

IT224 LAB ASSIGNMENT 1

Design and Analysis of Algorithms



Submitted By

Name : Deepjyoti Deka

Roll No : 190103014

Semester: 4th Semester , Branch : CSE

Experiment 1

Aim : To print “Hello World” using C++.

Objective : To write a program that prints the string “Hello World” using inbuilt standard output function.

Theory

This program prints hello world, cout is used to display text on screen, '\n' places cursor on the beginning of next line, iostream header file contains declaration of Standard Input / Output Streams Library. The code will work on all operating systems may be it's Linux, Mac or any other and compilers.

Source Code

```
#include <iostream>
using namespace std;

int main()
{
    cout<<"Hello World \n";
}
```

Output

```
deep@deep-Inspiron-15-3567:~/4thSem/DAA$ g++ helloworld.cpp
deep@deep-Inspiron-15-3567:~/4thSem/DAA$ ./a.out
Hello World
deep@deep-Inspiron-15-3567:~/4thSem/DAA$
```

Result

The “Hello world” string is printed.

Experiment 2

Aim : To copy a string without using strcpy()

Objective

- To illustrate how to copy one string to another.
- To show this without the use of strcpy ()

Theory

Strcpy can be used to copy one string to another. As C strings are character arrays. We must pass character array, or pointer to character array to this function where string will be copied.. But here we are giving one string in input and then with the help of a for-loop we transfer the content of first array to the second array. The standard output function is used to print message on the screen. It is displayed using the inbuilt header file for input/output.

Source Code

```
#include <iostream>
using namespace std;

int main()
{
    int index = 0;
    char copy[40];
    char original[40];
    cout << "Enter the string to be copied! \n";
    cin >> original ;
    while(original[index] != '\0'){
        copy[index] = original[index];
        index++;
    }
    cout << "The copied string is \n";
    cout << copy << endl ;
}
```

Output

```
deep@deep-Inspiron-15-3567:~/4thSem/DAA$ ./a.out
Enter the string to be copied!
enter
The copied string is
enter
deep@deep-Inspiron-15-3567:~/4thSem/DAA$ █
```

Result

The word “enter” is copied by just using while loop.

Experiment 3

Aim : To concatenate two strings.

Objective : To show the process of combining two strings stored in two different variables.

Theory

The concatenation of two strings is done to form one string by appending the second string to the right-hand side of the first string. The strings are user input and the final combined string is displayed using the standard output function.

Source Code

```
#include <iostream>
using namespace std;

int main()
{
    char first[50];
    char second[50];
    char concat[50];

    cout << "enter first" <<endl;

    cin >> first;

    cout << "enter second" <<endl;

    cin >> second;

    int i , j;

    for( i=0;first[i]!='\0';i++){
        concat[i] = first[i];
    }
    for(int j=0;second[j]!='\0';j++)
    {
```

```
        concat[i] = second[j];  
        i++;  
    }  
  
    cout << "The joined string is \n";  
    cout << concat << endl ;  
  
}
```

Output

```
deep@deep-Inspiron-15-3567:~/4thSem/DAA$ g++ string_concat_using_for.cpp  
deep@deep-Inspiron-15-3567:~/4thSem/DAA$ ./a.out  
enter first  
Gau  
enter second  
hati  
The joined string is  
Gauhati  
deep@deep-Inspiron-15-3567:~/4thSem/DAA$ █
```

Result

The strings Gau and hati are concatenated into Gauhati.

Experiment 4

Aim : To delete a character from an input string.

Objective

- To take two character as an input from a user.
- Delete the given character.
- Output the char variable after deletion using a for-loop.

Theory

The logic behind the algorithm is to iterate through the string and find the specified character to delete. The string array and character are taken as inputs. The deletion algorithm iterates through the array until it finds desired character, and then shifts the rest of the array one index towards left.

