Program no: 1 MERGING Date:

Aim: Merge two sorted arrays in a third array

Algorithm:- MERGING(array1, array2, merge, m, n)

Let array1 and array2 be sorted arrays with m and n elements, respectively. This algorithm merges array1 and array2 into an array 'merge' with m+n elements.

1. SET i:=0, j:=0, k:=0 //[initialize]
2. Repeat while i<m and j<n: //[compare]
   If array[i] < array2[j] then:
   SET merge[k] := array1[i]
   SET k:=k+1 and i:= i+1
   Else:
   SET merge[k] := array2[j]
   SET k:=k+1 and j:= j+1
   [End of If structure]
   [End of loop]
3. Repeat while i<m, then:
   SET merge[k] := array1[i]
   SET i:=i+1

SET K := K+1

[End OF loop]

4. Repeat while j<n, then:

SET merge[K] := array2[j].

SET j := j+1 and K := K+1.

[End of loop]

5. Exit.

Program no: 2 SINGLY LINKED STACK Date:

Aim: singly linked stack - push, pop, Linear search

Algorithm:-

1. start
2. If user select push operation then
3. create a newNode with the given data
4. If top == NULL then:
   (check whether the stack is empty)
5. SET top = newnode
   SET newnode → next = NULL

ELSE:

SET newnode → next = top
top = new node.

[End of IF structure]

6. If user select pop operation then
7. If top == NULL then:
   (check whether stack is empty)
   display "stack is empty"

Else:

SET temp = top

(create a temporary node and set it to top)

display temp → data

8. SET top = temp → next
(make Top point to the next node)

9. free (temp)
(Delete the temporary node)

10. If user select search operation then.

11. Declare a pointer variable temp and the variable key that holds the value to be searched

12. SET temp = TOP
SET flag = 0

13. Repeat while temp != NULL
If temp → data == key then:
display "element found"
SET flag = 1
[End of if structure]

go to step 14

ELSE:

SET temp = temp → next

[End of while]

[End of IF]

14: IF flag == 0 then:

display "element not found"

[End of if structure]

15: if user select display operation then

16: declare a pointer nodeptr

SET nodeptr = top

17: if nodeptr == NULL then:

display "stack is empty"

[end of if structure]

18 while nodeptr != NULL then:

print nodeptr → data

SET nodeptr = nodeptr → next

[End of if structure]

If nodeptr != NULL then:

print "-->"

[End of if structure]

[End of while]

10: Exit.

Date: 8/1/21

Program no: 1 MERGING

Aim: Merge two sorted arrays in a third array

Algorithm:- MERGING(array1, array2, merge, m, n)

Let array1 and array2 be sorted arrays with m and n elements, respectively. This algorithm merges array1 and array2 into an array 'merge' with m+n elements.

1. SET i := 0, j := 0, k := 0 //[initialize]
2. Repeat while i<m and j<n: //[compare]
   If array[i] < array2[j] then:
      SET merge[k] := array1[i]
      SET k := k+1 and i := i+1
   Else:
      SET merge[k] := array2[j]
      SET k := k+1 and j := j+1
   [End of If structure]
   [End of loop]
3. Repeat while i<m, then:
      SET merge[k] := array1[i]
      SET i := i+1

SET K := K+1

[End OF loop]

4. Repeat while j<n, then:

SET merge[K] := array[j].

SET j := j+1 and K := K+1.

[End of loop]

5. Exit.

program no: 2 SINGLY LINKED STACK date: 5/4/21

Aim: singly linked stack - push, pop, Linear search

Algorithm:-

1. start
2. If user select push operation then
3. create a newNode with the given data
4. If top==NULL then:
   (check whether the stack is empty)
5. SET top = newnode
   SET newnode -> next=NULL

ELSE:

   SET newnode -> next=top
   top = new node.

