ASSIGNMENT 3

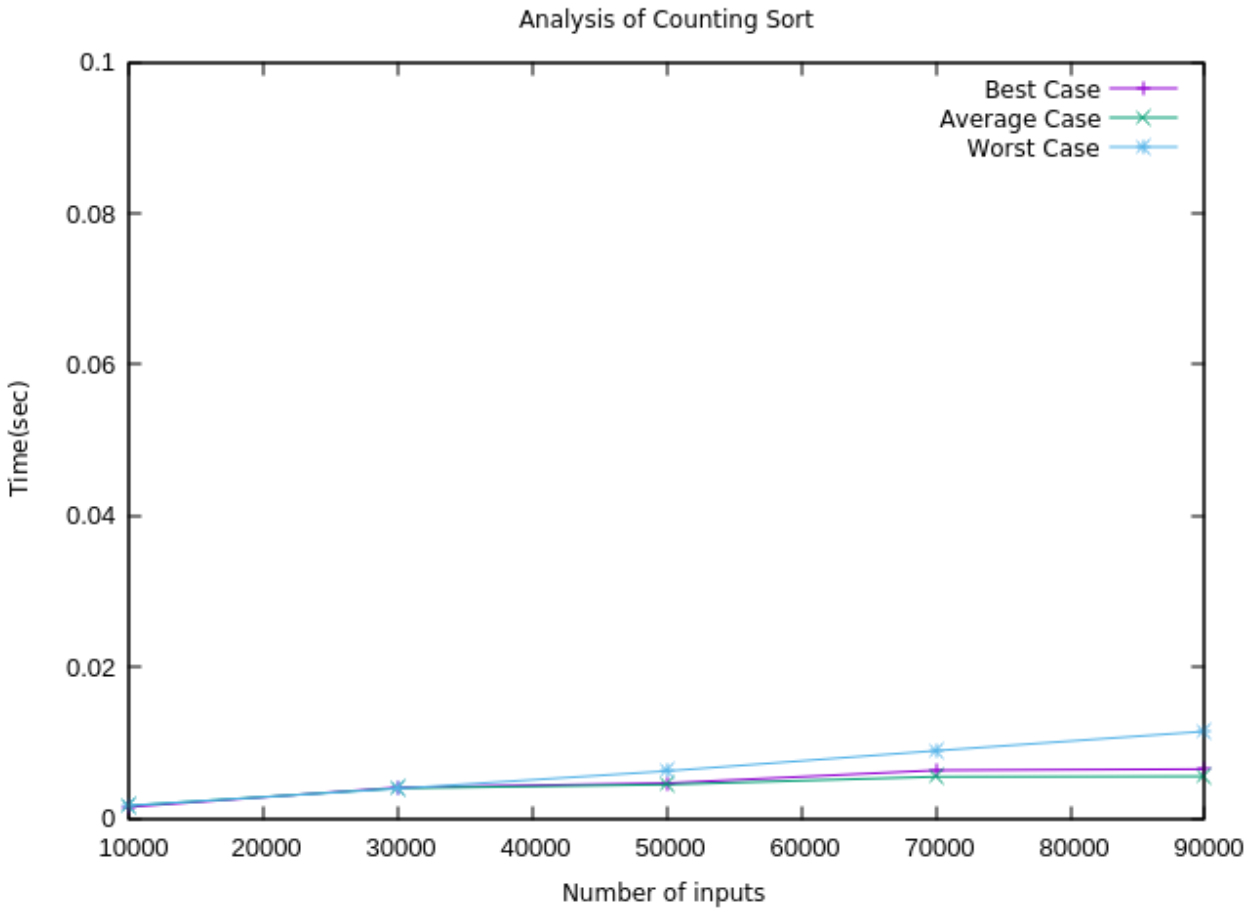
COUNTING SORT……

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
void countingsort(int arr[], int n,int k)
{
    int output[n];
    int i, count[k+1] = {0};
    for (i = 0; i < n; i++)
        count[arr[i]]++;
    for (i = 1; i <= k; i++)
        count[i] += count[i - 1];
    for (i = n - 1; i >= 0; i--)
    {
        output[count[ arr[i]] - 1] = arr[i];
        count[ arr[i] ]--;
    }

    for (i = 0; i < n; i++)
        arr[i] = output[i];
}

int main()
{
  int t1,t2,n;
  printf("enter number of test cases \n");
  scanf("%d",&t1);
  t2=t1;
  FILE *fp,*fp1,*fp2,*fp3;
  fp=fopen("count.txt","w");
  while(t1--)
  {
   n=(t2-t1)*20000;
   int i;
    int arr1[n],arr2[n],arr3[n],num;
    clock_t beg,end;
     double time1,time2,time3;
```

```
fp1=fopen("counta.txt","w");
for(i=0;i<n;i++)
{
   num=rand()%100;
   fprintf(fp1,"%d\n",num);
}
fclose(fp1);
 fp2=fopen("countb.txt","w");
 for(i=0;i<n;i++)
{
   fprintf(fp2,"%d\n",i*100/n);
}
fclose(fp2);
fp2=fopen("countb.txt","r");
  for(i=0;i<n;i++)
{
   fscanf(fp2,"%d\n",&arr2[i]);
}
fclose(fp2);
 fp3=fopen("countw.txt","w");
 for(i=0;i<n;i++)
{

   fprintf(fp3,"%d\n",arr2[n-i]);
}
fclose(fp3);
fp1=fopen("counta.txt","r");
 for(i=0;i<n;i++)
{
   fscanf(fp1,"%d\n",&arr1[i]);
}
fclose(fp1);
  fp3=fopen("countw.txt","r");
  for(i=0;i<n;i++)
{
   fscanf(fp3,"%d ",&arr3[i]);
}
fclose(fp3);
beg=clock();
countingsort(arr1,n,99);
end=clock();
time1=(double)(end-beg)/CLOCKS_PER_SEC;
beg=clock();
```

```
    countingsort(arr2,n,99);
    end=clock();
    time2=(double)(end-beg)/CLOCKS_PER_SEC;
    beg=clock();
    countingsort(arr3,n,99);
    end=clock();
    time3=(double)(end-beg)/CLOCKS_PER_SEC;
     fp1=fopen("countas.txt","w");
    for(i=0;i<n;i++)
    {

       fprintf(fp1,"%d\n",arr1[i]);
    }
    fclose(fp1);
     fp2=fopen("countbs.txt","w");
     for(i=0;i<n;i++)
    {
       fprintf(fp2,"%d\n",arr2[i]);
    }
    fclose(fp2);
     fp3=fopen("countws.txt","w");
     for(i=0;i<n;i++)
    {
       num=rand();
       fprintf(fp3,"%d\n",arr3[i]);
    }
    fclose(fp3);
    printf("%d %lf %lf %lf\n",n,time1,time2,time3);
    fprintf(fp,"%d %lf %lf %lf\n",n,time1,time2,time3);
  }
  return 0;
}
```

Inputs	Best	Average	Worst
10000	0.001414	0.001652	0.001669
30000	0.004065	0.003948	0.003904
50000	0.004627	0.004445	0.006240
70000	0.006338	0.005467	0.008921
90000	0.006484	0.005536	0.011483



MERGE SORT……..

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define infinite 999999999
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 =  r - m;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
```

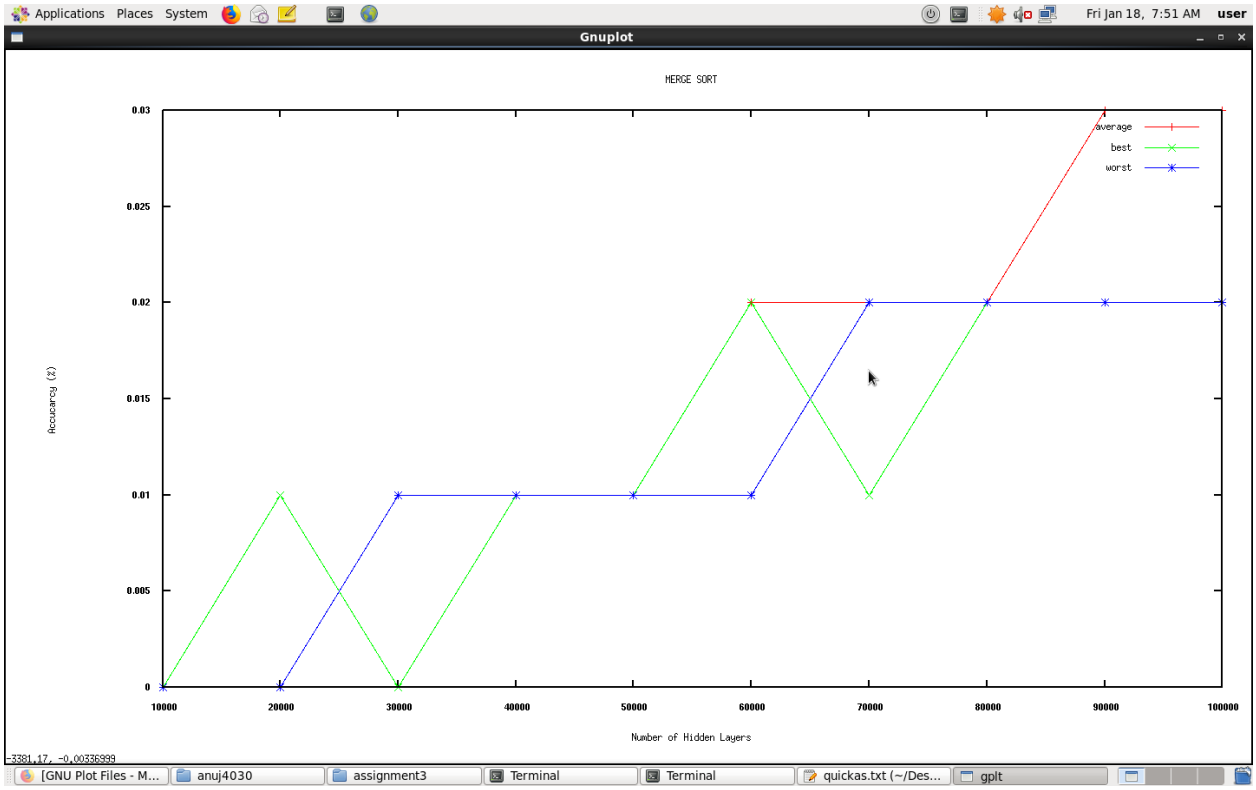
```
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1+ j];
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            arr[k] = L[i];
            i++;
        }
        else
        {
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1)
    {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2)
    {
        arr[k] = R[j];
        j++;
        k++;
    }
}

void mergesort(int arr[], int l, int r)
{
    if (l < r)
    {
        int m = l+(r-l)/2;
        mergesort(arr, l, m);
        mergesort(arr, m+1, r);
        merge(arr, l, m, r);
```

```
    }
}

int main()
{
  int t1,t2,n;
  printf("enter number of test cases \n");
  scanf("%d",&t1);
  t2=t1;
  FILE *fp,*fp1,*fp2,*fp3;
  fp=fopen("merge.txt","w");
  while(t1--)
  {
   n=(t2-t1)*40000;
   int i;
    int arr1[n],arr2[n],arr3[n],num;
    clock_t beg,end;
     double time1,time2,time3;
    fp1=fopen("mergea.txt","w");
    for(i=0;i<n;i++)
    {
       num=rand()%1000000;
       fprintf(fp1,"%d\n",num);
    }
    fclose(fp1);
     fp2=fopen("mergeb.txt","w");
     for(i=0;i<n;i++)
    {
       fprintf(fp2,"%d\n",i);
    }
    fclose(fp2);
     fp3=fopen("mergew.txt","w");
     for(i=0;i<n;i++)
    {

       fprintf(fp3,"%d\n",n-i);
    }
    fclose(fp3);
    fp1=fopen("mergea.txt","r");
     for(i=0;i<n;i++)
    {
       fscanf(fp1,"%d\n",&arr1[i]);
    }
```

```
fclose(fp1);
 fp2=fopen("mergeb.txt","r");
  for(i=0;i<n;i++)
{
   fscanf(fp2,"%d\n",&arr2[i]);
}
fclose(fp2);
  fp3=fopen("mergew.txt","r");
  for(i=0;i<n;i++)
{
   fscanf(fp3,"%d ",&arr3[i]);
}
fclose(fp3);
beg=clock();
mergesort(arr1,0,n-1);
end=clock();
time1=(double)(end-beg)/CLOCKS_PER_SEC;
beg=clock();
mergesort(arr2,0,n-1);
end=clock();
time2=(double)(end-beg)/CLOCKS_PER_SEC;
beg=clock();
mergesort(arr3,0,n-1);
end=clock();
time3=(double)(end-beg)/CLOCKS_PER_SEC;
 fp1=fopen("mergeas.txt","w");
for(i=0;i<n;i++)
{

   fprintf(fp1,"%d\n",arr1[i]);
}
fclose(fp1);
 fp2=fopen("mergebs.txt","w");
 for(i=0;i<n;i++)
{
   fprintf(fp2,"%d\n",arr2[i]);
}
fclose(fp2);
 fp3=fopen("mergews.txt","w");
 for(i=0;i<n;i++)
{
   num=rand();
   fprintf(fp3,"%d\n",arr3[i]);
```

```
    }
    fclose(fp3);
    printf("%d %lf %lf %lf\n",n,time1,time2,time3);
    fprintf(fp,"%d %lf %lf %lf\n",n,time1,time2,time3);
  }
  return 0;
}
```

	Avg	best	worst
40000	0.010000	0.010000	0.010000
80000	0.020000	0.020000	0.010000
120000	0.030000	0.030000	0.030000
160000	0.040000	0.040000	0.040000
200000	0.060000	0.050000	0.040000
240000	0.080000	0.050000	0.060000
280000	0.090000	0.060000	0.070000
320000	0.100000	0.080000	0.070000
360000	0.110000	0.090000	0.080000
400000	0.130000	0.090000	0.100000



QUICK SORT……

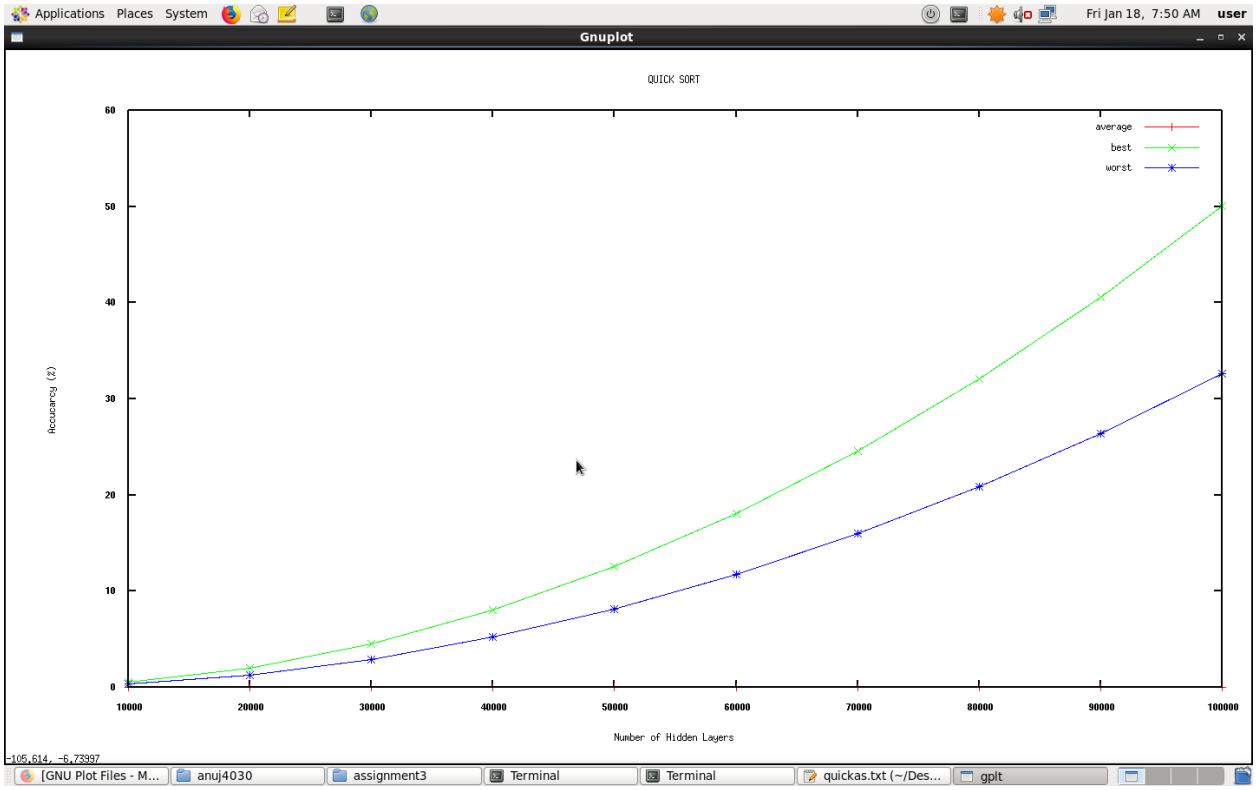
```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
void swap(int* a, int* b)
{
    int t = *a;
    *a = *b;
    *b = t;
}

int partition (int arr[], int low, int high)
{
    int pivot = arr[high];
    int i = (low - 1),j;

    for ( j = low; j <= high- 1; j++)
```

```
    {

        if (arr[j] <= pivot)
        {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}


void quicksort(int arr[], int low, int high)
{
    if (low < high)
    {

        int pi = partition(arr, low, high);
        quicksort(arr, low, pi - 1);
        quicksort(arr, pi + 1, high);
    }
}


int main()
{
  int t1,t2,n;
  printf("enter number of test cases \n");
  scanf("%d",&t1);
  t2=t1;
  FILE *fp,*fp1,*fp2,*fp3;
  fp=fopen("quick.txt","w");
  while(t1--)
  {
   n=(t2-t1)*10000;
   int i;
    int arr1[n],arr2[n],arr3[n],num;
    clock_t beg,end;
     double time1,time2,time3;
    fp1=fopen("quicka.txt","w");
    for(i=0;i<n;i++)
    {
```

```
   num=rand()%1000000;
   fprintf(fp1,"%d\n",num);
}
fclose(fp1);
 fp2=fopen("quickb.txt","w");
 for(i=0;i<n;i++)
{
   fprintf(fp2,"%d\n",i);
}
fclose(fp2);
 fp3=fopen("quickw.txt","w");
 for(i=0;i<n;i++)
{

   fprintf(fp3,"%d\n",n-i);
}
fclose(fp3);
fp1=fopen("quicka.txt","r");
 for(i=0;i<n;i++)
{
   fscanf(fp1,"%d\n",&arr1[i]);
}
fclose(fp1);
 fp2=fopen("quickb.txt","r");
  for(i=0;i<n;i++)
{
   fscanf(fp2,"%d\n",&arr2[i]);
}
fclose(fp2);
  fp3=fopen("quickw.txt","r");
  for(i=0;i<n;i++)
{
   fscanf(fp3,"%d ",&arr3[i]);
}
fclose(fp3);
beg=clock();
quicksort(arr1,0,n-1);
end=clock();
time1=(double)(end-beg)/CLOCKS_PER_SEC;
beg=clock();
quicksort(arr2,0,n-1);
end=clock();
time2=(double)(end-beg)/CLOCKS_PER_SEC;
```

```
    beg=clock();
    quicksort(arr3,0,n-1);
    end=clock();
    time3=(double)(end-beg)/CLOCKS_PER_SEC;
     fp1=fopen("quickas.txt","w");
    for(i=0;i<n;i++)
    {

       fprintf(fp1,"%d\n",arr1[i]);
    }
    fclose(fp1);
     fp2=fopen("quickbs.txt","w");
     for(i=0;i<n;i++)
    {
       fprintf(fp2,"%d\n",arr2[i]);
    }
    fclose(fp2);
     fp3=fopen("quickws.txt","w");
     for(i=0;i<n;i++)
    {
       num=rand();
       fprintf(fp3,"%d\n",arr3[i]);
    }
    fclose(fp3);
    printf("%d %lf %lf %lf\n",n,time1,time2,time3);
    fprintf(fp,"%d %lf %lf %lf\n",n,time1,time2,time3);
  }
  return 0;
}
```

	Avg	Best	worst
10000	0.000000	0.500000	0.320000
20000	0.010000	2.000000	1.310000
30000	0.010000	4.510000	2.930000
40000	0.010000	8.010000	5.220000
50000	0.010000	12.540000	8.15000
60000	0.010000	18.030000	11.740000
70000	0.020000	24.530000	15.980000
80000	0.020000	32.070000	20.850000
90000	0.020000	40.560000	26.400000
100000	0.020000	50.070000	32.590000


Applications
Places
System
Fri Jan 18, 7:50 AM
user
Gnuplot
QUICK SORT
average
best
worst
Accuracy (%)
Number of Hidden Layers
0
10
20
30
40
50
60
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
-105.614, -6.73397
[GNU Plot Files - M...
anuj4030
assignment3
Terminal
Terminal
quickas.txt (~/Des...
gplt