Experiment #1		Student ID	
Date		Student Name	

Experiment Title: Analysis of Algorithm based on Arrays

Aim/Objective: To understand the concept and implementation of Basic programs on arrays.

Description: The students will understand and able to implement programs on Arrays.

Pre-Requisites:

Knowledge: Arrays

Tools: Code Blocks/Eclipse IDE

Pre-Lab:
Calculate the time complexity of the following:

1. Algorithm linear_search(arr, target):

```
for i in range(len(arr)):
    if arr[i] == target:
        return i
return -1
```

- **Procedure/Program:**

```
#include <stdio.h>
int linear search (int arr[], int size, int target)
{
for(int i=0 ; i<size; i++) {-  -(n+1)
  if(arr[i] == target ) ——
  {
  return i; --
                          (n+1) + n + 1
                          } n+1
                            O(n)
```

- **Data and Results:** let arr size is(n) the
Best case:- the element is the first element in arr
Time complexity O(1)
worst case:- The element is not present in the arr
Time complexity O(n) avg:- O(n)

- **Analysis and Inferences:** The linear search of an algorithm each element in the array until it finds the target element or searches
Best case:- O(1) ;- worst case:- O(n)

Scanned with OKEN Scanner

Experiment #1		Student ID	
Date		Student Name	

2. Algorithm function(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]
return arr;
```

- **Procedure/Program:**

```
no.of bubble-sort(intarr[], inFor)
For(int i=0; i<n-1; i++) & (n+1)
For(int j=0; j<n-i-1; j++) & n (n+1)
  if(arr[j] > arr[j+1]) & n-1
    int temp= arr[j] >n-1
     arr[j ]=arr[j+1];n-1
    arr[j+1]=temp; }}
```

- **Data and Results:** let input array [13,12,5,12,22,11]

```
step-1:[25,13,11,12,22]
step-2[25,12,13,11,22]
step-3[12,25,11,13,22]
step-4[12,11,20,22,34]
step-5[11,12,22,25,34]
```

- **Analysis and Inferences:**

1. Best case:- The array is already sort make a Passes but no swap Performed O(n)
   Worst case:- it make n-i-1 comparises this result is Fine completely O(n²)

Scanned with OKEN Scanner

Experiment #1		Student ID	
Date		Student Name	

3. The median of a list of numbers is essentially its middle element after sorting. The same numb
of elements occur after it as before. Given a list of numbers with an odd number of elements, fi
the median? Also find its time complexity.

Example arr=[5,3,1,2,4]

The sorted array arr=[1,2,3,4,5]. The middle element and the median is 3.

Description: Write an algorithm findMedian with the following parameter(s):

arr[n]: an unsorted array of integers

Returns the median of the array

Sample Input

7

0 1 2 4 6 5 3

Sample Output

3

- **Procedure/Program:**

```
int compare(const void*a, const void*b)
{return (*(int*a-)*(int*)b);
int findMedian(int arr[], int n) {
    qsort(arr, n, size of(int), compare)
    int mid = n/2;
    return arr(mid);
}
```

- **Data and Results:**

let array [ ]

median=

- **Analysis and Inferences:** The most time-complexity of an algorithm. The time complexity = O(n log n)

Course Title	Design and Analysis of Algorithms	ACADEMIC YEAR: 2024-25
Course Code(s)	23CS2205R	Page 3 of 93

Experiment #1	
Date	

In-Lab:

1. Given an ar
each string,
in alphabeti
Input: arr[
Output: {
Explanat
order.
Find the

- **Proc**

Data

int cor
{

Scanned with OKEN Scanner

ment after sorting. The same numbe
with an odd number of elements, find

the median is 3.

wing parameter(s):

const void*b)
a-) (int*) b);
],int n) {
=(int), compare)
);

of an
ogn)

Experiment #1		Student ID	
Date		Student Name	

In-Lab:

1. Given an array of strings arr[], the task is to sort the array of strings according to frequency of each string, in ascending order. If two elements have same frequency, then they are to be sorted in alphabetical order.

   Input: arr[] = {"Ramesh", "Mahesh", "Mahesh", "Ramesh"}

   Output: {"Mahesh", "Ramesh"}

   Explanation: As both the strings have the same frequency, the array is sorted in alphabetical order.

   Find the time and space complexity for the procedure/Program.

- **Procedure/Program:**

```
Data struct {
    char str[100];
     int freq; }
     string freq;
int compare (const void *a, const void *b)
  {
    string freq *strA = (stringFreq*)a;
    string freq* strB = (stringFreq*)b;
     if (strA->freq == strB->freq){
     return strcmp(a->str, strB->str)
      }
    return strA->freq->freq-strB->freqB
```

