**GANPAT UNIVERSITY**

**U. V. PATEL COLLEGE OF ENGINEERING**

**DEPARTMENT OF CE/IT**

**ACADEMIC YEAR: JAN - MAY 2021**

**Subject:** 2CEIT402: Design & Analysis of Algorithm

**Sem/Branch:** B.Tech 4th (CE/IT/CE-AI)

**1. Write user defined functions for the following sorting methods and compare their performance by time measurement with random data and Sorted data.**

1. **Selection Sort**
2. **Bubble Sort**
3. **Insertion Sort**
4. **Merge Sort**
5. **Quick Sort**

1. Selection sort :-

# INPUT :-

#include <stdio.h> int main(void) { int a[]={10,40,50,20,30}; int n = sizeof(a)/sizeof(a[0]); Selection\_Sort(a,n); return 0;

}

void Selection\_Sort(int a[],int n)

{ int i,j,temp,min,l,p=0;

for(i=0;i<n-1;i++)

{ min=i; p++;

for(j=i+1;j<n;j++)

{

if(a[j]<a[min])

{ min=j;

}

} if(min!=i)

{ temp=a[i]; a[i]=a[min]; a[min]=temp;

}

printf("\nPass %d : ",p);

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

{ printf("%d ",a[l]);

}

}

}

2. Bubble sort :-

# INPUT :-

#include <stdio.h> int main(void) { int a[]={10,40,50,20,30}; int n = sizeof(a)/sizeof(a[0]); Bubble\_Sort(a,n); return 0; }

void Bubble\_Sort(int a[],int n)

{ int i,j,temp,m=0,l; for(i=0;i<n-1;i++)

{ m++;

for(j=0;j<n-1-i;j++)

{

if(a[j]>a[j+1])

{ temp=a[j]; a[j]=a[j+1]; a[j+1]=temp;

}

} printf("\nPass %d : ",m);

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

{ printf("%d ",a[l]);

}

}

3. Insertion sort :-

# INPUT :-

#include <iostream> using namespace std;

void insertionsort(int arr[],int n)

{

int i,key,j; for(i=1;i<n;i++)

{

key=arr[i]; j=i-1;

while(j>=0 && arr[j]>key)

{

arr[j+1]=arr[j];

j=j-1;

} arr[j+1]=key;

}

}

int main()

{ int arr[]={34,234,656,757,5};

int n=sizeof(arr)/sizeof(arr[0]);

insertionsort(arr,n); int i; for(i=0;i<n;i++) printf("%d",arr[i]); printf("\n"); return 0;

}

4. Merge sort :-

# INPUT :-

#include <stdio.h> #define max 10

int a[10] = {10,14,19,26,27,31,33,35,42,44}; int b[10];

void merge(int low, int mid, int high) {

int l1, l2, i;

for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++) { if(a[l1] <= a[l2]) b[i] = a[l1++]; else b[i] = a[l2++];

}

while(l1 <= mid) b[i++] = a[l1++]; while(l2 <= high) b[i++] = a[l2++]; for(i = low; i <= high; i++) a[i] = b[i]; } void Merge\_sort(int low, int high) { int mid; if(low < high) { mid = (low + high) / 2; Merge\_sort(low, mid); Merge\_sort(mid+1, high); merge(low, mid, high); } else {

return;

} } int main() {

int i;

Merge\_sort(0, max); for(i = 0; i <= max; i++) printf("%d ", a[i]); }

5. Quick sort :-

# INPUT :-

#include <iostream>

using namespace std; int step1=0; int step2=0;

int partition(int \*arr,int first,int last)

{

int pivot=arr[last]; int i=first-1;

for(int j=first;j<=last-1;j++)

{

if(arr[j]<pivot)

{ i++; int temp=arr[i]; arr[i]=arr[j];

arr[j]=temp; } step1++; }

for(int j=last-1;j>=i+1;j--) arr[j+1]=arr[j]; arr[i+1]=pivot; return i+1; }

void quicksort(int \*arr,int first,int last,int size)

{

if(last<=first) return; int pos= partition(arr,first,last); quicksort(arr,first,pos-1,size); quicksort(arr,pos+1,size-1,size);

step2++; } int main()

{ int arr[]={4,54,6,456,5,76,57,676,465,76,87,68,7,97,98};

int size=sizeof(arr)/sizeof(arr[0]); quicksort(arr,0,size-1,size); for(int i=0;i<size;i++) printf("%d ",arr[i]);

printf("size:%d,comparison:%d ",size,step1+step2);

return 0;

}

# TABLE :-

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NO.** | **SELECTION** | **BUBBLE** | **INSERTION** | **QUICK** | **MERGE** |
| 5 | 0.005824 | 0.005306 | 0.009639 | 0.007181 | 0.006532 |
| 10 | 0.006868 | 0.005577 | 0.009835 | 0.008128 | 0.006275 |
| 15 | 0.005764 | 0.00546 | 0.006817 | 0.007564 | 0.005938 |
| 20 | 0.008574 | 0.005864 | 0.009318 | 0.007306 | 0.006124 |
| 25 | 0.008256 | 0.005644 | 0.006578 | 0.007222 | 0.006049 |

# GRAPH :-

