Lab Manual PCS 409

Course Title : Design and Analysis of Algorithms

Week 1:

Note: Input, output format for problem I, II and III is same and is given at the end of this exercise.

I. Given an array of nonnegative integers, design a linear algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for each input case. (Time Complexity = O(n), where n is the size of input)

Sample I/O Problem - 1:

Input:	**Output:**

3
8
34 35 65 31 25 89 64 30
89
5
977 354 244 546 355
244
6
23 64 13 67 43 56
63
	Present 6
	Present 3
	Not Present 6

II. Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether given key element is present in the array or not. Also, find total number of comparisons for each input case. (Time Complexity = O(logn), where n is the size of input).

III. Given an already sorted array of positive integers, design an algorithm and implement it using a program to find whether a given key element is present in the sorted array or not. For an array arr[n], search at the indexes arr[0], arr[2], arr[4],. ,arr[2^k] and so on. Once the interval (arr[2^k] < key < arr[2^{k+1}]) is found, perform a linear search operation from the index 2^k to find the element key. (Complexity < O(n), where n is the number of elements need to be scanned for searching): **Jump Search**

Input format:
The first line contains number of test cases, T.
For each test case, there will be three input lines.
First line contains n (the size of array).
Second line contains n space-separated integers describing array.
Third line contains the key element that need to be searched in the array.

Output format:
The output will have T number of lines.
For each test case, output will be "**Present**" if the key element is found in the array, otherwise "**Not Present**".
Also for each test case output the number of comparisons required to search the key.

Sample I/O Problem - 2, 3:

Input:	**Output:**

3
5
12 23 36 39 41
41
8
21 39 40 45 51 54 68 72
69
10
101 246 438 561 796 896 899 4644 7999 8545
7999
	Present 3
	Not Present 4
	Present 3

Week 2:

I. Given a sorted array of positive integers containing few duplicate elements, design an algorithm and implement it using a program to find whether the given key element is present in the array or not. If present, then also find the number of copies of given key. (Time Complexity = O(log n))

Input format:
The first line contains number of test cases, T.
For each test case, there will be three input lines.
First line contains n (the size of array).
Second line contains space-separated integers describing array.
Third line contains the key element that need to be searched in the array.

Output format:
The output will have T number of lines.
For each test case T, output will be the key element and its number of copies in the array if the key element is present in the array otherwise print " **Key not present**".

Sample I/O Problem I:

Input:	**Output:**

2
10
235 235 278 278 763 764 790 853 981 981
981
15
1 2 2 3 3 5 5 5 25 75 75 75 97 97 97
75
	981 - 2
	75 - 3

II. Given a sorted array of positive integers, design an algorithm and implement it using a program to find three indices i, j, k such that arr[i] + arr[j] = arr[k].

Input format:
The first line contains number of test cases, T.
For each test case, there will be two input lines.
First line contains n (the size of array).
Second line contains space-separated integers describing array.

Output:
The output will have T number of lines.
For each test case T, print the value of i, j and k, if found else print "**No sequence found**".

Sample I/O Problem II:

Input:	Output:

3
5
1 5 84 209 341
10
24 28 48 71 86 89 92 120 194 201
15
64 69 82 95 99 107 113 141 171 350 369 400 511 590 666
	No sequence found.
	2, 7, 8
	1, 6, 9

III. Given an array of nonnegative integers, design an algorithm and a program to count the number of pairs of integers such that their difference is equal to a given key, K.

Input format:
The first line contains number of test cases, T.
For each test case, there will be three input lines.
First line contains n (the size of array).
Second line contains space-separated integers describing array.
Third line contains the key element.

Output format:
The output will have T number of lines.
For each test case T, output will be the total count i.e. number of times such pair exists.

Sample I/O Problem III:

Input:	Output:

2
5
1 51 84 21 31
20
10
24 71 16 92 12 28 48 14 20 22
4
	2
	4

Week 3:

I. Given an unsorted array of integers, design an algorithm and a program to sort the array using insertion sort. Your program should be able to find number of comparisons and shifts (shifts - total number of times the array elements are shifted from their place) required for sorting the array.

Input Format:
The first line contains number of test cases, T.
For each test case, there will be two input lines.
First line contains n (the size of array).
Second line contains space-separated integers describing array.

Output Format:
The output will have T number of lines.
For each test case T, there will be three output lines.
First line will give the sorted array.
Second line will give total number of comparisons.
Third line will give total number of shift operations required.