Source Code

```
#include <iostream>
using namespace std;

int main()
{
    char word[100],x;

    cout<<"enter the word \n";
    cin>> word;

    cout<<"Enter the character to delete \n";
    cin >> x;

    for (int i = 0 ; word[i]!='\0';i++)
    {
        if(word[i] == x){
```

```
        while(word[i+1] != '\\0'){

            word[i] = word[i+1];
            i++;
        }

        word[i] = '\\0';
        break;
    }

    cout<< word << "\\n";
}
```

Output

```
deep@deep-Inspiron-15-3567:~/4thSem/DAA$ g++ removeCharacterFromString.cpp
deep@deep-Inspiron-15-3567:~/4thSem/DAA$ ./a.out
enter the word
HELLO
Enter the character to delete
E
HLLO
deep@deep-Inspiron-15-3567:~/4thSem/DAA$
```

Result

The character E is deleted from the Hello resulting in HLLO.

Experiment 5

Aim : To delete a substring from a given string.

Objective

- Take string and substring (to delete) inputs from user.
- Formulate an algorithm to remove the substring from the string.
- Shifting the elements left to overwrite the substring.

Theory

The goal of this experiment is to develop an algorithm to remove a given substring from a string. An approach towards this problem is shifting of elements to overwrite the substring.

The process involves iterating through the string in a loop until it finds the first matching letter of both the strings. Then another loop is executed to continue iterating until it is confirmed that all the corresponding letters of the substring are also present after the first match. If it is present, then that part of the string is overwritten by shifting the elements towards left, from the index after the end of the substring in the string.

Source Code

```
#include <iostream>
using namespace std;

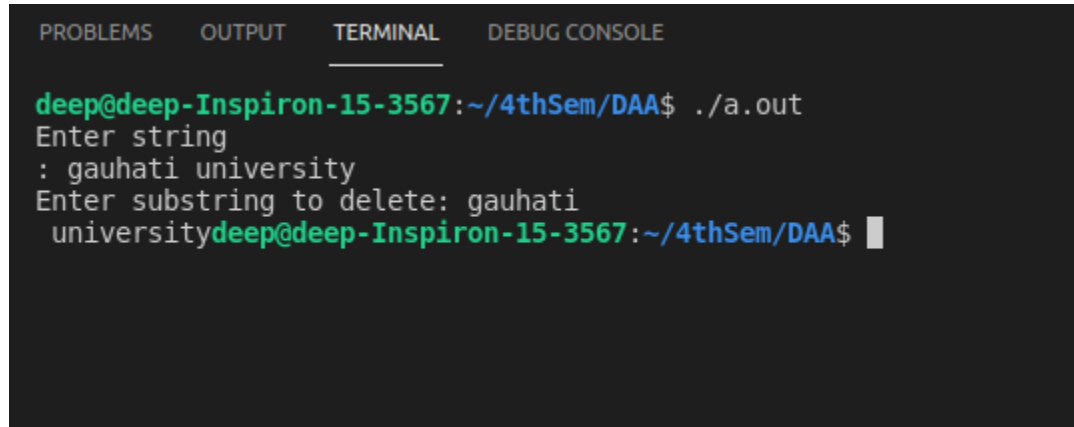
int get_size(char del[])
{
    int x = 0;
    for (int i = 0; del[i] != '\0'; i++)
    {
        x++;
    }
    return x;
}

int main()
{
    char s[20], del[10];
```

```
cout << "Enter string \n: ";
cin.getline(s, 20);
cout << "Enter substring to delete: ";
cin >> del;
int elem_size = get_size(del);
int j = 0, count = 0;

for (int i = 0; s[i] != '\0'; i++)
{
    if (s[i] == del[j])
    {
        count++;
        j++;
        if (count == elem_size)
        {
            i = i - elem_size + 1;
            while (s[i + elem_size] != '\0')
            {
                s[i] = s[i + elem_size];
                i++;
            }
            s[i] = '\0';
            break;
        }
    }
    else
    {
        j = 0;
        count = 0;
    }
}
cout << s;
}
```

Output



```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

deep@deep-Inspiron-15-3567:~/4thSem/DAA$ ./a.out
Enter string
: gauhati university
Enter substring to delete: gauhati
universitydeep@deep-Inspiron-15-3567:~/4thSem/DAA$
```

Result

The string gauhati is deleted from the gauhati university resulting in university.

Experiment 6

Aim : To multiply two matrices.

Objective

- Using the logic of matrix multiplication, the product of two user inputted matrices is found.
- 3 for-loops are implemented for traversing/iterating through the rows and columns of the matrices.

Theory

In matrix multiplication, row elements of the first matrix is multiplied by the column elements of the second matrix. To multiply two matrices, the number of columns of the first matrix should be equal to the number of rows to the second matrix. The matrices are taken as inputs from the user.

