|  |  |
| --- | --- |
| **Name** | **Dhruvil Doshi** |
| **UID** | **2021300027** |
| **Subject** | **Data Analysis Algorithm** |
| **Experiment No** | **1A & 1B** |

**Aim-**

1-A : To implement the various functions e.g. linear, non-linear, quadratic, exponential etc.

1-B : Experiment on finding the running time of an algorithm.

**Algorithm-**

1. **Insertion sort-**
   1. procedure insertionSort(A: list of sortable items)
   2. n = length(A)
   3. for i = 1 to n - 1 do
   4. j = i
   5. while j > 0 and A[j-1] > A[j] do
   6. swap(A[j], A[j-1])
   7. j = j - 1
   8. end while
   9. end for
   10. end procedure
2. **Selection sort-**
   1. Repeat Steps b and c for i = 0 to n-1
   2. CALL SMALLEST(arr, i, n, pos)
   3. SWAP arr[i] with arr[pos]
   4. [END OF LOOP]
   5. EXIT

* 1. SMALLEST (arr, i, n, pos)
  2. [INITIALIZE] SET SMALL = arr[i]
  3. [INITIALIZE] SET pos = i
  4. Repeat for j = i+1 to n
  5. if (SMALL > arr[j])
  6. SET SMALL = arr[j]
  7. SET pos = j
  8. [END OF if]
  9. [END OF LOOP]
  10. RETURN pos

**Code-**

1. **1A-**

**#include<stdio.h>**

**#include<math.h>**

**void n()**

**{**

**for (int i = 0; i <= 100; i++)**

**{**

**printf("%d, %d\n",i,i);**

**}**

**}**

**void n3()**

**{**

**double s;**

**for (double i = 0; i <= 100; i++)**

**{**

**s=pow(i,3.0);**

**printf("%f, %f\n",i,s);**

**}**

**}**

**void n2n()**

**{**

**double s;**

**for (double i = 0; i <= 100; i++)**

**{**

**s=i\*pow(2,i);**

**printf("%f, %f\n",i,s);**

**}**

**}**

**void e\_n()**

**{**

**double s;**

**for (double i = 0; i <= 100; i++)**

**{**

**s=exp(i);**

**printf("%f, %f\n",i,s);**

**}**

**}**

**void p\_2n()**

**{**

**double s;**

**for (double i = 0; i <= 100; i++)**

**{**

**s=pow(2,i);**

**printf("%f, %f\n",i,s);**

**}**

**}**

**void p\_32n()**

**{**

**double s;**

**for (double i = 0; i <= 100; i++)**

**{**

**s=pow(1.5,i);**

**printf("%f, %f\n",i,s);**

**}**

**}**

**void p\_2log()**

**{**

**double s;**

**for (double i = 0; i <= 100; i++)**

**{**

**s=log2(i);**

**s=pow(2,s);**

**printf("%f, %f\n",i,s);**

**}**

**}**

**void fact()**

**{**

**double s;**

**for (double i = 0; i <= 20; i++)**

**{**

**s=1;**

**for (double j = 1; j <= i; ++j)**

**{**

**s=s\*j;**

**}**

**printf(" %f\n",s);**

**}**

**}**

**void loglogn()**

**{**

**double s;**

**for (double i = 0; i <= 100; i++)**

**{**

**s=log2(i);**

**s=log2(s);**

**printf("%f, %f\n",i,s);**

**}**

**}**

**void log2n()**

**{**

**double s;**

**for (double i = 0; i <= 100; i++)**

**{**

**s=log2(i);**

**s=pow(s,2);**

**printf("%f, %f\n",i,s);**

**}**

**}**

**void log\_2n()**

**{**

**double s;**

**for (double i = 0; i <= 100; i++)**

**{**

**s=log2(i);**

**s=pow(s,0.5);**

**printf("%f, %f\n",i,s);**

**}**

**}**

**void main()**

**{**

**n();**

**n3();**

**n2n();**

**e\_n();**

**p\_2n();**

**p\_32n();**

**fact();**

**p\_2log();**

**loglogn();**

**log2n();**

**log\_2n();**

**}**

**Conclusion-** **The gradient of all Logarithmic functions decreases and gradient of all Exponential function increaes as n increaeses and other graphs are linear.**

1. **1B-**

**#include <stdio.h>**

**#include<stdlib.h>**

**#include<time.h>**

**void main()**

**{**

**int n=0;**

**for(int k=0; k<(100000/100); k++)**

**{**

**n=n+100;**

**int num[n];**

**int insert[n];**

**int select[n];**

**int j, min;**

**clock\_t start\_t, end\_t;**

**double total\_t;**

**printf("%d\t",n);**

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

**{**

**num[i]=rand() % 10;**

**insert[i]=num[i];**

**select[i]=num[i];**

**}**

**start\_t = clock();**

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

**{**

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

**j = i - 1;**

**while (j >= 0 && insert[j] > a)**

**{**

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

**j = j - 1;**

**}**

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

**}**

**end\_t = clock();**

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

**printf("%f\t", total\_t );**

**start\_t = clock();**

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

**{**

**min = i;**

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

**{**

**if (select[j] < select[min])**

**{**

**min = j;**

**}**

**}**

**if(min != i)**

**{**

**int temp=select[i];**

**select[i]=select[min];**

**select[min]=temp;**

**}**

**}**

**end\_t = clock();**

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

**printf("%f\n", total\_t );**

**}**

**}**

**Conclusion-**

**I have understood the Insertion Sort and Selection sort algorithm and their time complexities. I also understood how to calculate them and draw similar inferences.**