```
void sortbyfrequency (char *
                        a[], n)
  {
string frequency[100];
int Freq count = 0;
  int found;
  For(int i=0; i<n; i++)
   {
   found=0;
for(int j=0; j<Freq; j++)
   {
 if(strcmp(Freq[j]str
    ar[i]) =0) {
    Freq[j] Feq++
    found= 1;
    break
  }}
```

- **Data and Results:**

```
{ Ramesh; 2; Mahesh; 2)
{ (Mahesh; 2), (. Rmesh; 2))
sort array [mahesh, Ramesh]
```


Scanned with OKEN Scanner

Experiment #1		Student ID	
Date		Student Name	

- **Analysis and Inferences:**

2. Given an array of strings words [] and the sequential order of alphabets, our task is to sort the ar according to the order given. Assume that the dictionary and the words only contain lowerc alphabets.

Input: words = {"word", "world", "row"},

Order= "worldabcefghijkmnpqstuvxyz"

Output: {"world", "word", "row"}

Explanation: According to the given order 'l' occurs before 'd' hence the words "world" will kept first.

Find the time and space complexity for the procedure/Program.

- **Procedure/Program:**

```
int compare(const void *a, const void *b,
            void *order_map) {
stringorder *strA = (stringorder *) a;
stringorder *strB = (stringorder *) b;
    char *word1 = strA -> str
    char *order = (int *) order_map;
    int len1 = strlen (word1);
    int len2 = strlen (word2);
for(int i=0 ; i<len1 || i<len2 ;
 if(word1[i] != word2[i
return order[(int) word1
-order[int word2[i]
    }
}
return len1 - len2;
}
```

- **Data and Results:**

words = {"word", "world", "row"}

order: "worldabcefghijkmnpqstuvwxyz"

{w:0, o:1, r:2, l:3, d:4, a:5, b:6, c:7, e:8, f:9 ---}

sorted word = ["world", "word", "row"]

Scanned with OKEN Scanner

Experiment #1		Student ID	
Date		Student Name	

- **Analysis and Inferences:**

creating ~~order~~map $O(16) = O(1)$ these are 26 characters in the alph

The comple is $O(m \log n)$ where m is the number of words

space of complexity $O(16) = O(1)$

$S(n)$

Post-Lab:

Evaluate the time complexity of pre-lab using **master's theorem**.

- **Procedure/Program:**

$T(n) = aT(n/n) + f(n)$ $T = 3T(n/2) + n^2$

where

$a = 3$

$n/b = n/2$

$f(n) = n$

n = size of an input

a = number of each supportable, All sub Problems are assumed to have the same size

$f(n)$ = cost of the work done outside the recursive call which including the cost of dividing the Problem

$a \geq 1$ and $b > 1$ are constants

and $f(n)$ is as asymptotically for

- **Data and Results:**

If $f(n) = O(n^{\log_b a - \varepsilon})$ then

$T(n) = O(n^{\log_b a})$

Scanned with OKEN Scanner

Experiment #1		Student ID	
Date		Student Name	

∈ $F(n) = \theta(n \log_b a)$ then $T(n) = \theta(n \log a * \log n)$
$f(n) = a(n \log_b a + c)$ then $T(n) = \theta(F(n))$

- **Analysis and Inferences:**

n = size of an input
a = number of each suppable subproblems
here a = 1 and b = 1 are constants and
f(n) is a simtfically

- **Sample VIVA-VOCE Questions (In-Lab):**
  1. What is the significance of analyzing both time complexity and space complexity whe evaluating algorithms?
  2. How does the choice of data structures impact both time and space complexity in algorithr implementations? Give examples.
  3. When comparing two algorithms with different time and space complexities, how do yo decide which one is more suitable for a particular problem or application?
  4. What is the difference between best-case, average-case, and worst-case time complexity Provide examples.
  5. In the context of sorting algorithms, compare the time and space complexities of algorithm like Quick Sort and Merge Sort. Which one is more time-efficient, and which one is mor space-efficient?

Evaluator Remark (if Any):	Marks Secured: 42 out of 50
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

Scanned with OKEN Scanner

Student ID
ent Name

T(n) = O(n log a * log
c) then T(n) = O(F(n)g(n))

suppable subproblems
constants and
ally

mplexity and space complexity wh
ne and space complexity in algorith
d space complexities, how do yo
m or application?
and worst-case time complexity
space complexities of algorithms
fficient, and which one is mor

of 50

Sign Date

each experiment.

MIC YEAR: 2024-25
Page 7 of 93

WEEK-1

① Time complexity of an algorithm the amount we taken for an algorithm and function of time and input.

② Using the hash table can be reduce the average time complity of search operation of complexity O(1) (or) O(n)

③ If two algorithms with the different time and space complexity the using memory and input

④ • Best case:- minimum steps on input data of an nelement.
• Worst case :- the function which perform the minimum numbers of steps input has size
• Average case:- The steps performe the average number of steps of an nelement

Time complexity is Σ(n log n) of a Quick sort of an algorithm

ms
in
d an
olve
that

Scanned with OKEN Scanner