[End of IF structure]

6. If user select pop operation then
7. If top==NULL then:
   (check whether stack is empty)
   display "stack is empty"

Else:

SET temp = top

(create a temporary node and set it to top)

display temp->data

8. SET top = temp->next
(make Top point to the next node)

9. free (temp)
(Delete the temporary node)

10. If user select search operation then.

11. Declare a pointer variable temp and the variable key that holds the value to be searched

12. SET temp = TOP
SET flag = 0

13. Repeat while temp != NULL
If temp->data == key then:
display "element found"
SET flag = 1
[End of if structure]

go to step 14

ELSE:

SET temp = temp → next

[End of while]

[End of IF]

14: IF flag == 0 then:

display "element not found"

[End of IF structure]

15: if user select display operation then

16: declare a pointer nodeptr

SET nodeptr = top

17: if nodeptr == NULL then:

display "stack is empty"

[end of IF structure]

18 while nodeptr != NULL then:

print nodeptr → data

SET nodeptr = nodeptr → next

[End of IF structure]

If nodeptr != NULL then:

print "-->"

[End of if structure]

[End of while]

10: Exit.

Program no: 3 circular queue Date: 7/1/21

Aim: circular Queue - Add, Delete search

Algorithm:

1. start
2. If user select the insertion operation then
3. declare a variable item with given value.
4. If front == 0 && rear == size-1 || front == rear+1 then:

   Display queue overflow"

   [End of If structure]

   go to step 7
5. If front == -1 Then:

   SET front = 0

   SET rear = 0

   [End of If structure]

   go to step 7
6. else:

   If rear == size-1

   SET rear = 0

[End
else:
SET rear = rear+1.
[End of if structure]
.
7. cq[SET cq[rear] = item
8. If user select deletion operation then:
9. If front == -1 then
display "queue underflow"
[End of if structure]
10 If front == rear then:
SET front = -1
SET rear = -1
[End of if structure]
11. If front == size-1 then:
SET front = 0
Else
SET front = front+1
[End of if structure]

10. If user select the display operation then

13. SET front_pos = front

SET Rear_pos = rear

14. If front == -1 then:

Display "queue is empty"

[End of If structure]

15. If front_pos <= rear_pos then:

Repeat while front_pos <= rear_pos then:

print cq[front_pos]

SET front_pos = front_pos+1

[End of while]

Else:

Repeat while front_pos <= size-1

print cq[front_pos]

SET front_pos = front_pos+1

[End of while]

16. SET front_pos = 0

17. Repeat while front pos <= rear pos then:

display cq[front_pos]

set front_pos = front_pos+1

[End of while]

[End of If]

18. If user select search operation then.

19. Declare a veriable ser with value to be searched

20. declare a temporary veriable temp then

SET temp = ser

21 SET i = front

22: Repeat for i <= rear then:

If te == cq[i] then:

print "i+1 found at location"

SET j = j+1

[End of If]

If j == 0 then

Display "item not fond"

[End of if structure]

SET i = i+1

[End of For loop]

23: Exit.

Date: 21/1/21

Program no: 4 Bit string

Aim: set data structure and set operations (union, intersection and difference) using Bit string

Algorithm: 1. start

2. If user select the union operation then:

3. declare two array set1[i] and set2[i], declare two variable n1, n2 for holding the size of two arrays

4. Read elements into the arrays set1[i] and set2[i]

5. If n1==n2 then:

6. SET i=0

7. Repeat for i<n2 then

SET set3[i]=set1[i] || set2[i]

8. SET i=i+1

[End of for loop]

9. SET i=0
10 Repeat for i<2 then
    print set3[i]
11. SET i=i+1
    [End of for loop]
    [End of if]

ELSE:
    print "size are not equal"
    Exit
12. if user select insertion operation then.
13. declare two array set1[i] and set2[i] with size n1, n2 Respectively and Read elements to the arrays
14. if n1==n2 then
15. SET i=0
16. Repeat for i<n2 then.
17. SET set3[i] = set1[i] && set2[i]

18. SET i = i+1

[End of For loop]

19 SET i = 0

20 Repeat for i<n2 then:

print set3[i]

21: set i=i+1

[End of for loop]

[End of IF]

ELSE:

print "size are not equal"

Exit.

