# TASK 3

# Aim:

# To Sort a given set of elements using the Quicksort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list 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.

**Step 1 − Choose the highest index value has pivot**

**Step 2 − Take two variables to point left and right of the list excluding pivot**

**Step 3 − left points to the low index**

**Step 4 − right points to the high**

**Step 5 − while value at left is less than pivot move right**

**Step 6 − while value at right is greater than pivot move left**

**Step 7 − if both step 5 and step 6 does not match swap left and right**

**Step 8 − if left ≥ right, the point where they met is new pivot**

#include<stdio.h>

#include<sys/time.>

Voidswap(int\*x,int\*y)

{

Int temp;

temp=\*x;

\*x=\*y;

\*y=temp;

}

Void generate\_random(inta[],intn)

{

Int i;

srand(time(0));

for(i=0;i<n;i++)

a[i]=rand()%1000;

}

Int Partition(inta[10],intl ,int i,j,p;,inth)

{i=l;j=h+1;p=l;

do

{

do

{

i=i+;

}while(a[i]<a[p]);do{

j=j-1;

}while(a[j]>a[p]);

swap(&a[i],&a[j]);

}while(i<=j);swap(&a[i],&a[j]);

swap(&a[l],&a[j]);returnj;

}

Int Quicksort(inta[10],int m,intn)

{

int s;if(m<n+1)

{

s=Partition(a,m,n);Quicksort(a,m,s-1);Quicksort(a,s+1,n);returna;

}

}

Int main()

{

int a[100000],i,ch,n;struct timeval t;double start,end;FILE\*fp;

printf("Enterthetypeof operation\n");

printf("1-Randomly generate numbers and quicksort\n");

printf("2-Enterthenumberofvaluestogenerateand sort\n");scanf("%d",&ch);

switch(ch)

{

Case 1:

fp=fopen("quicksort.txt","w");

for(i=10000;i<100000;i=i+5000)

{

generate\_random(a,i);

gettimeofday(&t,NULL);

start=t.tv\_sec+(t.tv\_usec/1000000.0);

Quicksort(a,0,i-1);

gettimeofday(&t,NULL);

end=t.tv\_sec+(t.tv\_usec/1000000.0); printf("%d\t%lf\n",i,end-start); fprintf(fp,"%d\t%lf\n",i,end-start);

}

fclose(fp);break;

case 2:printf("Enter the number of values to be generated\n");

scanf("%d",&i);

generate\_random(a,i);gettimeofday(&t,NULL);

start=t.tv\_sec+(t.tv\_usec/1000000.0);Quicksort(a,0,i1);

gettimeofday(&t,NULL);

end=t.tv\_sec+(t.tv\_usec/100000 0.0);printf("%d\t%lf\n",i,end-start);

printf("Sorted numbers are\n");

printf("The sorted array is\n");

for(n=0;n<i;n++)

printf("%d\t",a[n]);break;

default:

printf("Invalid choice\n");

}

return0; }

Output:

Enter the type of operation

1-Randomly generate numbers and quicksort

2-Enter the number of values to generate and sort

1

10000 0.002030

15000 0.001000

20000 0.001996

25000 0.001996

30000 0.002994

35000 0.002993

40000 0.003990

45000 0.003991

50000 0.004987

55000 0.005986

60000 0.006014

65000 0.006983

70000 0.007980

75000 0.007980

80000 0.007980

85000 0.008977

90000 0.008975

95000 0.008976

--------------------------------

Process exited after 3.892 seconds with return value 0

Enter the type of operation

1-Randomly generate numbers and quicksort

2-Enter the number of values to generate and sort

2

Enter the number of values to be generated

10

978 439 857 868 982 985 813 587 318 480

10 0.000000

Sorted numbers are

The sorted array is

318 439 480 587 813 857 868 978 982 985

--------------------------------

Process exited after 6.245 seconds with return value 0

Press any key to continue . . .

test case2:

Enter the type of operation

1-Randomly generate numbers and quicksort

2-Enter the number of values to generate and sort

2

Enter the number of values to be generated

10

192 232 242 402 660 718 778 810 813 966

10 0.000000

Sorted numbers are

The sorted array is

192 232 242 402 660 718 778 810 813 966

--------------------------------

Process exited after 5.296 seconds with return value 0

Press any key to continue .

RESULT:

Thus, successfully completed the quick sort and executed the output.