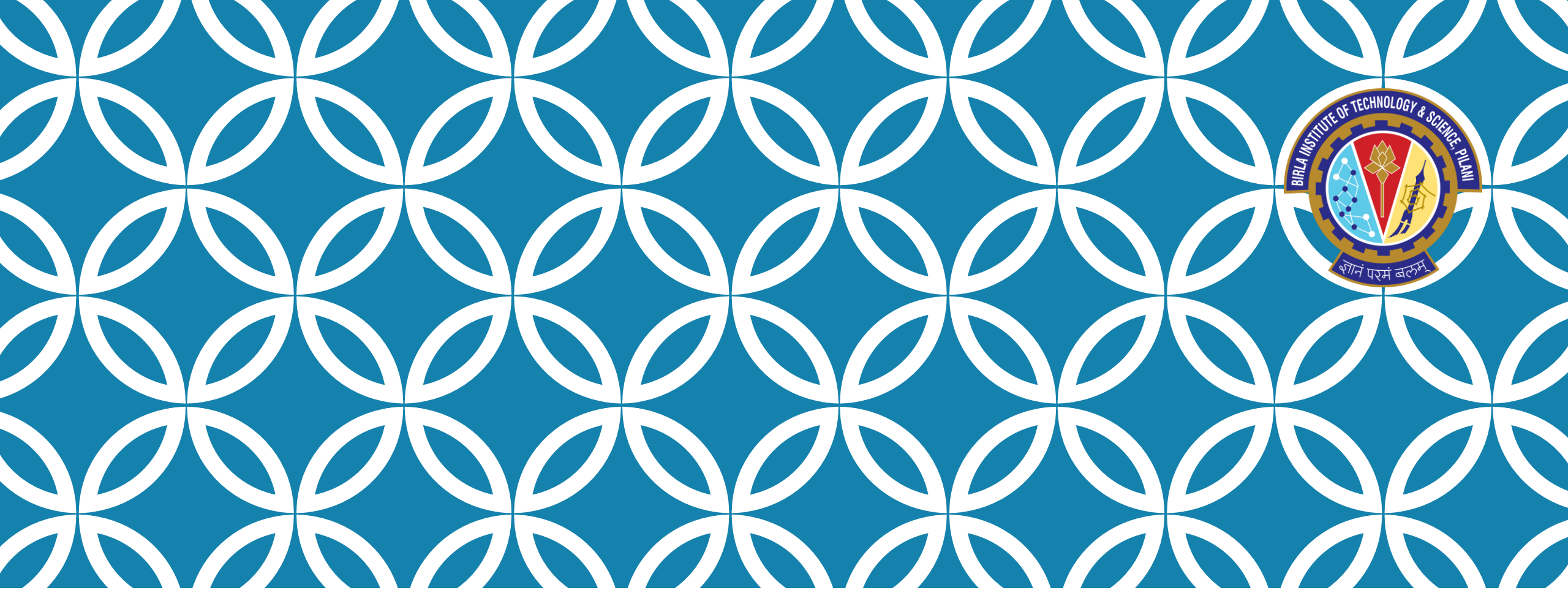


BITS F232: FOUNDATIONS OF DATA STRUCTURES & ALGORITHMS (1ST SEMESTER 2023-24) RECURSION CONTINUED…

Chittaranjan Hota, PhD
Sr. Professor of Computer Sc.
BITS-Pilani Hyderabad Campus
hota[AT]hyderabad.bits-pilani.ac.in

LINEAR RECURSION

- A linear recursive function is a function that makes at most one recursive call each time it is invoked (as opposed to one that would call itself multiple times during its execution).

