is a constant and for varying n values (10 to 30).

b) Three users in an online music portal listen to a playlist of 8 songs that are numbered from 1 to 8 in a random order. Each user needs to be recommended to another user playlist’s order that has minimum number of inversions. Design and implement an algorithm to determine the number of inversions. State the design strategy used and time complexity of the same.

Ans.

1. Program:

print("Your function is 4n+3") print ("g(n) = n ") print("Assuming c as 5")

for i in range (30):

a1 = 4\*i+3 a2 = 5\*i

if (a2>=a1):

n0 = i break

print("Value of n0: ", n0)

print ("Value\t\tF(n)\t\tc\*G(n)") for i in range (10,31):

print (i,"\t\t",4\*i+3,"\t\t",5\*i)

1. Algorithm:

Sort\_And\_Count(A[0,1,.......,n-1])

//Input: An array A of orderable elements

//Output: A sorted in non-decreasing order and number of inversions

if n>1

copy A[0,1,..........,[n/2]-1] to B[0,1,......,[n/2]-1]

copy A[[n/2]-1,.....,n-1] to C[0,1,......,[n/2]-1]

Mergesort (B[0,.....,[n/2]-1])

Mergesort (C[0,.....,[n/2]-1]) Merge(B,C,A)

Merge\_and\_count (B[0,.......,p-1], C[0,......,q-1], A[0,.........,p+q-1] )

//Input: Arrays B[0,....,p-1] and C[0,........,q-1] btoh sorted

//Output: Sorted array A[0,....,p+q-1] of the elements of B and C

I← 0; j← 0; k←0;count←0 While i < p and j < q do

If B[i]<=C[j]

A[k] ← B[i]; i ← i+1

Else

Endif Endwhile

If i=p then

A[k] ← C[j]; j ← j+1;count ← count + (p-i)

Else

Copy C[j,.....,q-1] to A[k,......,p+q-1]

Copy B[i,......,p-1] to A[k,........,p+q-1]

Program:

# import random import time count = 0

def merge\_sort(li):

if len(li) < 2: return li m = len(li) // 2

return merge(merge\_sort(li[:m]), merge\_sort(li[m:]))

def merge(l, r): global count result = []

i = j = 0

while i < len(l) and j < len(r): if l[i] < r[j]:

result.append(l[i])

i += 1

else:

result.append(r[j])

count = count + (len(l) - i) j += 1

result.extend(l[i:]) result.extend(r[j:]) return result

n = int(input("Enter number of elements in the list")) unsorted = [random.randint(0,100) for i in range(0,n)] print("Unsorted list" , unsorted)

start = time.perf\_counter()

print ("Sorted List" , merge\_sort(unsorted)) stop = time.perf\_counter()

print ("Number of inversions = " ,count , "\nTime taken", stop-start )

1. a) Given f(n)= 2n+3n+5, Write a program to prove that f(n)= Ω(n) and find the n0 value. Plot a graph for f(n) and c\*g(n) where c is a constant and for varying n values (10 to 30).

b) In a database of numbers there is a table of unsorted numbers. The database admin now wants to sort these numbers using an approach wherein a pivot element is selected for sorting. At certain point, the first half elements are less than the pivot and right half elements are greater than the pivot. Design and implement an algorithm to solve it using random numbers and also display the execution time. State the design strategy used and time complexity of the same.

Ans.

1. Program:

print ("The function is 2n^2 + 3n + 5") print ("G(n) = n ")

print ("Assuming c as 9") for i in range (30):

a1 = 2\*(i\*\*2)+3\*i+5 a2 = 19\*i

if (a1>=a2):

n0 = i break

print ("The value of n0: ", n0) print ("n \t\t F(n) \t\t c\*G(n)") for i in range (10,31):

a1 = 2\*(i\*\*2)+3\*i+5

a2 = 9\*i

print (i,"\t\t",a1,"\t\t",a2)

# Algorithm:

//Input: Array A=[0,1,...........,n-1] and the index of first and last element

//Output: Sorted array A

Quicksort(l,r)

If l<r then

mid ← partition(l,r) quicksort (l,mid-1) quicksort (mid+1,r)

partition (l,r)

pivot ← A[l] i ← l+1

j ← r

while True do

while i<=j and A[i]<= pivot do i← i+1

end while

while i<=j and A[j]>=pivot do j← j-1

end while if i>j then

Break from the while

else

Swap A[i] and A[j]

end if end while

swap A[l] and A[j] return j

Program:

