Lab. Manual Prepared by Prof. Anil Kumar Swain

CONTENTS

LAB - 5
Heap & Priority Queues

(Building a heap, Heap sort algorithm, Min-Priority queue, Max-Priority queue)

PROGRAM EXERCISE

Lab. Exercise (LE)

5.1 Write a menu (given as follows) driven program to sort an array of n integers in ascending order by heap sort algorithm and perform the operations on max heap. Determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the array to be sorted and plot a graph of the time taken versus n. The elements can be read from a file or can be generated using the random number generator.

MAX-HEAP & PRIORITY QUEUE MENU

0. Quit
1. n Random numbers=>Array
2. Display the Array
3. Sort the Array in Ascending Order by using Max-Heap Sort technique
4. Sort the Array in Descending Order by using any algorithm
5. Time Complexity to sort ascending of random data
6. Time Complexity to sort ascending of data already sorted in ascending order
7. Time Complexity to sort ascending of data already sorted in descending order
8. Time Complexity to sort ascending all Cases (Data Ascending, Data in Descending & Random Data) in Tabular form for values n=5000 to 50000, step=5000
9. Extract largest element
10. Replace value at a node with new value
11. Insert a new element
12. Delete an element

Enter your choice:

If the choice is option 8, the it will display the tabular form as follows:

Analysis of Max-Heap Sort Algorithm

Sl. No.	Value of n	Time Complexity (Sorted Data)	Time Complexity (Reversely Sorted Data)	Time Complexity (Random Data)
1	5000			
2	10000			
3	15000			

4	20000			
5	25000			
6	30000			
7	35000			
8	40000			
9	45000			
10	50000			

Home Exercise (HE)

5.2 Similar to above program no.5.1, write a menu driven program to sort an array of n integers in **descending order by heap sort algorithm.** Hints: Use min heap and accordingly change the menu options.

Round Exercise (RE)

5.3 Write a program with the following two functions.

a) MaxMin() - Takes a heap, returns 'MAX' if it is a max-heap and converts it into a mean-heap in linear(to the number of elements) time.

b) MinMax() - Takes a heap, returns 'MIN' if it is a min-heap and converts it into a max-heap in linear (to the number of elements) time.

LAB - 6
Greedy Methods

(Fractional knapsack problem, Activity selection problem, Huffman's code)

PROGRAM EXERCISE

Lab. Exercise (LE)

6.1 Write a program to implementation of **Fractional Knapsack algorithm.**

6.2 Write a program to implement the **activity-selection problem** stated as follows:

You are given n activities with their start and finish times. Select the maximum number of activities that can be performed by a single person, assuming that a person can only work on a single activity at a time. Example: Consider the following 6 activities (0, 1, 2, 3, 4, 5). start[] = {1, 3, 0, 5, 8, 5}; finish[] = {2, 4, 6, 7, 9, 9}; The maximum set of activities that can be executed by a single person is {0, 1, 3, 4}.

6.3 **Huffman coding** assigns variable length codewords to fixed length input characters based on their frequencies/probabilities of occurrence. Given an array of characters along with their frequency of occurrences. Write a menu driven programming to perform the following functions.

HUFFMAN CODE GEBERATION MENU

0. Quit
1. Input n unique characters with their frequencies into an array
2. Display the Array
3. Generate Huffman Tree and Traverse the tree with pre-order.
4. Generate Huffman Codes for n charachers and display the same.
5. Compare Huffman code with Fixed-Length Code

Enter your choice:

Home Exercise (HE)

6.4 You are given **n activities with their start and finish times**, input through keyboard. Write a program to select all possible maximum number of activities that can be performed by a single person including one activity of his/her choice, assuming that a person can only work on a single activity at a time.

Exact Input/Output Sample-1

Enter maximum number of activities: 11

Enter the name of activities with their start and finish time: (p, 3, 5) (q, 0, 6) (r, 3, 9) (s, 5, 9)

(t, 6, 10) (u, 8, 11) (v, 2, 14) (w, 12, 16) (x, 5, 7) (y, 8, 12) (z, 1, 4)

Enter the name of activity of person's choice: y

Maximum number of activity selected = 4
Selected Activity Set (s): {p, x, y w} or {z, x, y w}

Explanation
The activities sorted in ascending order with their finish time.
Activities
(z, 1, 4)
(p, 3, 5)
(q, 0, 6)
(x, 5, 7)
(r, 3, 9)
(s, 5, 9)
(t, 6, 10)
(u, 8, 11)
(y, 8, 12)
(v, 2, 14)
(w, 12, 16)

If user choice is t, the output will be

Maximum number of activity selected = 3
Selected Activity Set(s): {q, t, w} or {p, t, w} or {z, t, w}

If user choice is v, the output will be

Maximum number of activity selected = 1
Selected Activity Set(s): {v}

Round Exercise (RE)

6.5 Given a set of n intervals, input through keyboard, R = {II, 12, I3,….., In} where each Ij =[aj, bj] is an interval from the real line having length bj - aj and bj>aj. The objective is to find all possible largest sub set of intervals S ⊆ R, excluding all intervals of least length, such that no two intervals in S overlap each other.

Exact Input/Output Sample-1
Enter maximum number of intervals: 11

Enter the name of intervals with their start and end number: (p, 3, 5) (q, 0, 6) (r, 3, 9) (s, 5, 9)
(t, 6, 10) (u, 8, 11) (v, 2, 14) (w, 12, 16) (x, 5, 7) (y, 8, 12) (z, 1, 4)

The intervals with their lengths sorted in ascending order with their end number.

Intervals		Length of Intervals
(z, 1, 4)	-	3
(p, 3, 5)	-	2
(q, 0, 6)	-	6
(x, 5, 7)	-	2
(r, 3, 9)	-	6
(s, 5, 9)	-	4
(t, 6, 10)	-	4
(u, 8, 11)	-	3
(y, 8, 12)	-	4
(v, 2, 14)	-	12
(w, 12, 16)	-	4

The least length intervals :{p, x}

Excluding the least length intervals,
the largest sub set of non-overlapping intervals: {z, s, w}, {z, t, w}, {z, u, w}, {z, y, w}, {q, t, w}, {q, u, w}, {q, y, w}

Exact Input/Output Sample-1
If the input (x, 5, 7) is changed to (x, 5, 8), then output will be

The least length intervals :{p}

Excluding the least length intervals,
the largest sub set of non-overlapping intervals: {z, x, u, w}, {z, x, y, w}