For the multiplication, 3 for loops are used. The innermost for loop iterates throughout the current row and column of the first and second matrices respectively and adds the products of the corresponding elements. The penultimate loop is used for iterating throughout the columns of the second matrix, while outermost result is used for iterating throughout the rows of the first matrix.

Source Code

```
#include <iostream>
using namespace std;

int main()
{
    int a[10][10] , b[10][10] , c[10][10];
    int row , column;

    cout << "No of rows ? \n";
    cin>> row;
    cout << "No of column ? \n";
    cin>> column;
    cout << "Enter the first element \n";
```

```

for(int i = 0 ; i < row ; i++){
    for(int j=0 ; j < column ; j++){
        cin>> a[i][j];
    }
}

cout << "Enter the second element \n";

for(int i = 0 ; i < row ; i++){
    for(int j=0 ; j < column ; j++){
        cin>> b[i][j];
    }
};

cout << "Multiply the matrix \n";

for(int i=0 ; i< row ; i++){
    for(int j=0 ; j < column; j++){

        c[i][j] = 0 ;

        for(int k=0 ; k<column ; k++){

            c[i][j] += a[i][k] * b[k][j];
        }
    }
};

cout << "printing the result\n";

for(int i=0 ; i<row ; i++){
    for(int j=0; j<column ; j++){
        cout << c[i][j] << " " ;
    }
    cout << "\n";
}
}

```

Output

```
deep@deep-Inspiron-15-3567:~/4thSem/DAA$ ./a.out
No of rows ?
2
No of column ?
2
Enter the first element
1
2
3
4
Enter the second element
1
2
3
4
Multiply the matrix
printing the result
7 10
15 22
deep@deep-Inspiron-15-3567:~/4thSem/DAA$ █
```

Result

The matrix a and b are multiplied using the matrix multiplication rules.

Experiment 7

Aim : To find the determinant of a universal matrix.

Objective

- Illustrating the use of recursion to find determinant of the sub matrices (minors).
- Using the inbuilt mathematical header file for computation of powers for getting cofactors of the reference row/column.

Theory

The value of determinant of a matrix can be calculated by following procedure –for each element of first row or first column get cofactor of those elements and then multiply the element with the determinant of the corresponding cofactor, and finally add them with alternate signs. As a base case the value of determinant of a 1*1 matrix is the single value itself.

Cofactor of an element, is a matrix which we can get by removing row and column of that element from that matrix.

Source Code

```
#include <iostream>
using namespace std;

int det(int n, int mat[10][10])
{
    int ans = 0;
    int sign = -1;
    int minor[10][10];
    if (n == 2){
        return ((mat[0][0] * mat[1][1]) - (mat[1][0] * mat[0][1]));
    }
    else
    {
        for (int i = 0; i < n; i++)
        {
            int row = 0;
```

```

        for (int j = 1; j < n; j++)
        {
            int col = 0;
            for (int k = 0; k < n; k++)
            {
                if (k == i)
                    continue;
                minor[row][col] = mat[j][k];
                col++;
            }
            row++;
        }
        sign = sign * -1 ;
        ans = ans + ((sign) * mat[0][i] * det(n - 1, minor));
    }
    return ans;
}

int main()
{
    int order;
    cout << "Enter order of the matrix \n";
    cin >> order;
    int a[10][10];
    cout << "Enter the elements:\n";
    for (int i = 0; i < order; i++)
    {
        for (int j = 0; j < order; j++)
        {
            cin >> a[j][i];
        }
    }
    cout << "The determinant is \n ";
    cout << det(order, a) << "\n";
    return 0;
}

```


Output

```
deep@deep-Inspiron-15-3567:~/4thSem/DAA$ ./a.out
Enter order of the matrix
3
Enter the elements:
1
2
3
4
5
6
7
8
9
The determiant is
0
deep@deep-Inspiron-15-3567:~/4thSem/DAA$ █
```

Result

The determinant of the matrix is calculated using the rules recursively.

Experiment 8

Aim : To implement KMP search algorithm for pattern searching.

Objective :

- To illustrate the use of KMP search algorithm for finding out the number of times a pattern has occurred in a string, both of which are inputted by the user.