# import random import time

def partition(l,r): global a pivot=a[l] i=l+1

j=r

while True:

while (i<=j and a[i]<=pivot): i=i+1

while (i<=j and a[j]>=pivot): j=j-1

if i>j:

break else:

a[i], a[j] = a[j], a[i]

a[l],a[j] = a[j], a[l] return j

def quicksort(l,r): global a

if l<r:

mid=partition(l,r) quicksort(l,mid-1) quicksort(mid+1,r)

m= int(input("Enter the number of elements in the list: ")) a = [random.randint(0,100) for i in range(0,m)]

print(a) start=time.perf\_counter() quicksort(0,len(a)-1)

stop =time.perf\_counter() print(a)

print(stop-start)

1. a) Given f(n)= 8n2+3n+3, Write a program to prove that f(n)= O(n2 ) and find the n0 value. Plot a graph for f(n) and c\*g(n) where c is a constant and for varying n values (10 to 30).

b) A truck driver is given a set of locations to be covered with their distances by a company. The company strictly orders that truck should be started from a particular location. Design and implement an algorithm that gives a greedy solution to the truck driver’s problem and display the shortest path for a given source location to all other locations. State the design strategy used and time complexity of the same.

Ans.

1. Program:

print("Your function is 8n^2+3n+3") print ("g(n) = n^2 ") print("Assuming c as 9")

for i in range (30):

a1 = 8\*(i\*\*2)+3\*i+3 a2 = 9\*(i\*\*2)

if (a2>=a1):

n0 = i break

print("Value of n0: ", n0)

print ("Value\t\tF(n)\t\tc\*G(n)") for i in range (10,31):

print (i,"\t\t",8\*(i\*\*2)+3\*i+3,"\t\t",9\*(i\*\*2))

1. Algorithm: Dijkstra’s Algorithm (G,l)

For each u∊

Let S be the set of explored nodes

S we store a distance d(u) Initially S={s} and d(s)=0

Select a node v ∉ S with at least one edge from S for which

While S != V

∈

d’(v) = min [e=(u,v):u S] d(u) + le is as small as possible Add v to S and define d(v)=d’(v)

End While Program:

# import heapq as hq

def Dijk(graph,start): n = len(graph)

Q = [[0, start]] #print(Q)

d = [999 for i in range(n)] #print(d)

d[start]=0 while Q:

[length, u] = hq.heappop(Q) for v in range(n):

if d[v] > d[u] + graph[u][v]:

d[v] = d[u] + graph[u][v]

hq.heappush(Q, [d[v], v]) return d

#graph = [[0, 5, 10, 999], [5, 0 ,4, 11],[10, 4, 0, 5], [999, 11, 5,0]]

graph = []

n = int(input("Enter number of nodes")) print("Enter the weights of respective edges") for i in range(0,n):

m= []

print( "Next edge . . .") for k in range(0,n):

print( "From ", i+1 ," To ", k+1) val= int(input()) m.append(val)

graph.append(m) d = Dijk(graph,0) print (d)

1. a) Given f(n)= 8n2+3n+3, Write a program to prove that f(n)= Ω(n) and find the n0 value. Plot a graph for f(n) and c\*g(n) where c is a constant and for varying n values (10 to 30).

b) A car driver is given a set of locations to be covered with their distances by a company. Now the company gives a privilege for the car driver to start at any arbitrary location. But, the condition is the route chosen by the driver should be minimum i.e. the total cost of the entire driving should be minimum. Design and implement an algorithm that gives a greedy solution to the car driver and display the minimum cost achieved. State the design strategy used and time complexity of the same.

Ans.

a. Program:

print("Your function is 8n^2+3n+3") print ("g(n) = n^2 ") print("Assuming c as 7")

for i in range (30):

a1 = 8\*(i\*\*2)+3\*i+3 a2 = 7\*(i\*\*2)

if (a1>=a2):

n0 = i break

print("Value of n0: ", n0)

print ("Value\t\tF(n)\t\tc\*G(n)") for i in range (10,31):

print (i,"\t\t",8\*(i\*\*2)+3\*i+3,"\t\t",7\*(i\*\*2))

b) Algorithm:

Initially S = {s} and T = ∅

Prim’s Algorithm(G,c)

Select a node v∉S which has an edge into S and for which the Add edge e where the minimum is obtained to T

While S != V

Add v to S

Attachment cost min [e=(u,v):u∈S] ce is as small as possible

EndWhile

Return the spanning tree T Program:

# def findmin():

minn = 9999

for i in disc:

for j in range(v):

if weights[i][j] < minn: v1=i

v2=j minn=weights[i][j]

if v1 in disc and v2 in disc: weights[v1][v2]=weights[v2][v1]=999 return findmin()

else:

print("Edge taken = ",v1,v2) if v1 in disc:

disc.append(v2)

else:

disc.append(v1)

return weights[v1][v2]

v = int(input("Enter number of vertices")) e = int(input("Enter number of edges")) s = int(input("Enter starting node"))

disc = [s]

weights = [[999]\*v for i in range(v)] for i in range(e):

v1=int(input("Enter head")) v2=int(input("Enter tail")) w=int(input("Enter weight")) weights[v1][v2]=weights[v2][v1]=w

count = 0

while len(disc) < v :

count = count + findmin() print("Weight of mst = ",count)

1. a) Given f(n)= 6n2+2n+2, Write a program to prove that f(n)= O(n2 ) and find the n0 value. Plot a graph for f(n) and c\*g(n) where c is a constant and for varying n values (10 to 30).

b) A phone company wants to lay lines for communication in a city. Different amounts are charged for connecting between a pair of cities. Design and implement a greedy solution such that it forms a spanning tree with minimum cost. State the design strategy used and time complexity of the same.

Ans.

a. Program:

print("Your function is 6n^2+2n+2") print ("g(n) = n^2 ")

print("Assuming c as 7") for i in range (30):

a1 = 6\*(i\*\*2)+2\*i+2 a2 = 7\*(i\*\*2)

if (a2>=a1):

n0 = i break

print("Value of n0: ", n0)

print ("Value\t\tF(n)\t\tc\*G(n)") for i in range (10,31):

print (i,"\t\t",6\*(i\*\*2)+2\*i+2,"\t\t",7\*(i\*\*2))

b) Algorithm:

Kruskal’s algorithm(G,c)

∅

Sort the edges in order of increasing cost Initially T=

For each edge c = (v,w) in the sorted order

If there is currently no path from v to w in (V,T) then (Adding w won’t create a cycle) then

Program: edges = [] dic = [] def next():

# global f if f >=e:

return 0 v1=edges[f][0] v2=edges[f][1] w=edges[f][2] f = f+1

if v1 in dic and v2 in dic: return next()

else:

if v1 in dic:

dic.append(v2) print("Edge taken", v1,v2)

else:

dic.append(v1) print("Edge taken" ,v1,v2)

return w

n = int(input("Enter number of vertice")) e = int(input("Enter number of edge"))

for j in range(e): v1=int(input("Enter one node"))

v2=int(input("ENter other node")) w=int(input("Enter weight")) edges.append([v1,v2,w])

edges.sort(key=lambda x: x[2],reverse=False) count = 0

f=0

dic.append(edges[0][0]) #one vertex of first edge while len(dic) < n:

count = count + next()

1. a) Given f(n)= 4n2+3n+5, Write a program to prove that f(n)= θ(n2 ) and find the n0 value. Plot a graph for f(n), c1\*g(n)and c2\*g(n) where c1, c2is a constant and for varying n values (10 to 30).

b) Alia is planning for a trekking expedition with a backpack that can hold 7kg. She needs to select the most valuable items from the following list that can be accommodated within the backpack. Design and implement an algorithm that displays the most valuable items that can be carried by him using Dynamic programming principles. State the design strategy used and time complexity of the same.

Ans.

a. Program:

print ("The function is 4n^2 + 3n + 5") print ("G(n) = n^2 ")

print ("Assuming c1 as 3 and c2 as 5") for i in range (30):

a1 = 4\*(i\*\*2)+3\*i+5

a2 = 3\*(i\*\*2) a3 = 5\*(i\*\*2)

if a1>=a2 and a3>=a1: n0 = i

break

print ("The value of n0: ", n0)

print ("n \t\t F(n) \t\t c1\*G(n) \t\t c2\*G(n)") for i in range (10,31):

a1 = 4\*(i\*\*2)+3\*i+5 a2 = 3\*(i\*\*2)

a3 = 5\*(i\*\*2)

print (i,"\t\t",a1,"\t\t",a2, "\t\t", a3)

b) Algorithm:

//Input: A set of items 1,2,.......,n with w1,.......,wn, and values v1,v2,....,vn with knapsack capacity W

//Output: Max Profit M[n,W]

for w=0 to W

M[0,w] = 0

for i = 0 to n

M[i,0] = 0

for i = 1 to n

for w = 1 to W

if (wi > w)

