

BRAC University

Assignment No: 01

Course Code: CSE221

Semester: Summer 2023

Student Name: Sumaiya Azad Isha

ID: 22101503

Section: 15

Date of Submission: 7th July 2023

9.1

a. The given & code & snippet can no has undetermined time complexity. The inner loop will run infinitely times, so, the time complexity is infinity.

b. The time complexity of this code is also infinity. Because the 2nd part of the code will run infink time.

C. The sa statement is meaningless because doesn't give any details about the algorithm's performance. The work 'at least' means performance bound of the algorithm. Big-oh the lower bound of the algorithm. Big-oh notation means the upper bound. notation means the upper bound. so the statement doesn't make any sense.

Subject:

Date:

Q. 2 sty the exact was a rusult of the S.D

details = [[id, name],...,n]

the senting adjust algorithmos over; mostring

moderning = = details [] [] [] Tom to two neturn Trul Lines trong to

return false.

sonord northogollo promom fraistherni

b. countsort can be used here. or crash.

Algorithmo

Countsont (A, Output,

C = [0] * max [A] -> new array; output = [0] * len (A)

forc ? -> 0 to len(A): (Add) tracquem too c[A[i]] += 1

for j → o to len(c): :(N(1) +)

c[3] += c[j-i] s/(n+1) = bim

for K > len(A)-1 to -1

output [C[A[K]]-1] = A[K] = A]

C[A[K]] = C[A[K]-]

netures output (men) () And

-slinks

Time :

E. Yes, the creash was a result of the sonting algorithm. It is possible that the sonting algorithm was running out of memory if the maximum number of post count is very big. Due to insufficient memory allocation there will a creash.

d. Merge Sont has a time complexity of nlogn in best, worst and average case.

def mergesont (A, l, h):

lf (l>h):

mid = (1+h)//2

mergesorit (1

la = mergesort (1, mid A, 1, mid)

RA= merge sort (A, mid +1, h)

merge (IA, DRA, n, m):

mA= [o] (m+n)

While (

det menge (alA, nA, n, m): Date: m A=[0]* (m+n) Time: while (i'ch and jcm): Algorithm; if IA[i] < RA[j]: Det Binary Search [A [X] A] = + [X] Amthe target Value 1+=1 mA[k] = RA[j] 1-(A) mal = d else: while (LC=N): K+=1 511 (N+1) = bim while izn: thon # if x is [i] AJE ; Exj Am the left i+=1, Kt=1[bim]A Bon +; while JEm: I + bern = 1 them art MA[K] = TA[S] x 18 4 j+=1, K+=1 [bim] A tilo return mA 1-bim = i cloc: ME terror worked "Found" "Grant for " router # if me the element is not prese in the list. Friendship

C. The searching algorithm is Binary Search.

Algorithm:

Def Binary Search (A, X): # X is the target Value

: [6] AN > [7] AN 7

L=0

h = len (A)-1 [a] An = [a] Am

while (L Z=h):

mid=(1+h)/12 : 12 3/20

if x is greater, ignore the left half

if read A[mid] < X > (1 =+)

1 = mid + 1: msc status

if x is smaller, ig note the right half

elif A[mid] >n:

h=mid-1 Am mutin

else:

return mand "Found"

neturn "Not found" # if no the element is not present in the list.

Steps to search for post count of since the list is already sorted, the mid element is 5.

- do the searching to the right half ([8,9,12,14,17])
- The mid element of the rught half is
- in left half: [8,2]
- in the right half: [5] when he will search
- -> The mid element is 9. We found the target.

it are any [mid] cala.

elif array [mid] >cle:

1-bim =n

endship

Subject:

```
teps to escerch for post count of
1d The modified algorithm algorithm is:
def b-search (array; ele): a i tromas
    L=0, h= len(array)-1 notoone
   while (L <=h):
      mid= (L+h)112 ort of prinsings of ut ob
      if array [mid] < ele: ([FIII] (SI (C) 8])
         l=m-id+1
 the elif armay [mid] releises bim ont
         ah = mid-1
      else:
         neturin mid most makes of si
                     12++ WIF: 18,2
    return -1
```

