32 minimum and maximum

Aim:
To write a program to find the minimum and maximum elements in a given array of integers.

* pseudocode

Start
Read N
Read array a[0..N-1]
min = a[0]
max = a[0]
for i = 1 to n-1
if a[i] < min
min = a[i]
if a[i] > max
max = a[i]
end for
print min and max
end

* Input
N = 8
a[] = {5,7,3,4,9,12,6,2}

* output
min = 2
max = 12

33 Aim:
To write a program to find the minimum max elements in a given array of integers sorted in ascending order.

* pseudocode:

Start
Read N
Read array a[0..N-1]
min = a[0]
max = a[0]
for i = 1 to N-1
if a[i] < min
min = a[i]
if a[i] > max
max = a[i]
end for
print min and max
END

* Input

N=8

a[] = {2, 4, 6, 8, 10, 12, 14, 18}

* output

min = 2

max = 18

(34) Merge Sort

Aim: To write a program to sort a given unsorted array using the merge sort technique

* pseudocode:

```
MergeSort (arr, Low, high)
  If Low < high
    mid = (Low+high)/2
   mergeSort (arr, Low, mid)
   mergeSort (arr, mid+1, high)
   merge (arr, Low, mid, high)
  End if
  merge (arr, Low, mid, high)
  Create temporary arrays L and R
   copy data to L and R
   merge L and R back into arr
```

* Input

N=8

a[] = {31, 23, 35, 27, 11, 21, 15, 28}

* output

11, 15, 21, 23, 27, 28, 31, 35

(35) MergeSort → Count the no. of comparisons

Aim:

To Implement the merge sort algorithm, sort a given array and count the number of comparisons made during the sorting process

* pseudocode:

```
Count = 0
mergeSort (a, l, r){
 if (l < r)
```

```
{
 m=(l+r)/2
 mergeSort(a,l,m)
 mergeSort(a,m+1,r)
 merge(a,l,m,r)
 }
}
merge(a,l,m,r){
while (i<=m && j<=r)
 {
 count++
 copy smaller of a[i], a[j] to temp}
 copy remaining elements temp back to a
 }
```

* Input:

N=8

a[] = {12,4,78,23,45,67,89,1}

* Output:

Sorted Array: 1,4,12,23,45,67,78,89

No.of comparisons: 16

(3a) Aim:

To sort a given array using quick sort by choosing the first element as pivot, showing the array after each partition and recursive call

* Pseudocode

```
quickSort (a,l,r)
{
 if (l<r)
 {
 p= partition(a,l,r)
   quickSort(a,l,p-1)
   quickSort(a,p+1,r)
   }
 }
partition(a,l,r){
   pivot = a(l)
   rearrange elements
   return pivot position}
```

* Input

n=7

a[] = {38,27,43,3,9,82,10}

* Output

3,9,10,27,38,43,82

37 Quick-Sort Algorithm

Aim:
To Implement the Quick-Sort algorithm using the middle element as the pivot, display the array after each partition and recursive call and obtain the final sorted array

* pseudocode

```
QuickSort(a,l,r)
{
 if(l<r)
 {
  p=partition(a,l,r)
 QuickSort(a,l,p-1)
 QuickSort(a,p+1,r)
 }
}
```

* Input
N=8
a[]={19,72,35,46,58,91,22,31}

* output
19,22,31,35,46,58,72,91.

38 Binary Search Algorithm

Aim:
To implement the Binary Search algorithm to find the position of a given key element in a sorted array & count the no. of comparisons

* Pseudocode

```
BinarySearch(a,n,key) {
 l=0; r=n-1
 while(l<=r)
  {
 mid=(l+r)/2
 if (a[mid]==key) return mid
 elseif (key<a[mid]) r=mid-1
   else l=mid+1
  }
 }
```

* Input
N=7
a[] = {21,32,40,54,65,76,87}
search key = 32

* output
Position: 2
Comparisons: 2

39 Binary Search Algorithm

Aim:
To Locate a given element in a sorted array using the Binary search algorithm

* pseudocode

```
BinarySearch (a,n,key)
{
 l=0 ; r=n -1
   while(l<=r)
  {
  mid = (l+r)/2
  if(a[mid]==key) return mid+1
  else if (key<a[mid]) r=mid-1
  else l=mid+1
     }
    }
```

* Input:
 N=7
a[] ={13, 19, 24, 29, 35, 41, 42}
 Search key = 42

* output:
position = 7

40 / K-closest points

Aim:
To find the k closest points to the origin (0,0) from a given set of points on the x-y plane.