Theory

Pattern searching is an important problem in computer science. When we do search for a string in notepad/word file or browser or database, pattern searching algorithms are used to show the search results.

The Naive pattern searching algorithm doesn't work well in cases where we see many matching characters followed by a mismatching character.

The KMP (Knuth Morris Pratt) matching algorithm uses degenerating property (pattern having same sub-patterns appearing more than once in the pattern) of the pattern and improves the worst case complexity to $O(n)$. The basic idea behind KMP's algorithm is: whenever we detect a mismatch (after some matches), we already know some of the characters in the text of the next window. We take advantage of this information to avoid matching the characters that we know will anyway match.

Source Code

```
#include <iostream>
using namespace std;

int kmp(char arr[], char pattern[], int p)
{
    int count = 0, i = 0, j = 0, flag;
    while (arr[i] != '\0')
    {
        j = 0;
        flag = 0;

```

```

        if (pattern[j] == arr[i])
        {
            while (pattern[j] == arr[i])
            {
                flag++;
                if (flag == p)
                {
                    count++;
                }

                i++;
                j++;
            }
        }
        if (j > 0)
        {
            continue;
        }
        else
        {
            i++;
        }
    }
    return (count);
}

int main()
{
    char arr[10], pattern[10];
    int patternSize = 0;
    int answer;
    cout << "Enter the string: "<<endl;
    cin >> arr;
    cout << "Enter the pattern: "<<endl;
    cin >> pattern;

    for (int i = 0; pattern[i] != '\0'; i++)
    {

```

```
        patternSize++;  
    }  
  
    answer = kmp(arr, pattern, patternSize);  
    cout << answer << endl;  
}
```

Output

```
deep@deep-Inspiron-15-3567:~/4thSem/DAA$ ./a.out  
Enter the string:  
asdsadasd  
Enter the pattern:  
ds  
1  
deep@deep-Inspiron-15-3567:~/4thSem/DAA$
```

Result

The algorithm successfully detects the pattern from the given string.

Experiment 9

Aim : To implement Array rotation & Array reverse functions.

Objective

- To shift the array towards left to the start from a given index as option 1.
- To reverse the array as option 2.
- Do-while loop implementation for repetition of the choice menu.
- Taking inputs from user.

Theory

Array rotation is basically the process of moving the positions of the elements towards one side. Left rotation means shifting towards left and vice versa. A simple approach towards formulating an algorithm for this problem would be to take a new array start filling the elements from the index inputted by the user in case of left rotation, which is currently what's aimed to achieve here.

Array Reverse is simply a process of reversing the arrangement of elements in an array. The algorithm for reversing the array involves using start and end as the pivot to move towards the middle while swapping the corresponding elements.

Source Code

```
#include<iostream>

using namespace std;

void leftRotateOne(int arr[], int n){
    int temp = arr[0], i;
    for (i = 0; i < n -1;i++){
        arr[i] = arr[i+1];
    }
    arr[n-1] = temp;
}

void leftRotate(int arr[], int d, int n){
    for (int i = 0; i < d; i++){
        leftRotateOne(arr,n);
    }
}
```

```

}

void leftRotation(int arr[], int d, int n){
    leftRotate(arr, d, n);
    for (int i=0;i<n;i++){
        cout << arr[i] << " ";
    }
}

int main(){

    int n=1, d , arr[n];
    cout << "Enter array size : " << endl;
    cin >> n;
    cout << "Enter the elements: " << endl;
    for (int i=0; i < n ;i ++ ){
        cin >> arr[i];
    }

    cout << "Before rotation: " << endl;
    for (int i=0;i<n;i++){
        cout << arr[i] << " ";
    }

    cout << "\n";

    cout << "Enter the position to rotate: " << endl;
    cin >> d;
    if (d == n){
        cout << "Enter position less than size of the array" << endl;
    }
    else{
        leftRotation(arr, d, n);
    }
}

```

Output

```
deep@deep-Inspiron-15-3567:~/4thSem/DAA$ ./a.out
Enter array size :
4
Enter the elements:
1
2
3
4
Before rotation:
1 2 3 4
Enter the position to rotate:
2
3 4 1 2 deep@deep-Inspiron-15-3567:~/4thSem/DAA$
```

Result

The rotation of the array 1 2 3 4 is applied at index 2 and result is 3 4 2 1 as intended.

