Suraj Iyer

2021300045

SE Comps A

Batch C

DAA EXPERIMENT 1

Aim – To implement the various functions e.g. linear, non-linear, quadratic, exponential etc.

Details – A function is a relation between a set of inputs and a set of permissible outputs with the property that each

input is related to exactly one output. Let A & B be any two non-empty sets; mapping from A to B will be a function

only when every element in set A has one end, only one image in set B.

Code -

| #include <stdio.h>  #include <math.h>  // (3/2)^n  float one (int n) {  return pow(1.5,n);  }  // ln(ln(n))  float two (int n) {  return log(log(n));  }  // 2^(log2(n))  float three (int n) {  return pow(2,log2(n));  }  // log2(log2(n))  float four (int n) {  return log2(log2(n));  }  // n^3  float five (int n) {  return pow(n,3);  }  // log2(n)  float six (int n) {  return log2(n);  }  // log2(n)^log2(n)  float seven (int n) {  return pow(log2(n),log2(n));  }  // 2^2log2(n)^1/2  float eight (int n) {  return pow(2,pow(2\*log2(n),1/2));  }  // log2(n)^2  float nine (int n) {  return pow(log2(n),2);  }  // n\*2^n  float ten (int n) {  return n\*pow(2,n);  }  // n!  float fact (int n) {  int ans = 1;  for (int i = 2; i<n+1; ++i) ans\*=i;  return ans;  }  int main () {  printf ("Sr No.\t(3/2)^n\tln(ln(n))\t2^(log2(n))\tlog2(log2(n))\tn^3\tlog2(n)\tlog2(n)^log2(n)\t2^2log2(n)^1/2\t2^2log2(n)^1/2\tn\*2^n\tn!\n");  int cnt = 0;  for (int i = 0; i<101; ++i) {  printf ("%d\t", i);  printf ("%.3f\t", one(i));  printf ("%.3f\t", two(i));  printf ("%.3f\t", three(i));  printf ("%.3f\t", four(i));  printf ("%.3f\t", five(i));  printf ("%.3f\t", six(i));  printf ("%.3f\t", seven(i));  printf ("%.3f\t", eight(i));  printf ("%.3f\t", nine(i));  printf ("%.3f\t", ten(i));  if (cnt++<21) printf ("%.3f\t", fact(i));  printf("\n");  }  return 0;  } |
| --- |

Output -

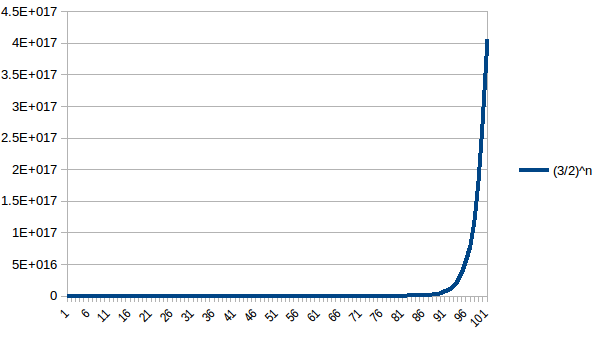
|  |
| --- |

Table -

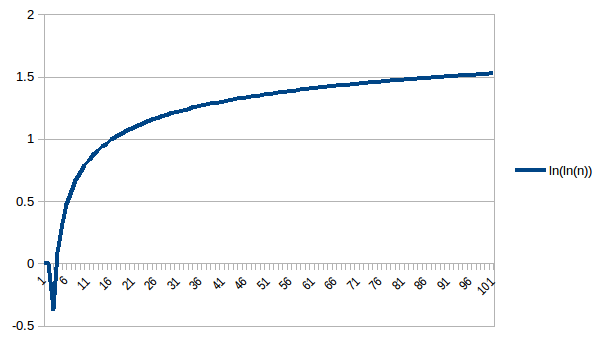
|  |
| --- |

Graphs -

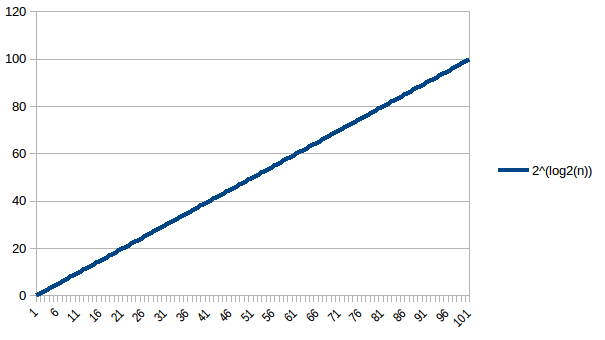
(1) (3/2)^n



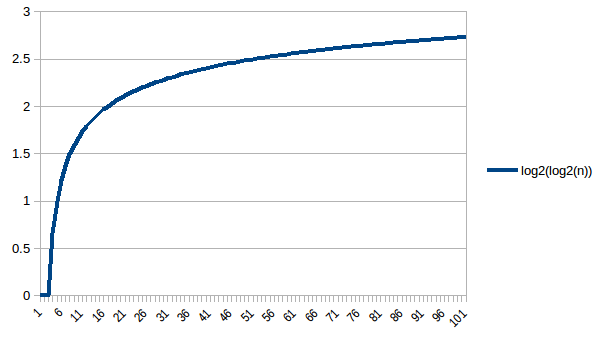
(2) ln(ln(n))



