Ex-no-14

04/01/2023

Factorial of N numbers

Aim:-

To Write a C Program for finding Factorial of N numbers

Algorithm:

Step 1. Start

Step 2. Read the number n

Step 3. [Initialize]

i=1, fact=1

Step 4. Repeat Step 4 through 6 until i=n

Step 5. fact= fact*i

Step 6. i=i+1

Step 7. Print fact

Step 8. Stop

RESULT:-

Thus the C Program for Finding factorial of N numbers has been executed and the output has been verified

Ex-no-1B

04/01/2023

AIM:

To write a C Program for sorting of N numbers using Insertion Sort.

ALGORITHM:

Step 1: - If the element is the first one, it is already Sorted

Step 2: Move to next element

Step 3: Compare the current element with all elements in the Sorted array

Step 4: If the element in the Sorted array is Smaller than the current element, iterate to the next element otherwise, Shift the entire greater element in the array by one Position towards the right

Step 5: Insert the value at the correct Position

Step 6: Repeat until the complete list is Sorted.

RESULT:-

Thus the C Program for Sorting of N numbers using Insertion Sort has been executed and the output has been verified

Ex-no-2

11/01/2023

AIM :

To write a c Program for Sorting of N numbers using Bubble Sort.

ALGORITHM:-

Step 1:- Repeat Steps 2 and 3 for i=1 to 10

Step 2: Set j=1

Step 3: Repeat While j<=n

if a[i] <a[j]

then interchange a[i] and a[j]

[End of if]

Set j=j+1

[End of Inner loop]

[End of Step 1 outer loop]

Step 4: Exit

RESULT:-

Thus the C Program for Sorting of N numbers using Bubble Sort has been executed and the output has been verified

Ex-no:-3A

11/01/2023

Recurrence Type - Merge Sort

AIM

To write a C Program for Sorting of N numbers using merge Sort

ALGORITHM :-

Step 1: Declare left and Right var which will mark the extreme indices of the array

Step 2: Left will be assigned to 0 and right will be assigned to n-1

Step 3: Find mid = (left+right)/2

Step 4: Call merge Sort on (left, mid) and (mid+1, rear)

```
Merge Sort (arr, left, right):
    if left > right
    return
mid = (left+ right) /2
merge Sort (arr, left, mid)
merge Sort (arr, mid+1, right)
merge (arr, left, mid, right)
end
```

Step 5: Continue till left<right

Step 6: merge on the 2 Sub Problems

RESULT:-

Thus the C Program for Sorting of N numbers using Merge Sort has been executed and the output has been verified

Ex-no: 3B

11/01/2023

Recurrence Type - Linear Search

AIM:-

To Write a C Program for Searching an element using Linear Search

ALGORITHM :-

Step 1 - Read the Search element from the user

Step 2 - Compare the Search element with the first element in the list.

Step 3 - If both are matched, then display "Given element is found!!!" and terminate the function

Step 4 - If both are not matched, then compare Search element with the next element in the list.

Step 5 - Repeat Steps 3 and 4 until Search element is compared with the last element in the list.

Step 6 - If the last element in the list also doesn't match, then display "Element is not Found!!!" and terminate the function.

RESULT:-

Thus the C Program for Searching an element by using linear Search has been executed and the output has been verified.

Ex-no-4A

18/01/2023

Quick Sort

AIM

To write a C Program for Sorting of N numbers using Quick Sort

ALGORITHM

Step 1- Consider the first element of the list as Pivot (i.e., Element at first position in the list).

Step 2- Define two variables i and j. Set i and j to first and last elements of the list respectively.

Step 3- Increment i until list[i] > Pivot then Stop.

Step 4- Decrement j until list[j] < Pivot then Stop.

Step 5 - If i < j then exchange list[i] and list[j].

Step 6- Repeat Steps 3,4 & 5 until i > j.

Step 7- Exchange the Pivot element with list[j] element.

Step 7- If the search element is larger than middle element, repeat steps 2,3,4 and 5 for the right Sublist of the middle element.