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

int linearSum(int A[],int n){
    if(n == 1) return A[0];
    return linearSum(A,n-1) + A[n-1];
}

int main(){
    int len;
    cout<<"Enter length of input array : ";
    cin>>len;
    cout<<endl;
    int A[len];
    int aux;
    cout<<"Enter the array : ";
    for(int i=0;i<len;i++){
        cin>>aux;
        A[i] = aux;
    }
    cout<<"\nSum of the array is : "<<linearSum(A,len)<<endl;
    return 0;
}
```

Result

compiled and executed in 13.95 sec(s)

```
Enter length of input array : 5

Enter the array : 4 3 6 2 5

Sum of the array is : 20
```

TAIL RECURSION: REVERSING AN ARRAY

```cpp
#include <iostream>
using namespace std;
void swap(int *a,int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}
void reverseArray(int A[],int start,int end){
    if(start < end){
        swap(&A[start],&A[end]);
        reverseArray(A,start+1,end-1);
    }
}
int main(){
    int len;
    cout<<"Enter length of the input array : ";
    cin>>len;
    cout<<endl;
    int A[len];
    cout<<"Enter the array : ";
    int aux;
    for(int i=0;i<len;i++){
        cin>>aux;
        A[i] = aux;
    }
    reverseArray(A,0,len-1);
    cout<<"\nReversed array : ";
    for(int i=0;i<len;i++) cout<<A[i]<<" ";
    cout<<endl;
    return 0;
}
```

- Tail recursion occurs when a linearly recursive method makes its recursive call as its last step.
- Such methods can be easily converted to non-recursive methods (which saves on some resources).
- Is Linear sum (previous program, tail recursive?)

Algorithm IterativeReverseArray(A, i, j):
Input: An array A and nonnegative integer indices i and j
Output: The reversal of the elements in A starting at index i and ending at j

while $i < j$ **do**
  Swap $A[i]$ and $A[j]$
  $i \leftarrow i + 1$
  $j \leftarrow j - 1$
return

Result
compiled and executed in 12.089 sec(s)

```
Enter length of the input array : 4

Enter the array : 4 5 7 8

Reversed array : 8 7 5 4
```

BINARY RECURSION

- What is binary recursion?

Problem: add all the numbers in an integer array A:

Algorithm BinarySum(*A*, *i*, *n*):
  Input: An array *A* and integers *i* and *n*
  Output: The sum of the *n* integers in *A* starting at index *i*
  if $n = 1$ **then**
    return *A*[*i*];
  return BinarySum(*A*, *i*, $n/2$) + BinarySum(*A*, $i + n/2$, $n/2$)

Let us see the recursion trace…

Used heavily in merging and tree traversals…

COMPUTING FIBONACCI NUMBERS

```cpp
#include<bits/stdc++.h>
using namespace std;

int fib(int n)
{
    if (n <= 1)
        return n;
    return fib(n-1) + fib(n-2);
}

int main ()
{
    int n = 9;
    cout << fib(n);
    getchar();
    return 0;
}
```

Is binary recursion better here?

n_k denote the number of calls performed in the execution of fib(k)

$$
\begin{aligned}
n_0 &= 1 \\
n_1 &= 1 \\
n_2 &= n_1 + n_0 + 1 = 1 + 1 + 1 = 3 \\
n_3 &= n_2 + n_1 + 1 = 3 + 1 + 1 = 5 \\
n_4 &= n_3 + n_2 + 1 = 5 + 3 + 1 = 9 \\
n_5 &= n_4 + n_3 + 1 = 9 + 5 + 1 = 15 \\
n_6 &= n_5 + n_4 + 1 = 15 + 9 + 1 = 25 \\
n_7 &= n_6 + n_5 + 1 = 25 + 15 + 1 = 41 \\
n_8 &= n_7 + n_6 + 1 = 41 + 25 + 1 = 67
\end{aligned}
$$

Let us draw the tree too…

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

// A tail recursive function to
// calculate n th fibnacci number
int fib(int n, int a = 0, int b = 1)
{
    if (n == 0)
        return a;
    if (n == 1)
        return b;
    return fib(n - 1, b, a + b);
}

// Driver Code
int main()
{
    int n = 9;
    cout << "fib(" << n << ") = "
         << fib(n) << endl;
    return 0;
}
```

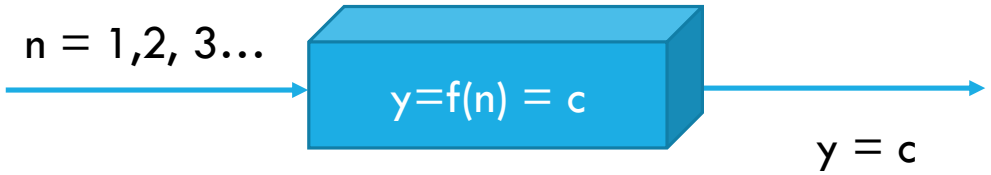
What is the type of this rec.?

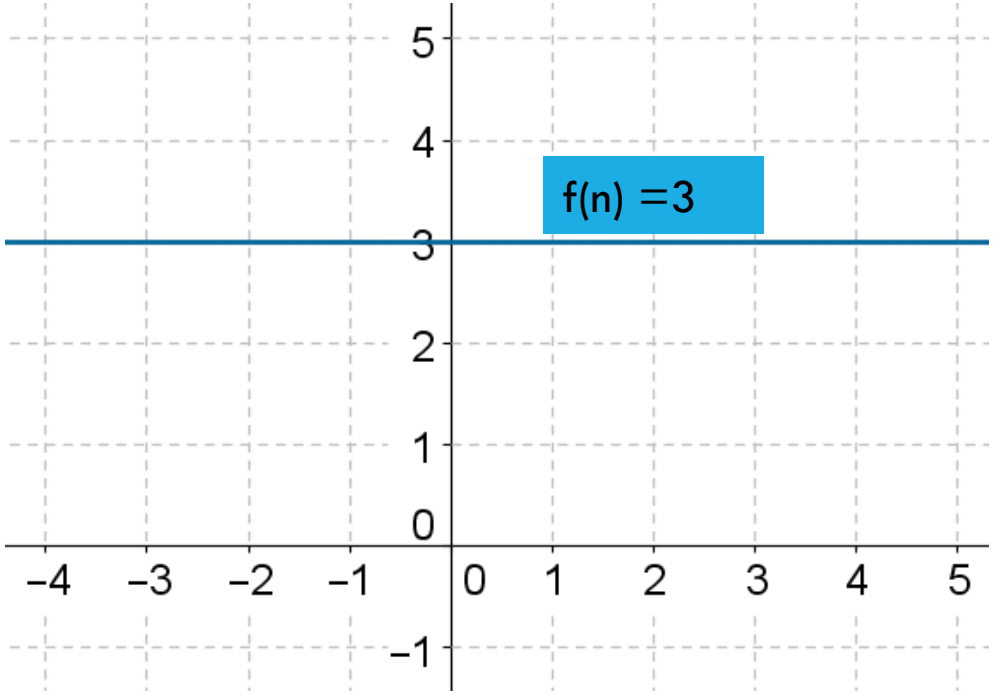
WHAT IS COMPLEXITY & HOW IMPORTANT IS IT?

You want to look for a word in a dictionary that has every word sorted alphabetically. How many algorithms are there and which one will you prefer?

FUNCTIONS FOR ALGORITHM ANALYSIS

- The Constant function: Output is same i.e. independent of input. It characterizes the number of steps needed to do a basic operation on a computer like, what types of operations? Constant algorithm does not depend on the input size.

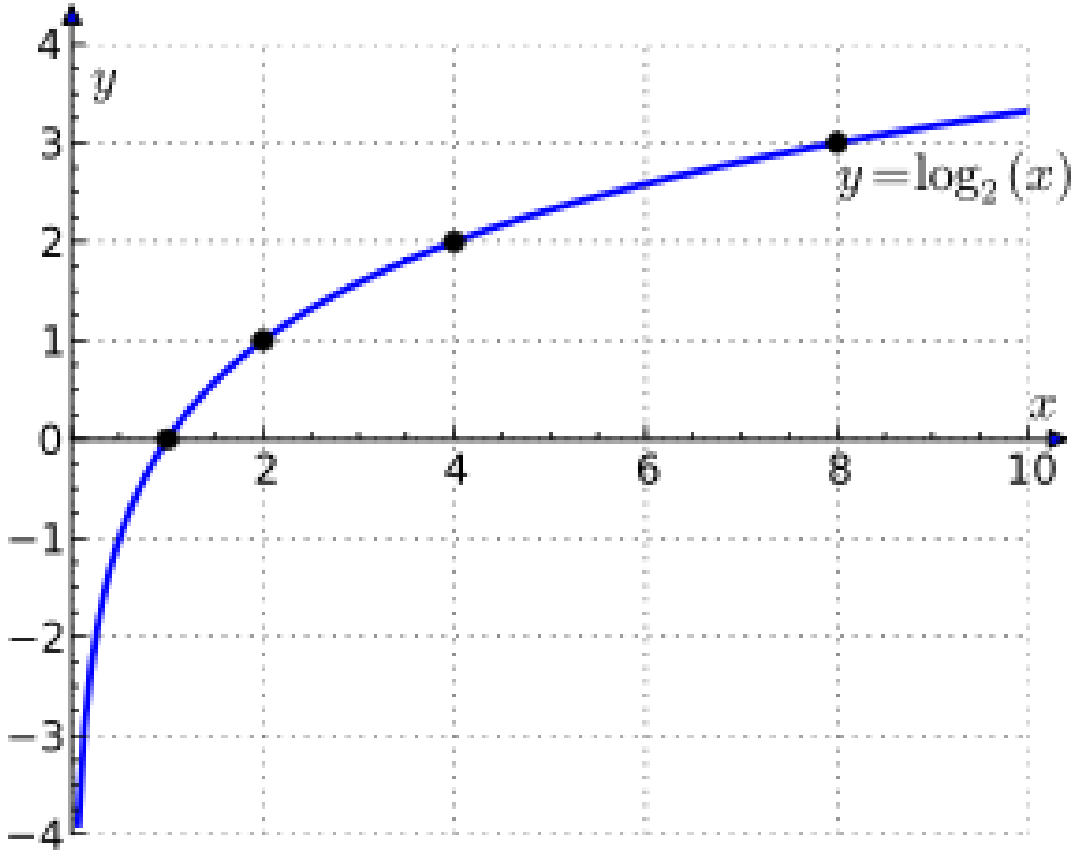




What about $f(w) = w^2$?

LOGARITHMIC FUNCTION

- Heavily used in Analysis of algorithms.
- Why is it used with base 2 most?



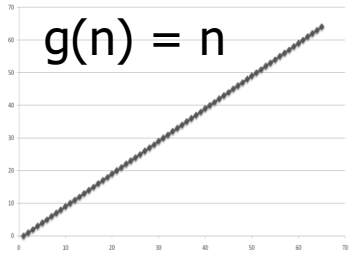
(Rules)

1. log(nm) = log n + log m
2. log (n/m) = log n – log m
3. Log (n^r) = r log n
4. Log_a n = $\log_b$ n/$\log_b$ a

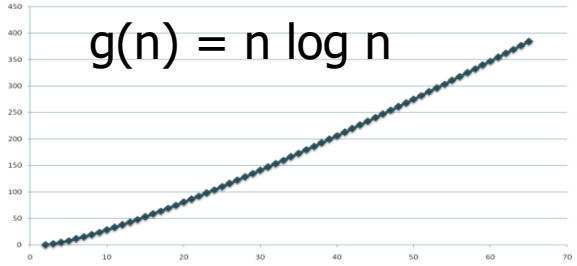
(Img. source: wiki)

LINEAR AND N-LOG-N FUNCTIONS

Linear Function: Given an input value ‘n’, the linear function g (n) assigns the value ‘n’ itself.



nlogn function: A function that assigns to an input ‘n’ the value of ‘n’ times logarithm base 2 of ‘n’.




```
void quicksort (list[ ], left, right) {
   int pivot = partition (list, left, right);
   quicksort (list, left, pivot-1);
   quicksort (list, pivot+1, right);
}
```

complexity?