2. To return the first index oif there are dupliates. det b-search (array, ele); that their out oil 1,=0, h= len(array)-1, pindx=-1 while (LEh):

mid=(l+h)112 it arcray [mid] Lele: l= mid+1

elif array [mid] >ele: h= mid-1

Date:

else:

indx = mid

return Fabre

S11(N+L) = 60m

: 2019

h= mid-1

return (findex, ocust)

return indx

3. To neturen index and first appearance:

det b-search (array, ele) une se mim-brit tob

1=0, h = len(array)-1, f-index= -1 0=1 while ILC=h)

while (l2=h):

mid= (1+h) 112

if array [mid] * cele :== bim) +i

bing pour = mild+pure | mild+pure | bing | bins

elif array [mid] > ele: h= mid-1/bim/pouro routini

else: Wind principle brus (ox Birm tills f-index = mid - pin - -

h=mid-1

Etemp = findex

while (temp < len(annay) and annay [temp] == ele): temp += 1 [bim] ponne number

count = temp - f-index

Date: Time:

Subject: if f_index ==-1: bim = xbri return False 1-bim = 1 return (f-index, count) netern indx 4. To find minimum element wave pattern.

3. To neturn index and first det find-min-ele (array): pomos) domosed tob 1=0, h=len (array)-1- (porus) no) = 1,0=1

while (12=h):

mid = (l+h) 112 (N+1) = bim if [mid == 0 on min array [mid-i] > array [mid] and (mid = per len(array)-1 on array [mid+1] >array [mid]);

while (LC= H):

return array [mid]

elif (mid > 0) and (array [min -i] zarray [mid]):

h = m - mid - 1 1-bim=d

else:

l= mid+1

while (temp & 10n (and return array [mid] 1 =+ quist

occurt = temp - 1-index

xobai-t = qanste

Date :

6.8.4:

For the given list if we search for T=2

the steps will be:

11 11 11 11 11 11			MOUNT - PULL - UNG - MOLK
midele	L_indx	R-indx	t=0, h= lentarua
	0	7	solvile (LC=K):
3	0	2	mid= (1+1)/12
(Janus)	ons (Li-bi	m Jaruas	it (mid= (1+1)/112
my Jane	· · · · · · · · · · · · · · · · · · ·		
& fou	nd T	[pimlhr	neturn aren

conincidently we have found the value T= 2
but if the value was 23 on 10 on any
1 on anything we would not be able to
find it because the annay list is not
sorted.

ei shos civil to prixalquios anit aut d

· (NBOD) O

Date: Subject: Using Binary Search we can solve it. def b-search-find-max (array): l=0, h= len(array)-1 while (LZ=h): mid= (1+h)1/2 if (array [mid]) array [mid-i]) and (array [mid]) array [mid+1]); ruturen armay[mid] elif (array [mid] > array [mid-i]) and farray [mid] else à ton bluou ou print pro tind it be cause the return & max (array [1], array [h]) b. The time complexity of this code is O (logN).

ton ? in range (1, max ele, - mm_e le 3, 10 ; 3)

a Sonting an armay before binary search is useful because to when we need to search for a multiple times. Sonting the armay takes O(NlogN) time (only once). Then we can search for the element in O(logN) time. But if we do linear search it will take O(N) time for each search which is not efficient way to do the searching.

b. Modified countsont algorithm:

def count sort (array):

find the max, min of the array

min_ele = min (array), max-ele = max (array

count = [0]* (max-ele - min_ele + 1)

for vail in array:

count [val - min_ele] += 1;

Subject :

Date :

for i in range (1, max-ele, min-ele +1):

of output = [0] * len (arrivay)

fon j'in range (tenlarray)-1,-1,-1):

output [count [array[j]]-min_ele]-1] = curray[j]

count [array [5] - min-ele] - = 1

take O(N) time for each search which property to do the searching

C. Same as the code word in (b).

Here we'll oust take the int value using

the int () function: (pours) trise truss tob

det count_sont (annay); rim, xom est brit #

max = int (max(array)), min = int(min(array))

count = [0]* (max - min+1)

fore val in array some ni lor not

count [int(val) = min) + = 1+muss

iendship

Date :

time when +

fore i in range (1, max-min+1): coun+[i] += count[i-1]

output = [o] * len (array)

for j in range (len (array)-1,-1,-1):

output [count [int (array [j]) - man]-1] =

array [j]

count [int (array [j]) - min] -= 1

return output

d. Suick Sort is faster than merge sort when the pivot is chosen wisely. If the the pivot is chosen poorly suick sort will have the worst-case time complexity of $O(n^2)$. Merge sont work in $O(N\log N)$ time complexity in every case. If we want to run the code at $O(n\log n)$ time for any case then mergesont is a better option.