Experiment 10

Aim : To find size versus time graph mapping for Quicksort algorithm.

Objective

- To write a program to print the time required by a quicksort algorithm to sort the elements.
- Repeat the experiment for different values of n - the number of elements in the list to be sorted.
- Plot a graph of the Size versus Time taken.
- Use of a time recording library to capture start and end time of a process.

Theory

Quicksort is a sorting algorithm based on the divide and conquer approach where an array is divided into sub-arrays by selecting a random pivot element (element selected from the array). While dividing the array, the pivot element should be positioned in such a way that elements less than pivot are kept on the left side and elements greater than pivot are on the right side of the pivot. The left and right sub-arrays are also divided using the same approach. This process continues until each sub-array contains a single element. At this point the array will be sorted. The mapping of the size versus time is a visualisation of the increasing time with increasing size of the input size. The values of the times taken would differ from pc to pc.

Source Code

```
#include <iostream>
#include <chrono>
using namespace std;
using namespace std::chrono;

void swap(int *a, int *b)
{
    int t = *a;
    *a = *b;
    *b = t;
}

int partition(int arr[], int left, int right)
{
    int pivot = arr[right];
```



```

    int i = (left - 1);

    for (int j = left; j < right; j++)
    {

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

    swap(&arr[i + 1], &arr[right]);
    return (i + 1);
}

void quickSort(int arr[], int left, int right)
{
    if (left < right)
    {
        int pivot = partition(arr, left, right);

        quickSort(arr, left, pivot - 1);
        quickSort(arr, pivot + 1, right);
    }
}

int graphtime(int arr[], int n)
{
    auto start = high_resolution_clock::now();
    quickSort(arr, 0, n - 1);
    auto end = high_resolution_clock::now();
    auto time_taken = duration_cast<microseconds>(end - start);
    return (time_taken.count());
}

void graph()
{
    int size[] = {10, 100, 1000, 10000};

```

```

int x[4], k = 0;

for (int n = 0; n < 4; n++)
{
    int arr[size[n]];
    for (int j = 0; j < n; j++)
    {
        arr[j] = rand() % size[n];
    }
    x[k] = graphtime(arr, size[n]);
    k++;
}

for (int i = 0; i < 4; i++)
{
    cout << size[i] << " --> " << x[i] << "ms" << endl;
}
}

int main()
{
    int array[1000];

    for (int i = 0 ; i<999 ; i++){
        array[i] = rand() % 1000;
    }

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

    auto start = high_resolution_clock::now();
    quickSort(array, 0, n - 1);
    auto end = high_resolution_clock::now();
    auto timetaken = duration_cast<microseconds>(end-start);

    cout << "Sorted array: \n";

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

```

```

        cout << array[i] << " ";
        cout << endl;

        cout << "\nThe time taken is " << timetaken.count() << "ms"<<endl;

        cout<<"the graph is \n";

        graph();
    }