Sample I/O Problem I:

Input:	**Output:**
3	-31 -23 32 45 46 65 76 89
8	comparisons = 13
-23 65 -31 76 46 89 45 32	shifts = 20
10	21 32 34 46 51 54 65 76 78 97
54 65 34 76 78 97 46 32 51 21	comparisons = 28
15	shifts = 37
63 42 223 645 652 31 324 22 553 -12 54 65 86 46 325	-12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
	comparisons = 54
	shifts = 68

II. Given an unsorted array of integers, design an algorithm and implement a program to sort this array using selection sort. Your program should also find number of comparisons and number of swaps required.

Input Format:
The first line contains number of test cases, T.
For each test case, there will be two input lines.
First line contains n (the size of array).
Second line contains space-separated integers describing array.

Output Format:
The output will have T number of lines.
For each test case T, there will be three output lines.
First line will give the sorted array.
Second line will give total number of comparisons.
Third line will give total number of swaps required.

Sample I/O Problem II:

Input:	**Output:**
3	-21 -13 12 45 46 65 76 89
8	comparisons = 28
-13 65 -21 76 46 89 45 12	swaps = 7
10	21 32 34 46 51 54 65 76 78 97
54 65 34 76 78 97 46 32 51 21	comparisons = 45
15	swaps = 9
63 42 223 645 652 31 324 22 553 12 54 65 86 46 325	12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
	comparisons = 105
	swaps = 14

III. Given an unsorted array of positive integers, design an algorithm and implement it using a program to find whether there are any duplicate elements in the array or not. (use sorting) (Time Complexity = O(n log n))

Input Format:
The first line contains number of test cases, T.
For each test case, there will be two input lines.
First line contains n (the size of array).
Second line contains space-separated integers describing array.

Output Format:

The output will have T number of lines.
For each test case, output will be **'YES'** if duplicates are present otherwise '**NO**'.

Sample I/O Problem III:

Input:	**Output:**

3
5
28 52 83 14 75
10
75 65 1 65 2 6 86 2 75 8
15
75 35 86 57 98 23 73 1 64 8 11 90 61 19 20
	NO
	YES
	NO

Week 4:

I. Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by dividing the array into two subarrays and combining these subarrays after sorting each one of them. Your program should also find number of comparisons and inversions during sorting the array.

Input Format:
The first line contains number of test cases, T.
For each test case, there will be two input lines.
First line contains n (the size of array).
Second line contains space-separated integers describing array.

Output Format:
The output will have T number of lines.
For each test case T, there will be three output lines.
First line will give the sorted array.
Second line will give total number of comparisons.
Third line will give total number of inversions required.

Sample I/O Problem I:

Input:	**Output:**

3
8
23 65 21 76 46 89 45 32
10
54 65 34 76 78 97 46 32 51 21
15
63 42 223 645 652 31 324 22 553 12 54 65 86 46 325
	21 23 32 45 46 65 76 89
	comparisons = 16
	inversions =
	21 32 34 46 51 54 65 76 78 97
	comparisons = 22
	inversions =
	12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
	comparisons = 43
	inversions =

II. Given an unsorted array of integers, design an algorithm and implement it using a program to sort an array of elements by partitioning the array into two subarrays based on a pivot element such that one of the sub array holds values smaller than the pivot element while another subarray holds values greater than the pivot element. Pivot element should be selected randomly from the array. Your program should also find number of comparisons and swaps required for sorting the array.

Input Format:

The first line contains number of test cases, T.
For each test case, there will be two input lines.
First line contains n (the size of array).
Second line contains space-separated integers describing array.

Output Format:
The output will have T number of lines.
For each test case T, there will be three output lines.
First line will give the sorted array.
Second line will give total number of comparisons.
Third line will give total number of swaps required.

Sample I/O Problem II:

Input:	**Output:**

3
8
23 65 21 76 46 89 45 32
10
54 65 34 76 78 97 46 32 51 21
15
63 42 223 645 652 31 324 22 553 12 54 65 86 46 325
	21 23 32 45 46 65 76 89
	comparisons = 14
	swaps = 10
	21 32 34 46 51 54 65 76 78 97
	comparisons = 29
	swaps = 21
	12 22 31 42 46 54 63 65 86 223 324 325 553 645 652
	comparisons = 45
	swaps = 39

III. Given an unsorted array of integers, design an algorithm and implement it using a program to find Kth smallest or largest element in the array. (Worst case Time Complexity = O(n))

Input Format:
The first line contains number of test cases, T.
For each test case, there will be three input lines.
First line contains n (the size of array).
Second line contains space-separated integers describing array.
Third line contains K.

Output Format:
The output will have T number of lines.
For each test case, output will be the Kth smallest or largest array element.
If no Kth element is present, output should be "**not present**".