M[i,w] = M[i-1,w]

endfor

endfor

else

M[i,w] = max (M[i-1,w], vi + M[i-1,w-wi])

return M[n,W] Program:

# n=int(input("enter the number of items")) W=int(input("enter the knapsack capacity")) items=[0]

for i in range(1,n+1): items.append(i)

weights=[0] value=[0]

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

print("enter the weight for item ",i, " : ") wi=int(input())

weights.append(wi)

print("enter the value for item ",i ," : ") vi=int(input())

value.append(vi) print(items) print(weights) print(value)

M = [[0 for i in range(W+1)] for j in range(n+1)] i = 1

while i <= n: x = 1

while x <= W:

if weights[i] > x: M[i][x] = M[i-1][x]

else:

M[i][x] = max(M[i-1][x],value[i]+M[i-1][x-weights[i]]) x = x + 1

i = i + 1 print(M)

print("maximum value= ",M[n][W]) i=n

k=W cont=[]

while i>0 and k>0:

if M[i][k]!=M[i-1][k]:

cont.append(i) k=k-weights[i] i=i-1

else:

i=i-1

print("items in knapsack are: ") print(cont)

1. a) Given f(n)= 5n2+6n+3, Write a program to prove that f(n)= θ(n2 ) and find the n0 value. Plot a graph for f(n), c1\*g(n)and c2\*g(n) where c1, c2is a constant and for varying n values (10 to 30).

b) A drama venue needs to be allocated for different drama school requests such that maximum profit is obtained for the company owning the drama venue. The requests are shown in the table with start–time, finish-time and the amount affordable by the drama school. Design and implement an algorithm such that maximum profit is obtained for the company owning the drama venue using Dynamic programming principles. State the design strategy used and time complexity of the same.

Ans.

1. Program:

print ("The function is 5n^2 + 6n + 3") print ("G(n) = n^2 ")

print ("Assuming c1 as 4 and c2 as 6") for i in range (30):

a1 = 5\*(i\*\*2)+6\*i+3 a2 = 4\*(i\*\*2)

a3 = 6\*(i\*\*2)

if a1>=a2 and a3>=a1: n0 = i

break

print ("The value of n0: ", n0)

print ("n \t\t F(n) \t\t c1\*G(n) \t\t c2\*G(n)") for i in range (10,31):

a1 = 5\*(i\*\*2)+6\*i+3 a2 = 4\*(i\*\*2)

a3 = 6\*(i\*\*2)

print (i,"\t\t",a1,"\t\t",a2, "\t\t", a3)

1. Algorithm:

M-Compute-OPT(j) if j = 0 then

return 0

else if M[j] is not empty then return M[j]

else

endif

Define M[j] = max ( vj + M-Compute-OPT(p(j)), M-Compute-OPT(j-1)) return M[j]

Find-Solution(j) if j>0 then

if (vj + M[p(j)]) > M[j-1]

print j

Find-Solution(p(j))

else

Program:

# M = []

Find-Solution(j-1)

# Schedule = []

def maximum(a,b): if a > b :

return a else:

return b

def calculate\_predecessor(jobs,n): p = [0 for i in range(n+1)] cur\_job = n

chosen\_job = cur\_job - 1 while cur\_job > 1 :

if chosen\_job <= 0 : p[cur\_job] = 0 cur\_job=cur\_job-1 chosen\_job=cur\_job-1

else:

if jobs[cur\_job][0] < jobs[chosen\_job][1]: chosen\_job = chosen\_job - 1

else:

p[cur\_job] = chosen\_job cur\_job = cur\_job-1

chosen\_job = cur\_job -1 return p

def opt(j,jobs,p): global M

if j == 0:

return M[j] elif j ==1:

M[j] = maximum(jobs[j][2],0) return M[j]

else:

if M[j] == -1:

M[j] = maximum(opt(j-1,jobs,p),jobs[j][2]+opt(p[j],jobs,p)) return M[j]

def wis(jobs,n):

p = calculate\_predecessor(jobs,n) value = opt(n,jobs,p)

return value,p

def find\_solution(j,jobs,p): global M

global Schedule if j > 0 :

if jobs[j][2] + M[p[j]] >= M[j-1]: Schedule.append(j) find\_solution(p[j],jobs,p)

else:

find\_solution(j-1,jobs,p) return

def main():

n = int(input("Enter the number of jobs: ")) global M

M = [-1 for i in range(n+1)]

M[0] = 0

jobs = [0]

for i in range(n):

s = int(input("Start time: ")) f = int(input("Finish time: ")) v = int(input("Value: "))

jobs.append((s,f,v)) max\_value,p = wis(jobs,n)

print(M) print(max\_value) global Schedule find\_solution(n,jobs,p) print(Schedule)

return

main()

1. a) Design and implement bubble sort algorithm to sort a set of numbers. State the design strategy used and time complexity of the same.

b) Design and implement Bellman ford algorithm to find the shortest path from a given source to all other nodes using dynamic programming. State the design strategy used and time complexity of the same.