Step 8- Repeat the same process until we find the search element in the list or until Sublist contains only one element.

Step 9- If that element also doesn't match with the search element, then display "Element is not found in the list !!!" and terminate the function.

RESULT:-

Thus the C Program for Sorting of N numbers using Quick Sort has been executed and the output has been verified.

Ex-no-4B

18/01/2023

Binary Search

AIM

To write a c Program for Searching an element using Binary Search.

ALGORITHM

Step 1- Read the search element from the user.

Step 2- Find the middle element in the sorted list.

Step 3- Compare the search element with the middle element in the sorted list.

Step 4- If both are matched, then display "Given element is found!!!" and terminate the function.

Step 5- If both are not matched, then check whether the search element is smaller or larger than the middle element.

Step 6- If the search element is smaller than middle element, repeat Steps 2,3,4 and 5 for the left Sublist of the middle element.

RESULT:-

Thus the C Program for Searching an element by using Binary Search has been executed and the output has been verified

Ex-no-5

25/01/2023

Strassen Matrix multiplication

AIM :

To write a C Program for Performing matrix Multiplication using divide and conquer.

ALGORITHM :

Step 1 : Algorithm Strassen (n, a, b, d)
begin

Step 2 : If n = threshold then compute
C = a*b is a conventional matrix
Else

Step 3 : Partition a into four sub matrices
$a_{11}, a_{12}, a_{21}, a_{22}$.
Partition b into four sub matrices $b_{11}, b_{12}, b_{21}, b_{22}$.
Strassen $(n/2, a_{11} + a_{22}, b_{11} + b_{22}, d_1)$
Strassen $(n/2, a_{21} + a_{22}, b_{11}, d_2)$
Strassen $(n/2, a_{11} + b_{12} - b_{22}, d_3)$
Strassen $(n/2, a_{22} + b_{21} - b_{11}, d_4)$
Strassen $(n/2, a_{11} + a_{12}, b_{22}, d_5)$
Strassen $(n/2, a_{21} - a_{11}, b_{11} + b_{22}, d_6)$
Strassen $(n/2, a_{12} - a_{22}, b_{21} + b_{22}, d_7)$

Step 4 : $C = d_1 + d_4 - d_5 + d_7 \quad d_{3+} d_5$

$d_2 + d_4 \qquad\qquad d_1 + d_3 - d_2 - d_6$

end if

return (C)

end.

RESULT:-

Thus the C Program for multiplication of 2*2 elements using Strassen Matrix multiplication has been executed and the output has been verified

Ex-no-6A

01/02/2023

Finding Max and Min in Array

AIM

To write a C Program to find the maximum and minimum elements of a given array.

ALGORITHM

1. Let MaxE and minE be the variable to store the minimum and maximum element of the array.

2. Initialise minE as INT_MAX and maxE as INT_MIN

3. Traverse the given array arr[].

4. If the current element is smaller than minE, then update the minE as current element.

5. If the current element is greater than maxE, then update the MaxE as current element.

6. Repeat the above two steps for the element in the array.

RESULT:- A code has been written to find the maximum and minimum elements, and the output has been verified

Ex-no-6B

01/02/2023

Convex Hull Problem

AIM :

To write a C Program to use convex hull algorithm to find the convex hull length of a set of 2D Points

ALGORITHM :-

1. Choose a Point roughly in the centre of your Point cloud.

2. Then Sort the Points radially, by angle from the centre. The topmost Point must be in the convex hull, so define it as having an angle of 0.0 and being first in the list.

3. Put Point 2 in the "tentative" hull list.

4. Then check Point 3. If the angle P1-P2-P3 is Concave (relative to the centre Point), remove P2 from the list, if it is convex, keep it.

5. Continue Steps 3-6. backtracking and removing Points if they go concave.

6. You only need two Points in your "tentative" lists, once you have three, they become definite.

RESULT:-

A code has been written and the output has been verified

Ex-no-7A

08/02/2023

Huffman Coding using Greedy

AIM :-

To write a C Program to implement Huffman coding using Greedy algorithm.

