**Student Name:-Ritam Hudait**

**Student Roll No.:- 1905797**

**Algorithm Lab. Class Assignment-4**

**CSE Group 1**

**Date: - 30th July 2021**

1. **Write a C program for bubble sort to**
2. **Compare the time complexity with the given data set given below and calculate the time complexity based on the CPU clock.**
3. **Plot a graph showing the comparison (n, the input data Vs. CPU times)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl No.** | **Value of n** | **Bobble Sort (Time Complexity)** | | |
| **Best case** | **Average case** | **Worst case** |
| **1** | **5000** | **0.027934** | **0.019358** | **0.080293** |
| **2** | **10000** | **0.112768** | **0.197932** | **0.309476** |
| **3** | **15000** | **0.255694** | **0.442817** | **0.710337** |
| **4** | **20000** | **0.451722** | **0.79504** | **1.252392** |
| **5** | **25000** | **0.702562** | **1.237586** | **1.957736** |
| **6** | **30000** | **1.010563** | **1.785654** | **2.823007** |
| **7** | **35000** | **1.373549** | **2.418552** | **4.062403** |
| **8** | **40000** | **1.801679** | **3.225625** | **5.017430** |
| **9** | **45000** | **2.297573** | **4.019314** | **6.358995** |
| **10** | **50000** | **2.818479** | **4.973275** | **7.813238** |

**Program**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<time.h>**

**void main()**

**{**

**int \*a,n;**

**clock\_t start,end;**

**printf("\nPlease enter the size of an array:");**

**scanf("%d",&n);**

**a=(int\*)malloc(n\*sizeof(int));**

**printf("\nBEST CASE");**

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

**{**

**a[i]=i+1;**

**}**

**start=clock();**

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

**{**

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

**{**

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

**{**

**int t=a[j];**

**a[j]=a[j+1];**

**a[j+1]=t;**

**}**

**}**

**}**

**end=clock();**

**printf("Start time=%ld\n",start);**

**printf("End time=%ld\n",end);**

**double total\_cputime=(double)(end-start)/CLOCKS\_PER\_SEC;**

**printf("Total CPU Time in second=%f\n",total\_cputime);**

**printf("\n");**

**printf("\nWORST CASE");**

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

**{**

**a[i]=n-i;**

**}**

**start=clock();**

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

**{**

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

**{**

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

**{**

**int t=a[j];**

**a[j]=a[j+1];**

**a[j+1]=t;**

**}**

**}**

**}**

**end=clock();**

**printf("Start time=%ld\n",start);**

**printf("End time=%ld\n",end);**

**total\_cputime=(double)(end-start)/CLOCKS\_PER\_SEC;**

**printf("Total CPU Time in second=%f\n",total\_cputime);**

**printf("\n");**

**printf("FOR AVERAGE CASE\n");**

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

**{**

**a[i]=rand()%n;**

**}**

**start=clock();**

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

**{**

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

**{**

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

**{**

**int t=a[j];**

**a[j]=a[j+1];**

**a[j+1]=t;**

**}**

**}**

**}**

**end=clock();**

**printf("Start time=%ld\n",start);**

**printf("End time=%ld\n",end);**

**total\_cputime=(double)(end-start)/CLOCKS\_PER\_SEC;**

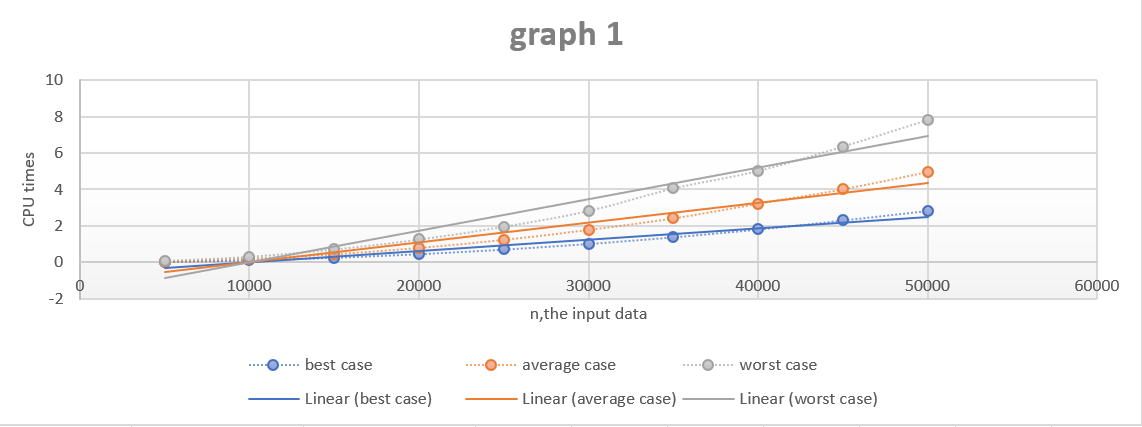
**printf("Total CPU Time in second=%f\n",total\_cputime);**

**}**

**Output**

****

**Graph**

****

1. **Write a C program for selection sort to**
2. **Compare the time complexity with the given data set given below and calculate the time complexity based on the CPU clock.**
3. **Plot a graph showing the comparison (n, the input data Vs. CPU times)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sl No.** | **Value of n** | **Selection Sort (Time Complexity)** | | |
| **Best case** | **Average case** | **Worst case** |
| **1** | **5000** | **0.033277** | **0.029980** | **0.030933** |
| **2** | **10000** | **0.118135** | **0.117758** | **0.123459** |
| **3** | **15000** | **0.264385** | **0.259134** | **0.277317** |
| **4** | **20000** | **0.458897** | **0.455815** | **0.484332** |
| **5** | **25000** | **0.713877** | **0.716458** | **0.754216** |
| **6** | **30000** | **1.019945** | **1.020666** | **1.080991** |
| **7** | **35000** | **1.388563** | **1.390047** | **1.466262** |
| **8** | **40000** | **1.810389** | **1.808287** | **1.918359** |
| **9** | **45000** | **2.292378** | **2.298355** | **2.433973** |
| **10** | **50000** | **2.848365** | **2.831862** | **2.993081** |

**Program**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<time.h>**

**void main()**

**{**

**int \*a,n;**

**clock\_t start,end;**

**printf("\nPlease enter the size of an array:");**

**scanf("%d",&n);**

**a=(int\*)malloc(n\*sizeof(int));**

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

**{**

**a[i]=i+1;**

**}**

**printf("\nFOR BEST CASE");**

**start=clock();**

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

**{**

**int s=a[i],p=i;**

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

**{**

**if(a[j]<s)**

**{**

**s=a[j];**

**p=j;**

**}**

**}**

**int t=a[i];**

**a[i]=a[p];**

**a[p]=t;**

**}**

**end=clock();**

**printf("Start time=%ld\n",start);**

**printf("End time=%ld\n",end);**

**double total\_cputime=(double)(end-start)/CLOCKS\_PER\_SEC;**

**printf("Total CPU Time in second=%f\n",total\_cputime);**

**printf("\n");**

**printf("\nFOR WORST CASE");**

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

**{**

**a[i]=n-i;**

**}**

**start=clock();**

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

**{**

**int s=a[i],p=i;**

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

**{**

**if(a[j]<s)**

**{**

**s=a[j];**

**p=j;**

**}**

**}**

**int t=a[i];**

**a[i]=a[p];**

**a[p]=t;**

**}**

**end=clock();**

**printf("Start time=%ld\n",start);**

**printf("End time=%ld\n",end);**

**total\_cputime=(double)(end-start)/CLOCKS\_PER\_SEC;**

**printf("Total CPU Time in second=%f\n",total\_cputime);**

**printf("\n");**

**printf("\nFOR AVERAGE CASE");**

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

**{**

**a[i]=rand()%n;**

**}**

**start=clock();**

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

**{**

**int s=a[i],p=i;**

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

**{**

**if(a[j]<s)**

**{**

**s=a[j];**

**p=j;**

**}**

**}**

**int t=a[i];**

**a[i]=a[p];**

**a[p]=t;**

**}**

**end=clock();**

**printf("Start time=%ld\n",start);**

**printf("End time=%ld\n",end);**

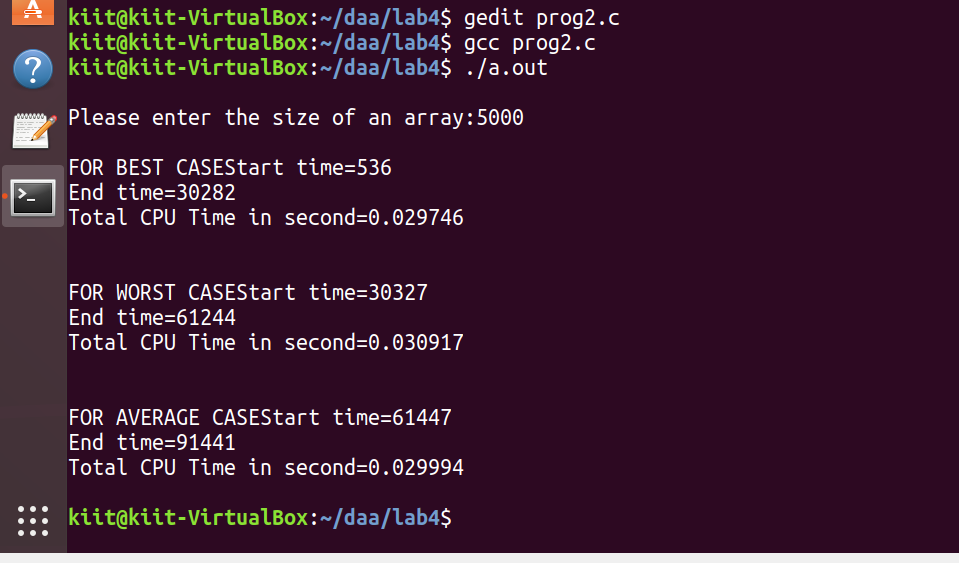
**total\_cputime=(double)(end-start)/CLOCKS\_PER\_SEC;**

**printf("Total CPU Time in second=%f\n",total\_cputime);**

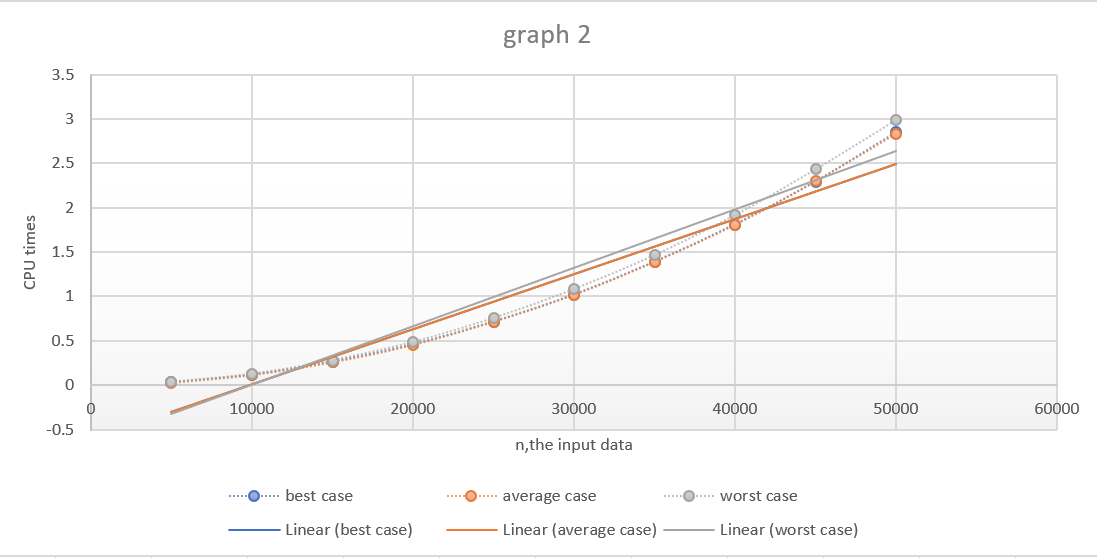
**printf("\n");**

**}**

**Output**

****

**Graph**

****