```

Output

```

deep@deep-Inspiron-15-3567:~/4thSem/DAA$ ./a.out
Sorted array:
0 0 0 2 2 4 6 8 9 10 11 11 12 16 17 17 18 19 21 21 21 21 22 23 25 27 27 28 29 30 30 30 31 32 33 34 36 36 36 39 40 41 42 42 42 43 43 46 47
49 49 50 51 52 53 54 55 57 57 57 58 59 59 59 60 62 63 67 67 69 71 71 72 73 74 74 80 81 81 81 81 81 82 82 84 84 84 84 86 87 87 88 90 90 90
91 93 94 95 97 97 97 98 99 100 100 100 105 107 109 109 114 115 116 117 117 120 121 122 122 123 124 124 125 126 126 127 127 127 127 128 130 131
131 133 134 134 135 136 137 139 139 142 143 144 147 149 149 150 151 152 153 154 154 155 157 157 157 159 159 163 167 168 170 171 171 172 1
72 172 173 176 177 178 179 179 179 181 181 181 183 184 188 189 189 189 190 190 190 193 195 195 196 197 197 198 199 199 202 202 204 205 205
205 206 207 209 210 211 211 211 214 215 217 217 218 218 219 222 223 224 224 225 226 226 227 228 228 228 229 231 231 231 232 232 233 234 2
34 235 235 236 237 237 238 244 245 248 249 250 253 253 254 255 255 258 259 260 261 262 262 266 266 266 269 269 269 270 270 270 272 273 274 275
276 278 278 280 281 281 282 283 283 284 285 285 286 289 289 290 291 291 292 292 292 294 296 297 297 298 298 299 299 301 301 303 303 304 3
05 305 306 308 308 309 309 310 311 312 312 313 314 314 315 317 320 321 321 324 324 324 325 325 326 327 328 333 334 335 335 335 336 336 336
337 338 338 338 339 340 340 340 340 342 342 343 343 346 346 347 348 348 350 350 350 353 353 355 355 355 358 358 360 360 362 363 363 364 365 3
65 367 367 367 368 368 368 368 368 369 370 372 373 375 376 376 377 378 378 379 379 379 379 379 382 382 383 385 386 386 388 390 390 393 393
395 396 398 398 399 403 403 404 404 404 407 412 412 412 413 414 415 415 416 417 417 418 421 421 422 422 422 425 426 426 427 428 428 428 429 4
29 431 432 433 433 434 434 435 436 437 437 438 440 441 443 443 444 444 445 445 451 452 456 457 458 459 460 460 462 464 465 466 467 468 470
471 474 476 478 479 481 483 483 485 486 486 487 488 490 490 491 491 492 492 493 494 497 497 498 499 499 500 500 500 503 503 504 504 5
05 505 506 506 506 507 512 516 516 518 518 521 522 522 524 524 525 525 526 528 528 528 528 528 529 529 529 530 532 532 535 536 537 537 538
538 539 539 540 541 542 542 543 545 550 551 551 552 552 555 555 556 559 560 560 563 563 566 566 567 567 567 567 567 568 569 569 570 570 571 5
73 573 574 574 577 578 579 580 581 582 582 583 584 586 586 587 587 589 590 590 590 593 593 593 595 596 599 600 601 603 604 605 606 606 607
610 610 611 613 613 614 614 618 618 618 619 620 621 621 622 622 624 624 625 626 626 627 629 630 631 633 637 637 639 639 640 640 641 641 6
43 644 647 647 648 649 651 651 652 653 657 657 658 659 660 661 661 667 668 669 672 672 675 676 677 677 681 681 682 682 682 683 683 684 685
685 685 686 688 689 690 690 691 692 697 698 699 699 701 703 705 708 708 708 709 710 710 711 713 713 713 714 715 715 717 720 720 721 722 7
22 723 723 725 726 728 729 729 729 729 730 732 732 732 735 736 736 739 740 743 743 743 744 746 746 747 748 750 753 754 754 754 756 756 757
759 761 762 763 763 763 764 764 768 769 770 770 771 772 773 774 775 776 776 776 776 777 777 782 783 783 784 784 786 787 788 788 789 7
91 793 793 794 794 795 795 796 796 797 797 801 802 804 805 805 805 805 806 808 808 810 811 812 813 813 814 814 815 818 818 818 819 819 820
821 822 825 826 827 828 829 829 830 836 839 839 840 841 842 846 846 847 848 849 850 850 850 851 853 856 856 856 857 857 857 857 858 858 8
59 860 860 862 862 862 865 865 868 868 871 872 873 873 873 875 878 881 882 884 886 886 887 888 888 888 890 890 892 892 892 892 894 895 898 898
898 899 899 900 902 902 904 904 904 904 907 908 910 911 914 915 916 917 917 918 919 919 920 921 921 924 924 925 925 926 926 927 928 928 9
29 930 930 931 932 932 933 933 934 936 936 939 940 943 944 945 946 947 949 949 950 950 951 952 954 954 954 955 955 955 956 958 959 961
962 963 964 965 967 969 970 970 971 972 973 975 976 977 977 980 981 981 982 982 984 984 987 987 988 989 990 990 991 993 993 994 994 996 9
96 996 996 996 997 999

The time taken is 742ms
the graph is
10 --> 4ms
100 --> 61ms
1000 --> 3826ms
10000 --> 86302ms
deep@deep-Inspiron-15-3567:~/4thSem/DAA$

```

Result

1000 random elements are sorted and the graph of 10 , 100 , 1000 , 10000 elements is plotted..