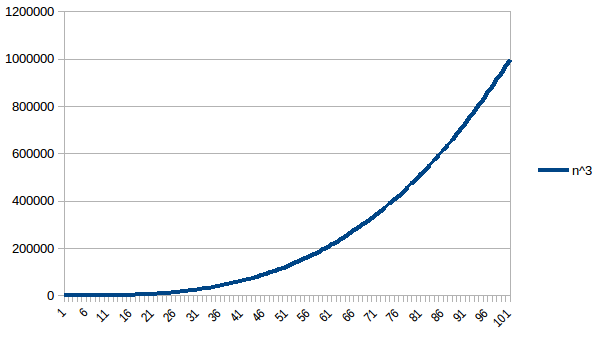
(3) 2^(log2(n))



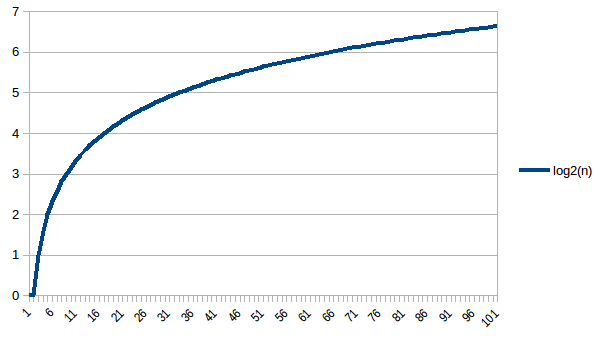
(4) log2(log2(n))



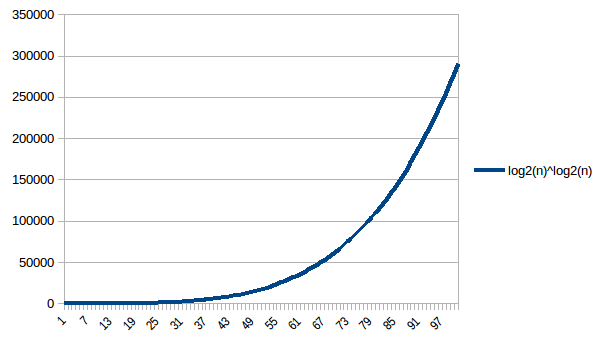
(5) n^3



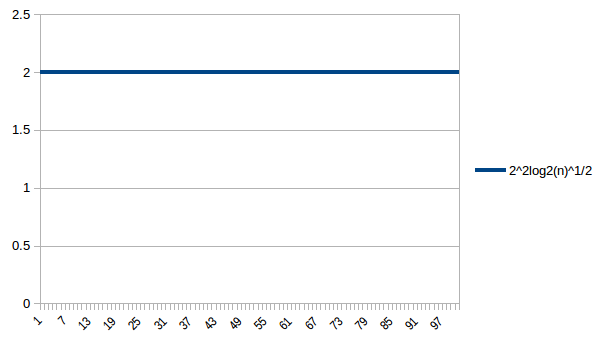
(6) log2(n)



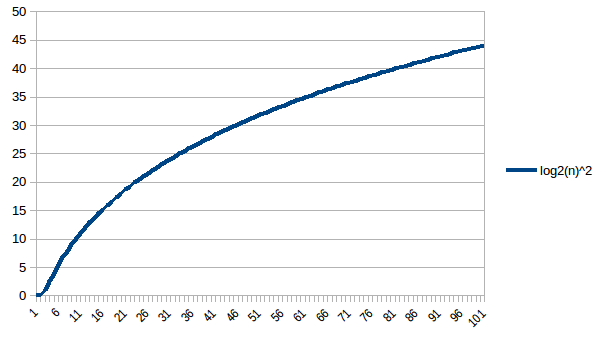
(7) log2(n)^log2(n)



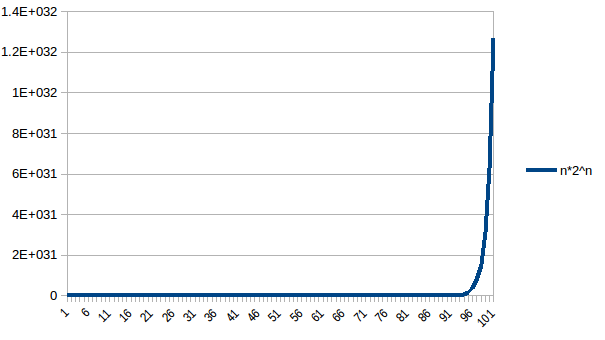
(8) 2^2log2(n)^½



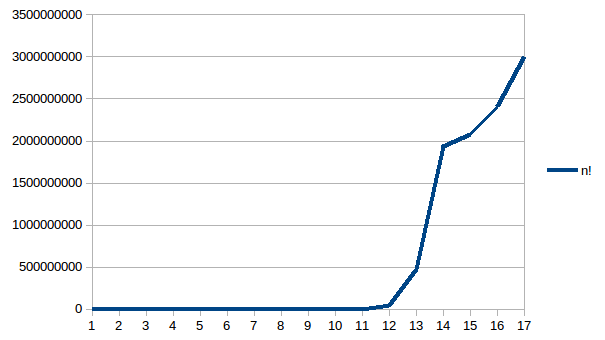
(9) log2(n)^2



(10) n\*2^n



(11) n!



Conclusion -

I have successfully computed the calculations of more than 10 functions, elaborated them for over a hundred values, and visualized them in the form of graphs. What I have learned from this is that some numerical values cannot be stored in programming variables, it is a phenomenon known as buffer overloading. The graphical representation of several functions also helped me visualize the working of several functions and helped me understand how they work conceptually behind the scenes.