ALGORITHM :-

1. Build a min heap that contains 6 nodes where each node represents root of a tree with single node

2. Extract two minimum frequency nodes from min heap. Add a new internal node with frequency 5+9 = 14

3. Extract two minimum frequency nodes from heap. Add a new internal node with frequency 12+13 = 25

4. Extract two minimum frequency nodes. Add a new internal node with a frequency 14 + 16 = 30

5. Extract two minimum frequency nodes. Add a new internal node with frequency 25 + 30 = 55

6. Extract two minimum frequency nodes. Add a new internal node with frequency ~~25~~ 45+55 =100

7. Print the resultant output.

RESULT:-

The code has been written, and the output has been verified

Ex-no -7B

08/02/2023

Knapsack using Greedy

AIM :

To Solve Knapsack Problems using greedy algorithm

ALGORITHM :

1) Sort the items in decreasing order is their value-to-weight ratio (i.e., value per unit weight).

2) Initialize the Knapsack to be empty, and set the total value to 0.

3) For each item in the sorted list, add the item to the Knapsack if there is enough capacity left and update the total value accordingly.

4) If the Knapsack is full, stop adding items.

RESULT:

The code has been written, and the output has been verified

Ex-no-8A

15/02/2023

Tree Traversal

AIM :

To write a code to demonstrate tree traversal.

ALGORITHM :

In order Traversal

1. First, visit all the nodes in the left subtree
2. Then the root node
3. visit all the nodes in the right subtree

Pre order traversal

1. visit root node
2. visit all the nodes in the left subtree
3. visit all the nodes in the right subtree

Post order traversal

1. visit all the nodes in the left subtree
2. visit all the nodes in the right subtree
3. visit the root node.

RESULT:-

The code has been written and the output has been verified

Ex-no-8B

15/02/2023

Krushkal's Minimum Spanning Tree

AIM :

To write a code to demonstrate Krushkal's MST algorithm

ALGORITHM :

1. Sort all the edges in non-decreasing order of their weight

2. Pick the smallest edge. check if it forms a cycle with the spanning tree formed so far. If cycle is not formed, include this edge. Else, discard it.

3. Repeat Step#2 until there are (V-1) edges in the spanning tree.

RESULT:-

The code has been written and the output has been verified

Ex.no-8
15/02/2023

Prim's Minimum Spanning Tree

AIM :

To write a code to demonstrate Prim's MST.

ALGORITHM :-

1. Create edge list of graph, with their weights
2. Draw all nodes ~~the~~ create Skeleton for spanning tree.
3. Select an edge. While adding an edge take care that the one end of the edge should always be in the Skeleton tree and it cost should be minimum.
4. Add other edges. While adding an edge take care that one end of the edge should always be in the Skeleton tree and its cost should be minimum
5. Repeat Step 5 until n-1 edges are added
6. Return

RESULT:-

The code has been written and the output has been verified

Ex-no-9

22/12/2023

Longest Common Subsequence

AIM :

To write a C Program to Print the longest common Sequence

ALGORITHM :

1) Construct L [m+1][n+1]

2) The value L[m][n] contains length of LCS. Create a character array lcs[] of length equal to the length of lcs plus 1 (one extra to store \0).

3) Traverse the 2D array starting from L[m][n]. Do following for every cell L[i][j] .... a) If characters (in x and y) corresponding to L[i][j] are same (or x[i-1] == y[j-1]), then include this character as part of LCS.

..... b) Else compare values of L[i-1][j] and L[i][j-1] and go in direction of greater value.

RESULT:

The code has been verified and the output has been verified.

Ex-no-10
01/03/2023

N Queen Problem.

AIM :-

To write a C Program to display and explain the N Queens Problem.

ALGORITHM :-

1) Start in the leftmost column.