* pseudocode

```
 Kclosest (points, n, k)
 {
Sort points by (x*x + y*y)
 return first k points
 }
```

* Input:
 points = [[1,3], [-2,2], [5,8], [0,1]]
 k=2

* output
 [[-2,2], [0,1]]

41 Aim:
To compute the number of tuples (i,j,k,l) such that
 A[i]+B[j]+C[k]+D[l]=0

* pseudocode.

FourSumCount (A, B, C, D) {
Store sums of A[i]+B[j] in map. for each C[k]+D[l]
add map [-(C[k]+D[l])] to count
return count
}

* Input

A = [1, 2]
B = [-2, -1]
C = [-1, 2]
D = [0, 2]

* output

2

(42) Aim

To find the k-th smallest element in an unsorted array using the median of medians algorithm

* pseudocode

Select (arr, n, k)
{
if (n <= 5) return sort (arr) [k-1] divide arr into groups of 5 find medians of groups
pivot = Select (medians, m, m/2) partition arr around pivot recurse on required part
}

* Input

arr = [12, 3, 5, 7, 19]
k = 2

* Output

5

(43)

K^th smallest element

Aim: To find the k-th smallest element in an unsorted array using the median of medians algorithm

* pseudocode

medianofmedians (arr, n, k) {
if (n <= 5) return sorted (arr) [k-1]
divide arr into groups of 5
pivot = median of medians

partition arr around pivot
recurse on required part
}

* Input

arr = [1,2,3,4,5,6,7,8,9,10]
k = 6
* output

6

(44)

Meet in the middle

Aim:
To find a subset whose sum is closest to a target using meet in the middle technique.

* pseudocode:

```
MeetInMiddle(arr, n, target)
{
divide arr into left and right halves
 Compute all subset sums of left -> L
 Compute all subset sums of right -> R
 Sort R
 for each sum in L
 find closest sum in R to target - sum
 update closest_sum if better return closest_sum
}
```

* Input

Set = {45, 34, 4, 12, 5, 2}
Target = 42
* output

Closest subset sum = 41

(45)

MITM technique

Aim: To determine whether there exists a subset of a given array that sums exactly to a target sum E using meet-in-the middle technique.

* pseudocode

```
MeetInmiddle(arr, n, target) {
divide arr into left and right halves
  L = all subset sums of left
  R = all subset sums of right sort R
 for each sum in L
   if target - sum exists in R
 return true
 return false
```

* Input

S = [3, 34, 4, 12, 5, 2]

exact sum = 15

* output

True

46

2×2 matrices

Aim:

To compute the product of two 2×2 matrices using Strassen's matrix multiplication Algorithm and obtain the resulting matrix C = A × B.

* pseudocode

read a, b, c, d, e, f, g, h

M1 = (a+d) * (e+h); M2 = (c+d) * e

M3 = a * (f-h); M4 = d * (g-e)

M5 = (a+b) * h; M6 = (c-a) * (e+f)

M7 = (b-d) * (g+h)

C11 = M1 + M4 - M5 + M7

C12 = M3 + M5

C21 = M2 + M4

C22 = M1 - M2 + M3 + M6

print C11, C12, C21, C22

* Input

enter elements of matrix A:

a b
c d

enter elements of matrix B:

e f
g h

* output

product matrix C:

C11 C12
C21 C22

47.

Karatsuba multiplication

Aim

To compute the product of two integers using Karatsuba multiplication algorithm which re[duces] the number of multiplication's compar[ed] to the traditional method.

* Pseudocode

```
Karatsuba(x, y):
if x<10 or y<10 return x*y
split x into a,b; y into c,d
  p1 = Karatsuba(a, c)
  p2 = Karatsuba(b, d)
  p3 = Karatsuba(a+b, c+d)
return p1*10^n + (p3-p1-p2)*10^(n/2) + p2
```

* Input

$x = 1234$
$y = 5678$

* output

$z = 7006652$.