22. If user select the substraction then

23. declar two array set1[i] and set2[i] with n1, n2 size respectively and input the elements to the array

24. if n1==n2 then

25. set i=0

26 : Repeat for i<n2 then:

SET set3[i]=set1[i]&&!set2[i]

27: i = i+1

[End of for loop]

28: SET i=0

29: Repeat for i<n2 then:

print set3[i]

30 : set i=i+1

[End of for loop]

[End of IF]

Else:

print size are not equal

31: Exit

Programme:5 Binary search tree Date:3/2/21

Aim Binary Search trees - Insertion deletion search

Algorithm:
1. Start
2. If user select the insertion operation then
3. Create a new BST node and assign values to it
4. createtree (node, data) // call the createtree function with the root value and the data entered by user.
5. If root == NULL then:
6. Declare a temporary variable temp
   SET temp → data = data.
   SET temp → left → right = NULL;
   return the new node temp to the calling function
   [End of If]

7. If (data < (node->data))

8. call the createnode function with node->left and assign the return value in node->left

node->lef = createtree(node->lef, data)
[End of if structure]

9. If data > node->data

10: call the createtree function with node->right and assign the return value in node->right

node->right = createtree(node->right data)

[End of if structure]

11. return the original root pointer (node) to the calling function.

12. If the user select the search element operation then.

13. search (node, data) // call the search function with root value and the element the to be searched

14. if node==NULL

print "element not found"

[End of if]

15 if data < node->data then:

call the search function with node->left and assign the Return value in node->left

node->lef = search(node->left, data

[End of if structure]

16. if data > node-> data then.

call search fuction with node-> right and assign the Return value in node-> right

16: node->right=search(node->right,data

[End of if structure]

Else:

print "Element found is" node->data

17: return the original root pointer (node) to the calling function.

18. If the user select the deletion operation then:

19. del(node,data) // call the del function with root value and the element to be deleted

18. declare a temporary variable *temp.

19. If node == NULL then:

print "Element not found"

[End of if]

20: If data < node->data then:
call the del function with node->left
and assign the Return value to
node->left

node->left = del(node->left, data)

[End of if]

21 If data > node->data then
call the del function with node->right
and assign the return value to
node->right

[End of if]

22. Else:

//delete this node and replace with
either minimum element in the
right sub tree or maximum element
in the left subtree

23. If node->right && node->left

// replace with minimum element in the right sub tree

24 SET call findmin function with node->right then Return value assign in temp
GO to step 30
SET temp=findmin(node->right)

SET node->data = temp->data
// replaced it with some other node.

25 SET call function del with value node->right, temp->data and return value assign in node->right

Else:

26. SET temp=node

// If there is only one or zero children then we can directly remove it from the tree and connect its parent to its child

if node->left == NULL then:

    SET node = node->right

ELSE:

    if node->right == NULL then:

        SET node = node->left

    free (temp)

[End of if]

[End of if]

[End of if]

return the original root pointer

findmin(node) calling

31. If node == NULL then:

return NULL
go to step 34
[End of If]

32. If nod → left then
call the function findmin
return f
with value (node → left) then
reden the value to caling function
return findmin (node → left)
Else
return node
goto step 34
[End of If]

33. If the user select the display option then:

34. inorder (node)
call the inorder function with root value

35. if node != NULL then

inorder(node→left)

call the function inorder with value node→left

36. print node→data

inorder(node→right

call the function inorder with value node→right

[End of if]

37. Exit.

Date: 14/12/21

Program6 Doubly Linked List

Aim : Doubly linked list- insertion, deletion Search

Algorithm: 1. Start

2. If user select the insert operation at beginning then:

3. If head == NULL then:

4. perform step 5.6 to 5.9
(Call the function create())

5. SET head = temp
temp1 = head

6. ELSE

7. perform step 5.6 to 5.9
(call the function create)

8. SET temp -> next = head
SET head -> prev = temp
SET head = temp
[End of IF structure]

9. If user choose the operation
   Insert node at end then:
10. If head == NULL then:
11. perform step 56 to 59
12. SET head = temp
    SET temp1 = head
13 Else.
14 perform step 56 to 59
15 SET temp1 -> next = temp
    SET temp -> prev = temp1
    SET temp1 = temp
16. [End of if structure]
16: If user choose insert at
    any position then:
17: Read the position and store
    it in to the variable pos

18. SET temp2 = head

19: If pos<1 || pos >= count+1 then:

Display "position out of range to insert"

Exit

[End of If structure]

20: If head==NULL && pos!=1 then:

Display "Empty list cannot insert other than 1st position"

Exit

[End of If structure]

21: If head==NULL && pos==1 then:

22. Perfoom step 56 to 59
(call function create())

23: SET head = temp

SET temp1 = head

[End of if]

Exit

24: Else
Repeat
25 : while i < pos then:

26: SET temp2 = temp2 → next

SET i = i+1
[End of while]

27 : Go perform step 56 to 59

28 : SET temp → prev = temp2

SET temp → next = temp2 → next

SET temp2 → next → prev = temp

SET temp2 → next = temp

29 : If user choose the operation
deletion then:

30: Read the position the it store
in to the variable pos

31 : SET temp2 = head

32: If pos < 1 || pos >= count + 1 then:

Display "position out of range to delete"

[End of if structure]

Exit

33. If head == NULL then

Display "Empty list no elements to delete"

[End of if structure]

Exit

34 ELSE

35. Repeat while i < pos

36. SET temp2 = temp2 → next

SET i = i + 1

[End of while]

36. If i == 1 then:

If temp2 → next == NULL then:

Display "Node deleted from list"

37: free temp 2

SET temp2 = head = NULL
[End of if structure]
Exit.

36. If temp2 → next = NULL then:

SET temp2 → prev → next = NULL

free temp2

Display "Node deleted from list"
[End of if structure]
Exit

37: SET temp2 → next → prev = temp2 → prev

38: If i != 1 then:

temp2 → prev → next = temp2 → next

~~then~~:

[End of if structure]

39. If i == 1 then

SET head = temp2 → next

Display "Node deleted"

40: free temp2

[End of if structure]

41: SET count = count-1

42: if user select display operation then:

43: SET temp2 = head

44 if temp2 == NULL

Display "List empty to display"

[End of if structure]

EXIT

45: while temp2->next != NULL then:

print temp2->n

SET temp2 = temp2->next

[End of while]

46: print temp2->n

47: if user select search operation then:

48: SET temp2 = head

49: If temp2 == NULL then:

display "List Empty to search for data"

Exit [End of if structure]

50: Read the value to be search and store it into the variable data

51 while temp2 != NULL then:

52 if temp2 -> n == data then.

display "data found in"

print count+1

Exit.

53: ELSE

SET temp2 = temp2 -> next

SET count = count + 1

[End of if]

[End of while]

54 : Display " not found)

55 : Exit

56 : //when call create function

56 : SET temp→prer=NULL //create()

57 : SET temp→next=NULL

Display " Enter value to node "

58 : Read data and assign it into

temp→n

temp→n = data

59 : count = count+1.

60 : Exit

program 7 disjointset date: 20/2/24

Aim: Disjoint sets and the associated operations (create, union, find)

program:
1. Start
2. Read the number of elements from user and store it in to the dis.n
3. call function makeset() then:
4. SET i=0
5. Repeat for i<dis.n then:
   SET dis.parent[i]=i
   SET dis.rank[i]=0
   SET i=i+1
   [End of for loop]
6: user select the union operation then:
7. Read the elements to perform union and store

in to x and y Respectively

8. //perform find operation with x and y store Result in to xset and yset perform step 23

9. If xset == yset then:
   [End of if]

10: If dis.rank[xset] < dis.rank[yset] then:
   SET dis.parent[xset] = yset.
   SET dis.rank[xset] = -1
   [End of if]

11 else if dis.rank[xset] > dis.rank [yset] then:
   SET dis.parent[yset] = xset;
   SET dis.rank[yset] = -1
   [End of if]

12: Else

13: SET dis.parent [yset] = xset

SET dis.rank [xset] = dis.rank [xset] + 1

SET dis.rank [yset] = -1

14. If user choose find operation then:

15: Read the elements to check If and store the value in to the variables x and y respectively

16 If find x == find y then:

Display "connected components"

17: else

Display "not connected components"

18: If user select the display operation then:

```
19: SET i=0
20: Repeat for i<dis.n then
        print dis.parent[i]
        SET i=i+1
    [End of for loop]
21: print dis.rank[i]
    SET i=0
22: Repeat for i<dis.n then:
        print dis.rank[i]
        SET i=i+1
    [End of for loop]
23: Exit if dis.parent[x]!=x
    then:
        SET dis.parent[x]=find(dis.parent[x]
        return dis.parent[x]
24. Exit
```