Ans.

1. Algorithm:

bubble\_sort(A[0,........,n-1]) for i ← 0 to n-1

for j ← 0 to n-i-2

if A[j] > A[j+1]

swap A[j] and A[i]

endfor

endfor

endif

Program:

def bubbleSort(arr): n = len(arr)

for i in range(n):

for j in range(0, n-i-1): if arr[j] > arr[j+1] :

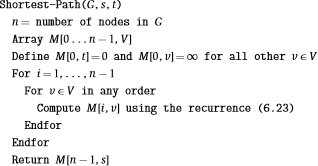
arr[j], arr[j+1] = arr[j+1], arr[j]

import random,time alist=[random.randint(0,1000) for i in range(1000)] start=time.time()

bubbleSort(alist)

end=time.time() print(end-start,"Seconds") print(alist)

1. Algorithm:



Program:

# def bf(adj\_list,n,s):

distance = [999 for i in range(n+1)] distance[s] = 0

for i in range(n-1): for pair in adj\_list:

distance[pair[1]] = min(distance[pair[1]],distance[pair[0]]+pair[2])

del(distance[0]) print(distance)

adj\_list= []

n=int(input("enter the number of vertices")) e=int(input("enter the number of edges")) for i in range(e):

u = int(input("Enter the vertex u: ")) v = int(input("Enter the vertex v: "))

w = int(input("Enter the corresponding weights: ")) l=[u,v,w]

adj\_list.append(l)

print(adj\_list)

source = int(input("Enter a source:")) bf(adj\_list,n,source)

1. a) Design and implement selection sort algorithm to sort a set of numbers. State the design strategy used and time complexity of the same.

b) Design and implement an algorithm for Travelling salesman problem using Branch and bound technique.

Ans.

1. Algorithm

selection\_sort(A[0,1,......,n-1]) for i ← 0 to n-2

min ← i

for j ← i+1 to n-1

if A[j] < A[min] then

min ← j

end if

endfor

swap A[i] and A[min]

end for

Program

# def sort(L):

for i in range(len(L)-1): min=i

for j in range(i+1,len(L)): if L[j]<L[min]:

min=j

L[i],L[min] = L[min],L[i]

import random,time

alist=[random.randint(0,1000) for i in range(1000)] start=time.time()

sort(alist) end=time.time()

print(end-start,"Seconds") print(alist)

1. Algorithm:

Program

#include<stdio.h>

int matrix[25][25], visited\_cities[10], limit, cost = 0; int tsp(int c)

{

int count, nearest\_city = 999; int minimum = 999, temp;

for(count = 0; count < limit; count++)

{

if((matrix[c][count] != 0) && (visited\_cities[count] == 0))

{

if(matrix[c][count] < minimum)

{

minimum = matrix[count][0] + matrix[c][count];

}

temp = matrix[c][count]; nearest\_city = count;

}

}

if(minimum != 999)

{

cost = cost + temp;

}

return nearest\_city;

}

void minimum\_cost(int city)

{

int nearest\_city; visited\_cities[city] = 1; printf("%d ", city + 1); nearest\_city = tsp(city); if(nearest\_city == 999)

{

nearest\_city = 0;

printf("%d", nearest\_city + 1);

cost = cost + matrix[city][nearest\_city]; return;

}

minimum\_cost(nearest\_city);

}

int main()

{

int i, j;

printf("Enter Total Number of Cities:\t"); scanf("%d", &limit);

printf("\nEnter Cost Matrix\n"); for(i = 0; i < limit; i++)

{

printf("\nEnter %d Elements in Row[%d]\n", limit, i + 1); for(j = 0; j < limit; j++)

{

scanf("%d", &matrix[i][j]);

}

visited\_cities[i] = 0;

}

printf("\nEntered Cost Matrix\n"); for(i = 0; i < limit; i++)

{

printf("\n");

for(j = 0; j < limit; j++)

{

printf("%d ", matrix[i][j]);

}

}

printf("\n\nPath:\t"); minimum\_cost(0); printf("\n\nMinimum Cost: \t"); printf("%d\n", cost);

return 0;

}