2) If all queens are placed
   return true

3) Try all rows in the current column.
   Do following for every tried row.
   a) If the queen can be Placed safely in this row
   then mark this [row, column] as Part of the Solution and recursively check if Placing queen here leads to ~~the~~ a Solution

   b) If Placing the queen in [row, column] leads to a Solution then return ~~true~~ true.

   c) If Placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to Step (a) to try other rows.

4) If all rows have been tried and nothing worked, return false to trigger back tracking.

RESULT:-

The code has been written and the output has been verified

Ex-no-11

15/03/2023

Travelling Salesman Problem

AIM :-

To write a c Program to solve the travelling Sales man Problem using the dynamic Programming approach.

ALGORITHM :-

Step 1: Start the Process

Step 2: Enter the number of cities

Step 3: Enter the cost matrix of all the cities

Step 4: Find all Possible feasible Solutions by taking the Permutation of the cities which is to be covered.

Step 5: Find out the cost of each Path using the cost matrix

Step 6: Find out the Path with minimum cost

Step 7: If more than one Path having the Same cost considers the first occuring Path.

Stp 8: That is Selected as the optimum Solution

Step 9: Stop the Process

RESULT:-

Thus the travelling Salesman Problem using the dynamic Programming approach was executed Successfully

Ex-no-12a

29/03/2023

BFS Implementation With Array

AIM :-

To Write a C Program to Solve the BFS Problem With Array.

ALGORITHM :-

Step 1: Start by Putting any one of the graph's vertices at the back of a queue.

Step 2: Take the front item of the queue and add it to the visited list

Step 3: Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.

Step 4: Keep repeating Steps 2 and 3 until the queue is empty.

Step 5: The graph might have two different disconnected Parts so to make Sure that we cover every vertex, we can also run the BFS algorithm on every node

RESULT:-

Thus the BFS Problem using array was executed Success fully.

Ex-no-12b

29/03/2023

DFS Implementation with Array

AIM :

To write a C Program to solve the DFS Problem with array implementation

ALGORITHM :

Step 1: Start by Putting any one of the graph's vertices on top of a Stack

Step 2: Take the top item of the Stack and add it to the visited list.

Step 3: Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of the Stack.

Step 4: Keep repeating Steps 2 and 3 until the Stack is empty.

RESULT:

Thus, The DFS Problem using arrays was executed Successfully

Ex-no-13
12/04/2023

Randomized Quick Sort

AIM :-

To write a C Program to Solve the Randomized quick Sort

ALGORITHM :-

Quick Sort (A,P,r)

if p<r

then q PARTITION(A,P,r)

QUICK SORT (A,P,q)

QUICK SORT(A,q+1,r)

To Sort an entire array A, the initial call is QUICKSORT (A, 1, length[A]).

RESULT:-

Thus, the Randomized Quick Sort Problem using array was executed successfully.

Ex-no-14
19/04/2023

String Matching Algorithm

AIM :-

To write a C Program to solve the String Matching algorithm

ALGORITHM :-

Start
Pat_len := Pattern size
Str_len := String size
for i = 0 to (Str_len - Pat_len), do
for j := 0 to Pat_len, do
if text [i+j] ≠ Pattern [j], then
break
If j == Pat_len, then
display the Position i, as there Pattern found
End

RESULT:-

Thus the String matching algorithm was executed Successfully

Ex-no-15

26-04-2023

Analysing - Real Time Problem

AIM :-

To write a C Program to Sort an array using Merge sort and manipulate the time complexity of the Program

ALGORITHM :-

Step 1: Merge Sort (A[0..n-1])

Step 2: Sorts Array A[0..n-1] by recursive merge sort

Step 3: Input: An Array A[0..n-1] of orderable elements

Step 4: Output: Array A[0..n-1] Sorted in nondecreasing order

Step 5: Merge (B[0..p-1], C[0..q-1], A[0..p+q-1])

Step 6: Merges two Sorted Arrays into one Sorted array.

RESULT:-

Thus the merge Sort Program was executed Successfully with the time complexity