Sample for Kth smallest:

Input:	**Output:**

3
10
123 656 54 765 344 514 765 34 765 234
3
15
43 64 13 78 864 346 786 456 21 19 8 434 76 270 601
8
	123
	78

Week 5:

I. Given an unsorted array of alphabets containing duplicate elements. Design an algorithm and implement it using a program to find which alphabet has maximum number of occurrences and print it. (Time Complexity = O(n)) (Hint: Use counting sort)

Input Format:
The first line contains number of test cases, T.
For each test case, there will be two input lines.
First line contains n (the size of array).
Second line contains space-separated integers describing array.

Output:
The output will have T number of lines.
For each test case, output will be the array element which has maximum occurrences and its total number of occurrences.
If no duplicates are present (i.e. all the elements occur only once), output should be “**No Duplicates Present**”.

Sample I/O Problem I:

Input:	**Output:**
3 10 a e d w a d q a f p 15 r k p g v y u m q a d j c z e 20 g t l l t c w a w g l c w d s a a v c l	a – 3 No Duplicates Present l - 4

II. Given an unsorted array of integers, design an algorithm and implement it using a program to find whether two elements exist such that their sum is equal to the given key element. (Time Complexity = O(n log n))

Input Format:
The first line contains number of test cases, T.
For each test case, there will be two input lines.
First line contains n (the size of array).
Second line contains space-separated integers describing array.
Third line contains key

Output Format:
The output will have T number of lines.
For each test case, output will be the elements arr[i] and arr[j] such that arr[i]+arr[j] = key if exist otherwise print '**No Such Elements Exist**”.

Sample I/O Problem II:

Input:	**Output:**

2
10
64 28 97 40 12 72 84 24 38 10
50
15
56 10 72 91 29 3 41 45 61 20 11 39 9 12 94
302
	10 40
	No Such Element Exist

III. You have been given two sorted integer arrays of size m and n. Design an algorithm and implement it using a program to find list of elements which are common to both. (Time Complexity = O(m+n))

Input Format:
First line contains m (the size of first array).
Second line contains m space-separated integers describing first array.
Third line contains n (the size of second array).
Fourth line contains n space-separated integers describing second array.

Output Format:
Output will be the list of elements which are common to both.

Sample I/O Problem III:

Input:	**Output:**
7	10 10 34 55
34 76 10 39 85 10 55	
12	
30 55 34 72 10 34 10 89 11 30 69 51	

Note: Consider the following input format in the form of adjacency matrix for graph based questions (directed/undirected/weighted/unweighted graph).

Input Format: Consider example of below given graph in Figure (a).
A boolean matrix AdjM of size V X V is defined to represent edges of the graph. Each edge of graph is represented by two vertices (start vertex u, end vertex v). That means, an edge from u to vis represented by making AdjM[u,v] and AdjM[v,u] = 1. If there is no edge between u and v then it is represented by making AdjM[u,v] = 0. Adjacency matrix representation of below given graph is shown in Figure (b). Hence edges are taken in the form of adjacency matrix from input. In case of weighted graph, an edge from u to v having weight w is represented by making AdjM[u,v] and AdjM[v,u] = w.

Input format for this graph is shown in Figure (c).
First input line will obtain number of vertices V present in graph.
After first line, V input lines are obtained. For each line i in V, it contains V space separated boolean integers representing whether an edge is present between i and all V.

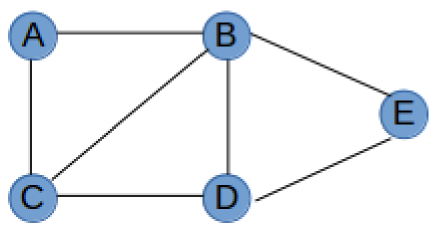
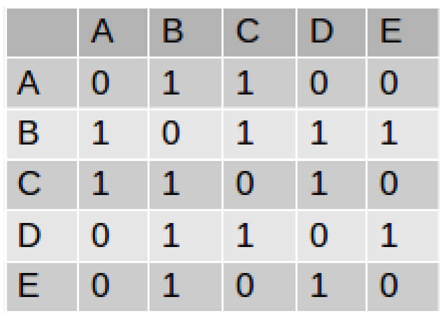
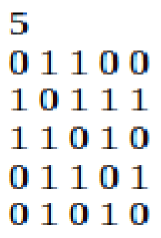


Figure (a)



	A	B	C	D	E
A	0	1	1	0	0
B	1	0	1	1	1
C	1	1	0	1	0
D	0	1	1	0	1
E	0	1	0	1	0

Figure (b)



```
5
0 1 1 0 0
1 0 1 1 1
1 1 0 1 0
0 1 1 0 1
0 1 0 1 0
```

Figure (c)

Week 6: