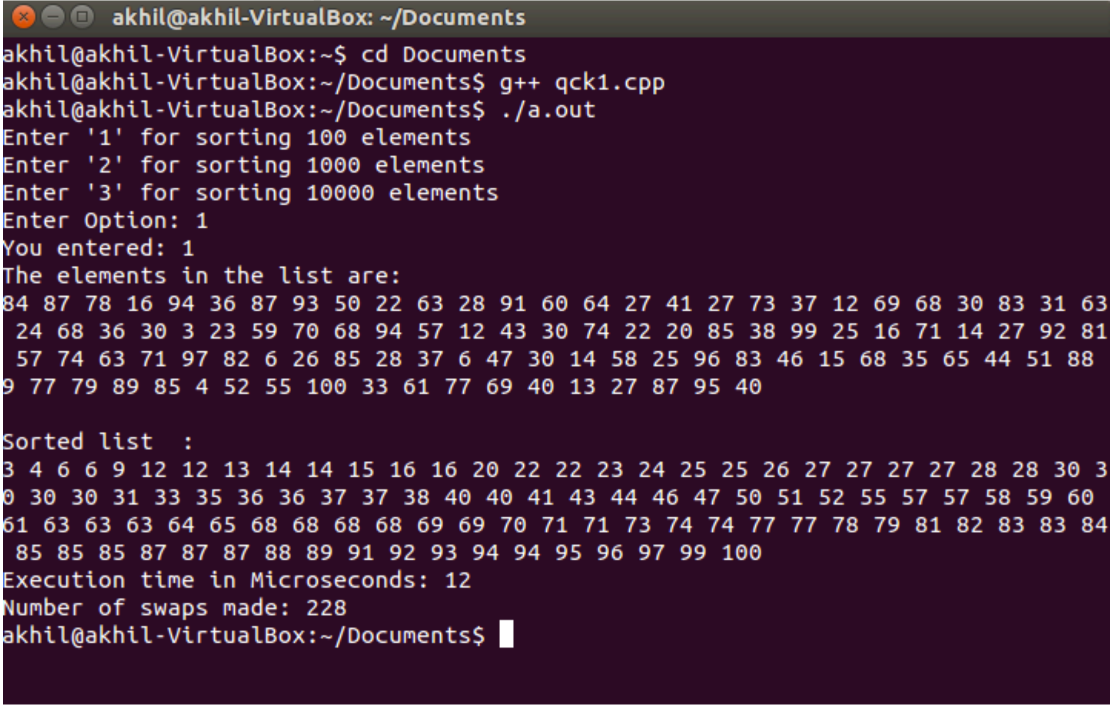
/*Akhil Pal*/
/*Sanman Pradhan*/
Spring 20

Quick sort & Merge sort Algorithm

1. Quick sort outputs.

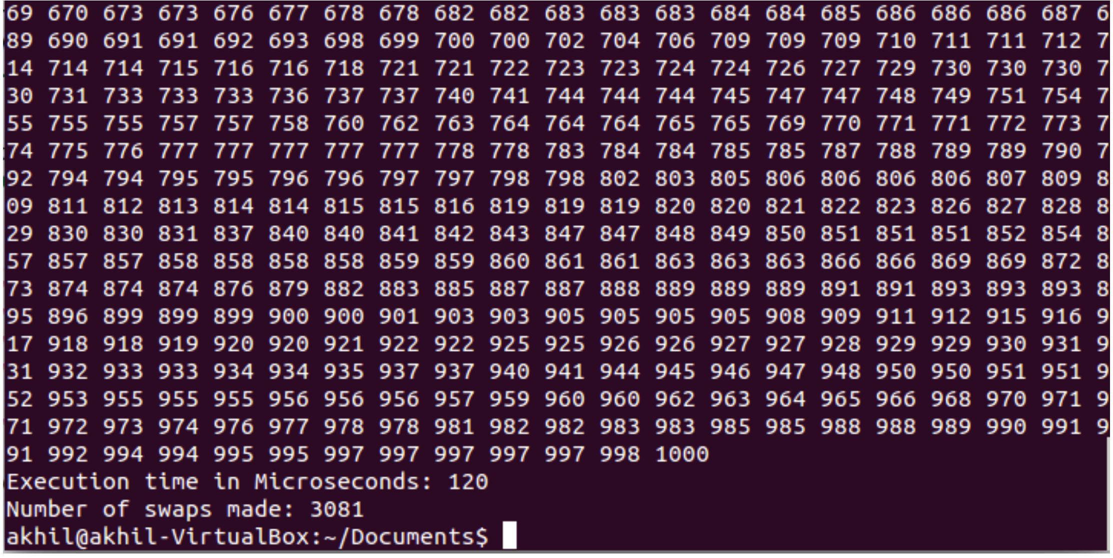
- For 100 inputs :



```
akhil@akhil-VirtualBox:~$ cd Documents
akhil@akhil-VirtualBox:~/Documents$ g++ qck1.cpp
akhil@akhil-VirtualBox:~/Documents$ ./a.out
Enter '1' for sorting 100 elements
Enter '2' for sorting 1000 elements
Enter '3' for sorting 10000 elements
Enter Option: 1
You entered: 1
The elements in the list are:
84 87 78 16 94 36 87 93 50 22 63 28 91 60 64 27 41 27 73 37 12 69 68 30 83 31 63
 24 68 36 30 3 23 59 70 68 94 57 12 43 30 74 22 20 85 38 99 25 16 71 14 27 92 81
 57 74 63 71 97 82 6 26 85 28 37 6 47 30 14 58 25 96 83 46 15 68 35 65 44 51 88
9 77 79 89 85 4 52 55 100 33 61 77 69 40 13 27 87 95 40

Sorted list  :
3 4 6 6 9 12 12 13 14 14 15 16 16 20 22 22 23 24 25 25 26 27 27 27 27 28 28 30 3
0 30 30 31 33 35 36 36 37 37 38 40 40 41 43 44 46 47 50 51 52 55 57 57 58 59 60
61 63 63 63 64 65 68 68 68 68 69 69 70 71 71 73 74 74 77 77 78 79 81 82 83 83 84
 85 85 85 87 87 87 88 89 91 92 93 94 94 95 96 97 99 100
Execution time in Microseconds: 12
Number of swaps made: 228
akhil@akhil-VirtualBox:~/Documents$ 
```

Number of swaps = 228.
Execution time in microseconds = 12

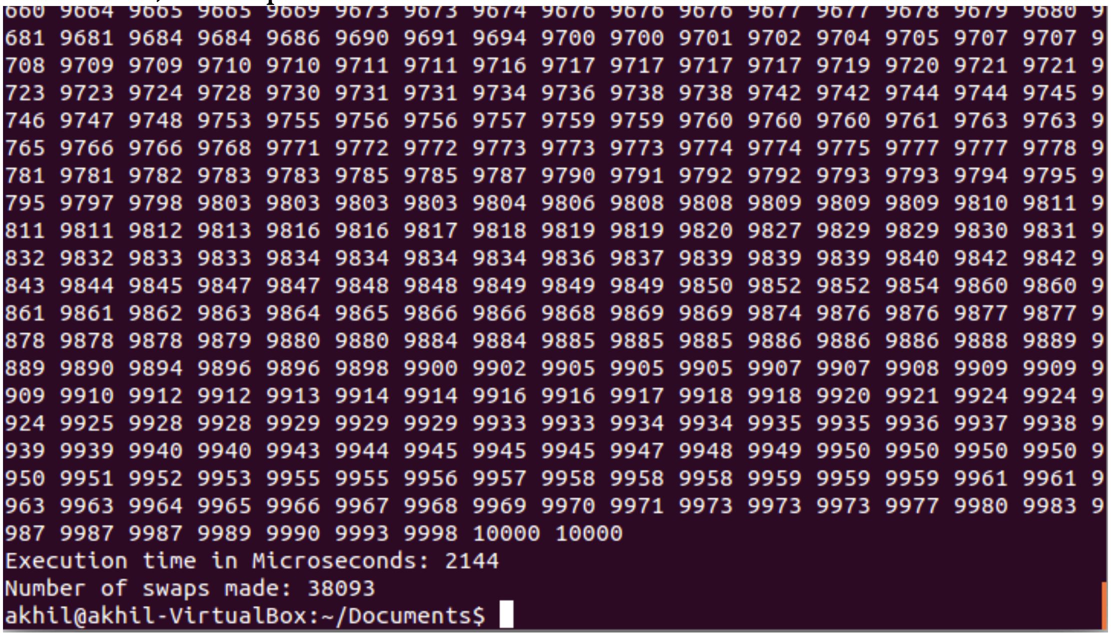
- For 1000 inputs



```
69 670 673 673 676 677 678 678 682 682 683 683 683 684 684 685 686 686 686 687 6
89 690 691 691 692 693 698 699 700 700 702 704 706 709 709 709 710 711 711 712 7
14 714 714 715 716 716 718 721 721 722 723 723 724 724 726 727 729 730 730 730 7
30 731 733 733 733 736 737 737 740 741 744 744 744 745 747 747 748 749 751 754 7
55 755 755 757 757 758 760 762 763 764 764 764 765 765 769 770 771 771 772 773 7
74 775 776 777 777 777 777 777 778 778 783 784 784 785 785 787 788 789 789 790 7
92 794 794 795 795 796 796 797 797 798 798 802 803 805 806 806 806 806 807 809 8
09 811 812 813 814 814 815 815 816 819 819 819 820 820 821 822 823 826 827 828 8
29 830 830 831 837 840 840 841 842 843 847 847 848 849 850 851 851 851 852 854 8
57 857 857 858 858 858 858 859 859 860 861 861 863 863 863 866 866 869 869 872 8
73 874 874 874 876 879 882 883 885 887 887 888 889 889 889 891 891 893 893 893 8
95 896 899 899 899 900 900 901 903 903 905 905 905 905 908 909 911 912 915 916 9
17 918 918 919 920 920 921 922 922 925 925 926 926 927 927 928 929 929 930 931 9
31 932 933 933 934 934 935 937 937 940 941 944 945 946 947 948 950 950 951 951 9
52 953 955 955 955 956 956 956 957 959 960 960 962 963 964 965 966 968 970 971 9
71 972 973 974 976 977 978 978 981 982 982 983 983 985 985 988 988 989 990 991 9
91 992 994 994 995 995 997 997 997 997 997 998 1000
Execution time in Microseconds: 120
Number of swaps made: 3081
akhil@akhil-VirtualBox:~/Documents$ 
```

Number of swaps = 3081
Execution time in microseconds =120

- For 10,000 inputs

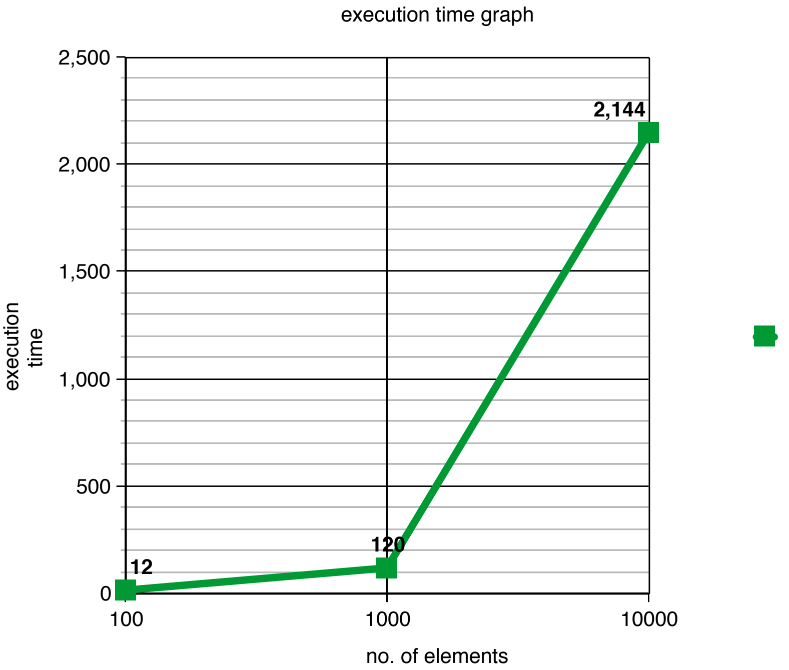


```
660 9664 9665 9665 9669 9673 9673 9674 9676 9676 9676 9677 9677 9678 9679 9680 9
681 9681 9684 9684 9686 9690 9691 9694 9700 9700 9701 9702 9704 9705 9707 9707 9
708 9709 9709 9710 9710 9711 9711 9716 9717 9717 9717 9717 9719 9720 9721 9721 9
723 9723 9724 9728 9730 9731 9731 9734 9736 9738 9738 9742 9742 9744 9744 9745 9
746 9747 9748 9753 9755 9756 9756 9757 9759 9759 9760 9760 9760 9761 9763 9763 9
765 9766 9766 9768 9771 9772 9772 9773 9773 9773 9774 9774 9775 9777 9777 9778 9
781 9781 9782 9783 9783 9785 9785 9787 9790 9791 9792 9792 9793 9793 9794 9795 9
795 9797 9798 9803 9803 9803 9803 9804 9806 9808 9808 9809 9809 9809 9810 9811 9
811 9811 9812 9813 9816 9816 9817 9818 9819 9819 9820 9827 9829 9829 9830 9831 9
832 9832 9833 9833 9834 9834 9834 9834 9836 9837 9839 9839 9839 9840 9842 9842 9
843 9844 9845 9847 9847 9848 9848 9849 9849 9849 9850 9852 9852 9854 9860 9860 9
861 9861 9862 9863 9864 9865 9866 9866 9868 9869 9869 9874 9876 9876 9877 9877 9
878 9878 9878 9879 9880 9880 9884 9884 9885 9885 9885 9886 9886 9886 9888 9889 9
889 9890 9894 9896 9896 9898 9900 9902 9905 9905 9905 9907 9907 9908 9909 9909 9
909 9910 9912 9912 9913 9914 9914 9916 9916 9917 9918 9918 9920 9921 9924 9924 9
924 9925 9928 9928 9929 9929 9929 9933 9933 9934 9934 9935 9935 9936 9937 9938 9
939 9939 9940 9940 9943 9944 9945 9945 9945 9947 9948 9949 9950 9950 9950 9950 9
950 9951 9952 9953 9955 9955 9956 9957 9958 9958 9958 9959 9959 9959 9961 9961 9
963 9963 9964 9965 9966 9967 9968 9969 9970 9971 9973 9973 9973 9977 9980 9983 9
987 9987 9987 9989 9990 9993 9998 10000 10000
Execution time in Microseconds: 2144
Number of swaps made: 38093
akhil@akhil-VirtualBox:~/Documents$
```

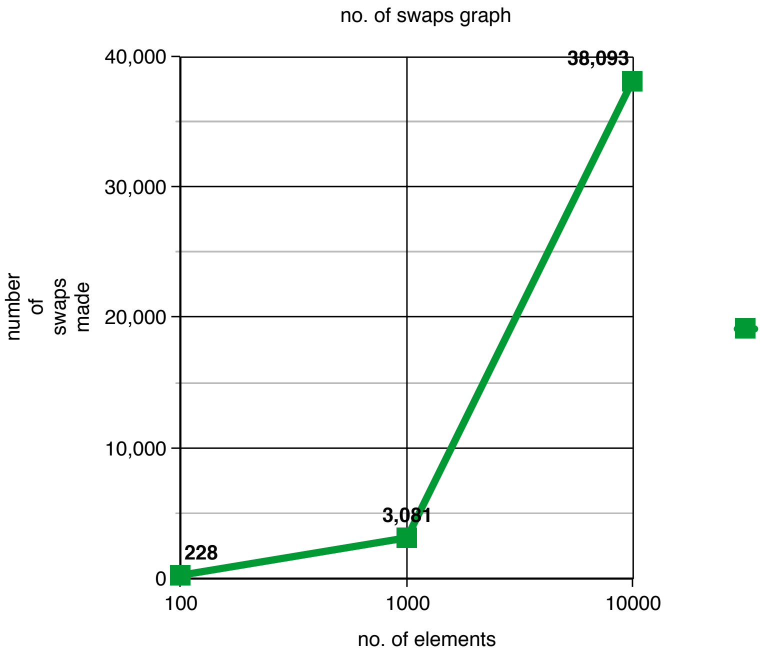
Number of swaps = 38093

Execution time in microseconds =2144

GRAPHS



EXECUTION TIME GRAPH



No. of swaps made .

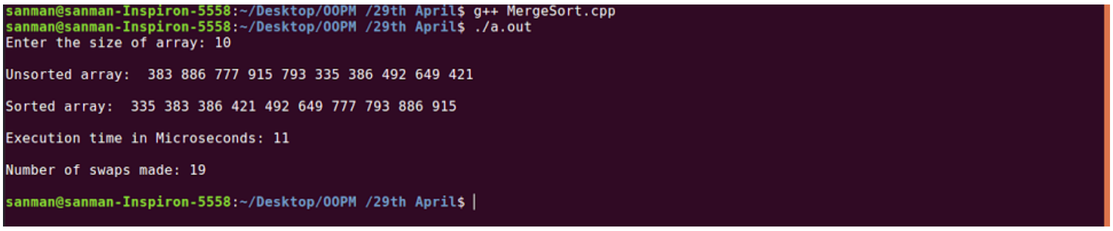
Quick sort program :

```
// quick sort program
#include <iostream>
#include<cstdlib>
 #include<ctime>
using namespace std;
void printArray(int* arr, int n); // function declaration
void qckSort(int* arr, int startIndex, int endIndex);  // function
declaration
int divideArray(int* arr, int pivotValue, int startIndex, int endIndex);  //
function declaration
void swap(int &a, int &b);  // function declaration
  int n,count;
 int main(void)   // main function
  { int num,t1,t2,t;
   cout<<"Enter '1' for sorting 100 elements"<<endl;
 cout<<"Enter '2' for sorting 1000 elements"<<endl;
 cout<<"Enter '3' for sorting 10000 elements"<<endl;
 cout<<"Enter Option: ";
 cin >> num;
  cout<<"You entered: "<<num<< endl;
 switch(num)      // switch case
 {     case 1:         n=100;         break;
 case 2:         n=1000;
break;
 case 3:         n=10000;
       break;
default :       n=10;
} int arr[n];            // array declaration with random input
 for (int i=0; i<n;i++)
 {
arr[i]=rand()%n+1; }
cout <<"The elements in the list are: "<<endl;
printArray(arr, n);
cout<<endl; t1=clock();
qckSort(arr,0,n-1);
 t2=clock();
t=(double)(t2-t1)/CLOCKS_PER_SEC*1000000.0;
 cout<<endl; cout<<"Sorted list  : "<<endl;
printArray(arr, n);
cout<<endl;
cout<<"Execution time in Microseconds: "<<t<<endl;
cout<<"Number of swaps made: "<<count<<endl;
return 0; }
 void swap(int &p, int &q)        // swap function
{ int temp;
temp = p;
 p = q;
q = temp; }
  void printArray(int* arr, int n)  // print function
  { int i;
for( i = 0; i < n; i++)
{ cout<<arr[i] << " "; }
}
void qckSort(int* arr, int startIndex, int endIndex) // sort function
{ int pivot = arr[startIndex];
    int divPoint;
if(endIndex > startIndex)
{
 divPoint = divideArray(arr, pivot, startIndex, endIndex);
arr[divPoint] = pivot;
 qckSort(arr, startIndex, divPoint-1);
qckSort(arr, divPoint+1, endIndex);
  }         }
 int divideArray(int* arr, int pivot, int startIndex, int endIndex) /*
function for splitting the list from the pivot point */
```

```
{
 int left = startIndex; int right = endIndex;
while(left < right)
       {
while( pivot < arr[right] && right > left)
{ right--;
 }
count++;
swap(arr[left], arr[right]);
while( pivot >= arr[left] && left < right)
 { left++;
         }
swap(arr[right], arr[left]);
  } count++;
return left;
      }
```

2. Merge sort outputs

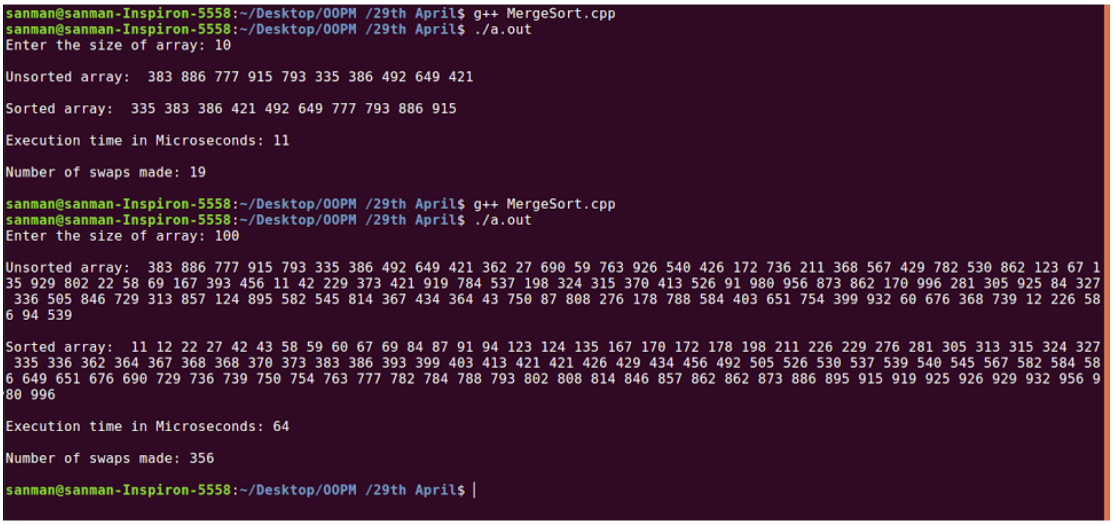
- For 100 inputs



```
sanman@sanman-Inspiron-5558:~/Desktop/OOPM /29th April$ g++ MergeSort.cpp
sanman@sanman-Inspiron-5558:~/Desktop/OOPM /29th April$ ./a.out
Enter the size of array: 10

Unsorted array:  383 886 777 915 793 335 386 492 649 421

Sorted array:  335 383 386 421 492 649 777 793 886 915

Execution time in Microseconds: 11

Number of swaps made: 19

sanman@sanman-Inspiron-5558:~/Desktop/OOPM /29th April$ |
```

Execution time = 11 microseconds
Number of swaps made = 19

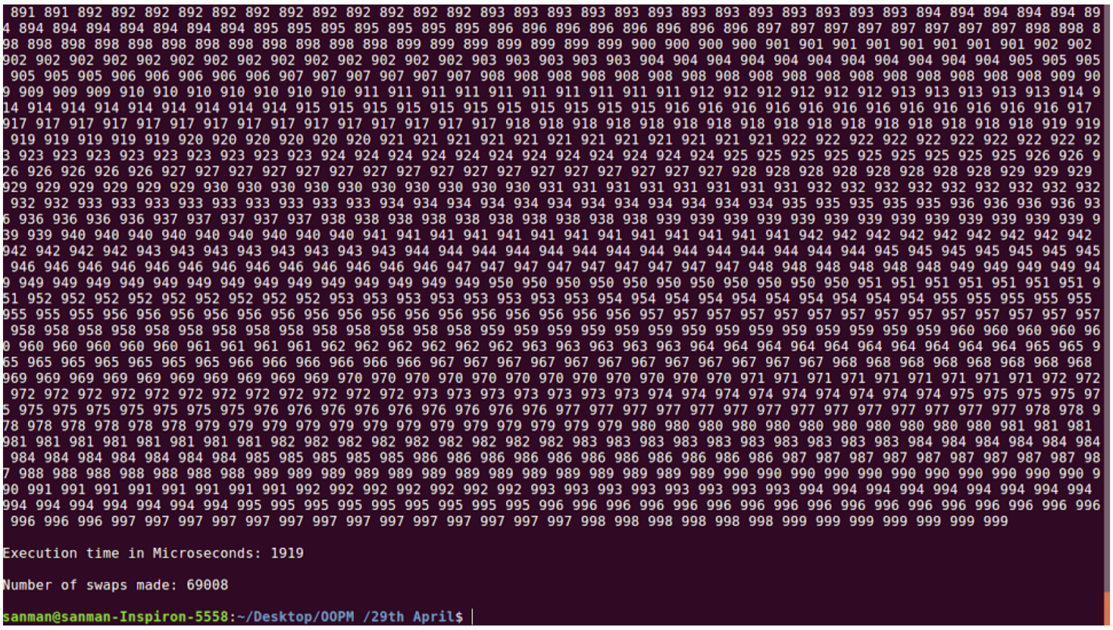
- For 1000 inputs



```
sanman@sanman-Inspiron-5558:~/Desktop/OOPM /29th April$ g++ MergeSort.cpp
sanman@sanman-Inspiron-5558:~/Desktop/OOPM /29th April$ ./a.out
Enter the size of array: 10

Unsorted array:  383 886 777 915 793 335 386 492 649 421

Sorted array:  335 383 386 421 492 649 777 793 886 915

Execution time in Microseconds: 11

Number of swaps made: 19

sanman@sanman-Inspiron-5558:~/Desktop/OOPM /29th April$ g++ MergeSort.cpp
sanman@sanman-Inspiron-5558:~/Desktop/OOPM /29th April$ ./a.out
Enter the size of array: 100

Unsorted array:  383 886 777 915 793 335 386 492 649 421 362 27 690 59 763 926 540 426 172 736 211 368 567 429 782 530 862 123 67 1
35 929 802 22 58 69 167 393 456 11 42 229 373 421 919 784 537 198 324 315 370 413 526 91 980 956 873 862 170 996 281 305 925 84 327
 336 505 846 729 313 857 124 895 582 545 814 367 434 364 43 750 87 808 276 178 788 584 403 651 754 399 932 60 676 368 739 12 226 58
6 94 539

Sorted array:  11 12 22 27 42 43 58 59 60 67 69 84 87 91 94 123 124 135 167 170 172 178 198 211 226 229 276 281 305 313 315 324 327
 335 336 362 364 367 368 368 370 373 383 386 393 399 403 413 421 421 426 429 434 456 492 505 526 530 537 539 540 545 567 582 584 58
6 649 651 676 690 729 736 739 750 754 763 777 782 784 788 793 802 808 814 846 857 862 862 873 886 895 915 919 925 926 929 932 956 9
80 996

Execution time in Microseconds: 64

Number of swaps made: 356

sanman@sanman-Inspiron-5558:~/Desktop/OOPM /29th April$ |
```

execution time in microseconds = 64
number of swaps = 356

- For 10,000 inputs



```
 891 891 892 892 892 892 892 892 892 892 892 892 892 892 893 893 893 893 893 893 893 893 893 893 893 893 893 894 894 894 894 894 89
4 894 894 894 894 894 894 894 895 895 895 895 895 895 895 896 896 896 896 896 896 896 896 897 897 897 897 897 897 897 897 898 898 8
98 898 898 898 898 898 898 898 898 898 898 898 899 899 899 899 899 899 899 900 900 900 900 901 901 901 901 901 901 901 901 902 902
902 902 902 902 902 902 902 902 902 902 902 902 902 902 903 903 903 903 903 904 904 904 904 904 904 904 904 904 904 904 905 905 905
 905 905 905 906 906 906 906 906 907 907 907 907 907 907 908 908 908 908 908 908 908 908 908 908 908 908 908 908 908 908 908 909 90
9 909 909 909 910 910 910 910 910 910 910 911 911 911 911 911 911 911 911 911 911 912 912 912 912 912 912 913 913 913 913 913 914 9
14 914 914 914 914 914 914 914 914 915 915 915 915 915 915 915 915 915 915 916 916 916 916 916 916 916 916 916 916 916 917
917 917 917 917 917 917 917 917 917 917 917 917 917 917 917 918 918 918 918 918 918 918 918 918 918 918 918 918 918 918 918 919 919
 919 919 919 919 919 920 920 920 920 920 920 921 921 921 921 921 921 921 921 921 921 921 921 922 922 922 922 922 922 922 922 922 92
3 923 923 923 923 923 923 923 923 923 924 924 924 924 924 924 924 924 924 924 924 925 925 925 925 925 925 925 925 925 926 926 9
26 926 926 926 926 927 927 927 927 927 927 927 927 927 927 927 927 927 927 927 927 927 928 928 928 928 928 928 928 928 929 929 929
929 929 929 929 929 929 930 930 930 930 930 930 930 930 930 930 931 931 931 931 931 931 931 931 932 932 932 932 932 932 932 932 932
 932 932 933 933 933 933 933 933 933 933 933 934 934 934 934 934 934 934 934 934 934 934 934 935 935 935 935 935 936 936 936 936 93
6 936 936 936 936 937 937 937 937 937 938 938 938 938 938 938 938 938 938 938 939 939 939 939 939 939 939 939 939 939 939 939 939 9
39 939 940 940 940 940 940 940 940 940 940 941 941 941 941 941 941 941 941 941 941 941 941 941 942 942 942 942 942 942 942 942 942
942 942 942 942 943 943 943 943 943 943 943 943 944 944 944 944 944 944 944 944 944 944 944 944 944 944 945 945 945 945 945 945 945
 946 946 946 946 946 946 946 946 946 946 946 946 946 947 947 947 947 947 947 947 947 947 948 948 948 948 948 948 949 949 949 949 94
9 949 949 949 949 949 949 949 949 949 949 949 949 949 949 950 950 950 950 950 950 950 950 950 950 950 951 951 951 951 951 951 951 9
51 952 952 952 952 952 952 952 952 952 953 953 953 953 953 953 953 953 954 954 954 954 954 954 954 954 954 954 955 955 955 955 955
955 955 955 956 956 956 956 956 956 956 956 956 956 956 956 956 956 956 956 957 957 957 957 957 957 957 957 957 957 957 957 957 957
 958 958 958 958 958 958 958 958 958 958 958 958 958 958 959 959 959 959 959 959 959 959 959 959 959 959 959 959 960 960 960 960 96
0 960 960 960 960 960 961 961 961 961 962 962 962 962 962 962 963 963 963 963 963 964 964 964 964 964 964 964 964 964 965 965 9
65 965 965 965 965 965 965 966 966 966 966 966 967 967 967 967 967 967 967 967 967 967 967 967 968 968 968 968 968 968 968 968
969 969 969 969 969 969 969 969 969 969 970 970 970 970 970 970 970 970 970 970 970 970 971 971 971 971 971 971 971 971 971 972 972
 972 972 972 972 972 972 972 972 972 972 972 972 973 973 973 973 973 973 973 974 974 974 974 974 974 974 974 974 975 975 975 975 97
5 975 975 975 975 975 975 975 976 976 976 976 976 976 976 976 977 977 977 977 977 977 977 977 977 977 977 977 977 977 978 978 9
78 978 978 978 978 978 979 979 979 979 979 979 979 979 979 979 979 979 979 980 980 980 980 980 980 980 980 980 980 980 981 981 981
981 981 981 981 981 981 981 981 982 982 982 982 982 982 982 982 982 983 983 983 983 983 983 983 983 983 983 984 984 984 984 984 984
 984 984 984 984 984 984 984 985 985 985 985 985 986 986 986 986 986 986 986 986 986 986 986 987 987 987 987 987 987 987 987 987 98
7 988 988 988 988 988 988 988 989 989 989 989 989 989 989 989 989 989 989 989 989 989 990 990 990 990 990 990 990 990 990 990 990 9
90 991 991 991 991 991 991 991 991 992 992 992 992 992 992 992 993 993 993 993 993 993 993 993 994 994 994 994 994 994 994 994 994
994 994 994 994 994 994 994 995 995 995 995 995 995 995 995 995 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996 996
 996 996 996 997 997 997 997 997 997 997 997 997 997 997 997 997 997 998 998 998 998 998 998 999 999 999 999 999 999 999

Execution time in Microseconds: 1919

Number of swaps made: 69008

sanman@sanman-Inspiron-5558:~/Desktop/OOPM /29th April$ |
```

Execution time =1919
Number of swaps = 69088

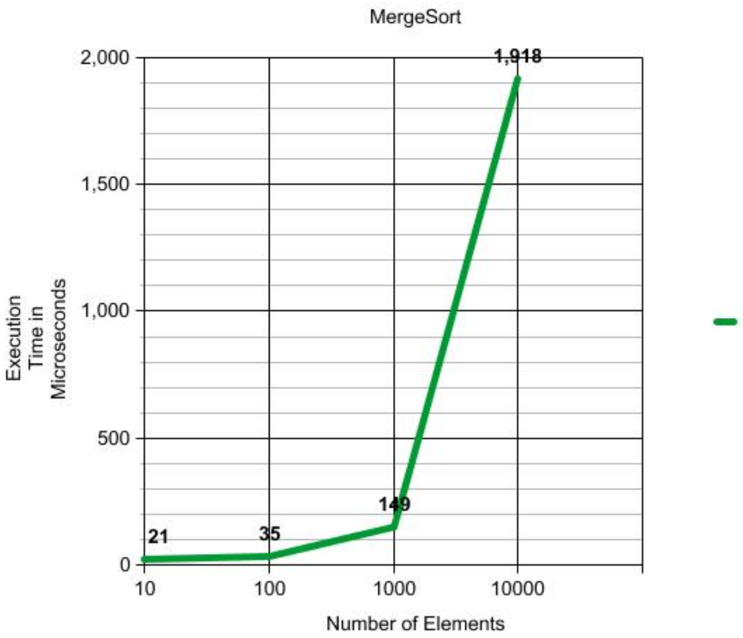
Program :

```
/*MergeSort*/
#include<stdio.h>
#include <iostream>
#include<stdlib.h>
```

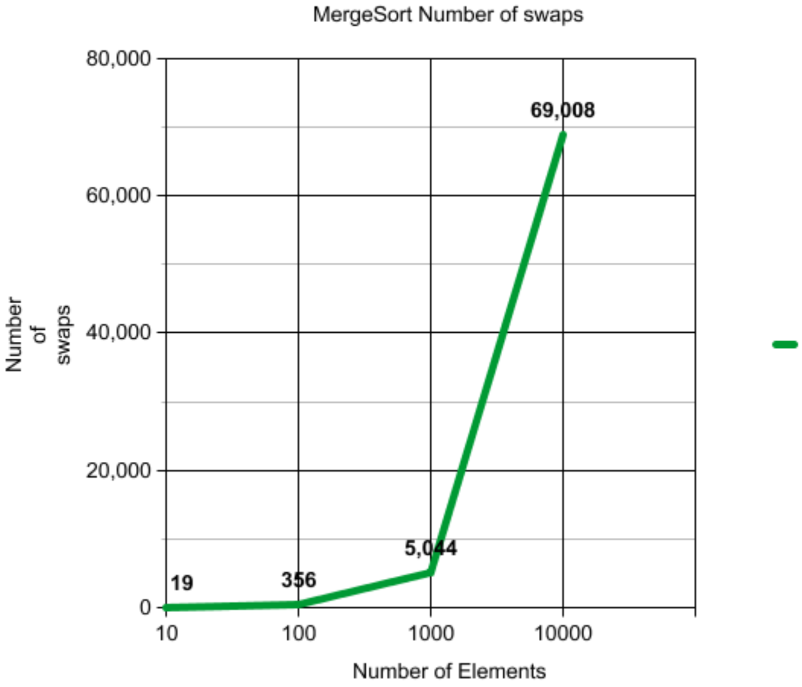
```
#include<stdio.h>
#include<time.h>
#include<math.h>


using namespace std;

int a[10000];       // array to be sorted
int count;


void merge(int a[],int l,int m,int h)
{
  int a1[10000],a2[10000];        // Two temporary arrays to hold the two arrays
to be merged
  int n1,n2,i,j,k;
  n1=m-l+1;
  n2=h-m;

  for(i=0; i<n1; i++)
    a1[i]=a[l+i];

  for(j=0; j<n2; j++)
    a2[j]=a[m+j+1];


  a1[i]=99999;
  a2[j]=99999;

  i=0;
  j=0;
  for(k=l; k<=h; k++)
{
    if(a1[i]<=a2[j])
      {
      a[k]=a1[i++];
       count++;}
    else{
      a[k]=a2[j++];
       //count++;
}
  }


}

void merge_sort(int a[],int left,int right)
{
  int centre;
  if(left<right)
{
    centre=(left+right)/2;
    merge_sort(a,left,centre);
    merge_sort(a,centre+1,right);
    merge(a,left,centre,right);
  }


}


int main()
{
  int n,i,t1,t2,t;

  cout<<"Enter the size of array: ";  // input the elements
  cin>>n;
  for(i=0; i<n; i++)
 {
     a[i]=rand()%1000;
  }
  cout<<endl;
  cout<<"Unsorted array: ";
  for(i=0; i<n; i++)
 {
  cout<<" "<<a[i];
 }
  t1=clock();
  merge_sort(a,0,n-1);  // sort the array
  t2=clock();
  t=(double)(t2-t1)/CLOCKS_PER_SEC*1000000.0;
  cout<<endl;
  cout<<endl;
  cout<<"Sorted array: ";  // print sorted array
  for(i=0; i<n; i++)
    cout<<" "<<a[i];
    cout<<endl;
    cout<<endl;
    cout<<"Execution time in Microseconds: "<<t<<endl;
    cout<<endl;
    cout<<"Number of swaps made: "<<count<<endl;
    cout<<endl;



  return 0;
}
```

Graph for merge sorts

- Execution time graph :



- Number of swaps graph .


MergeSort Number of swaps
Number of swaps
Number of Elements