upGrad

Full-Stack Software Development

upGrad

Lecture On: Analysis of Algorithms and Recursion

Instructor: Arkoprovo Dey

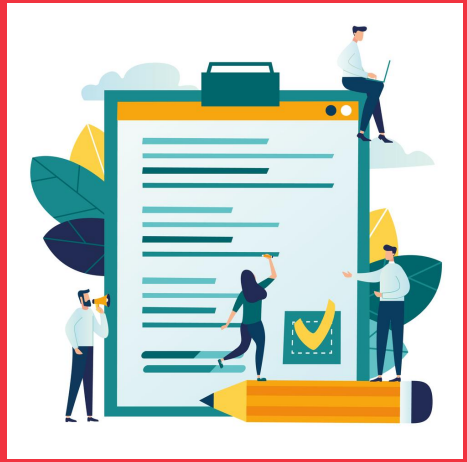
Course RoadMap

- Analysis of Algorithms
- Mathematics
- Bit Manipulation
- Recursion
- Arrays
- Searching Algorithms
- Sorting Algorithms
- Strings
- Linked List
- Stacks
- Queues
- Trees

Today's Agenda

1 Introduction to Data Structures and Algorithms

2 Asymptotic Analysis of Algorithm



Why Should We Study Data Structures and Algorithms?

We have all come across several quotes along the lines of *“Work Smarter, Not Harder”*. The same principle applies to software development, maybe even more than anywhere else.

Any program that we write must be able to scale to a large extent seamlessly. This is where good knowledge of Data Structures and Algorithms comes in handy. It would help us devise efficient solutions to our problems.

We also need to define what the word **‘efficient’** means to us.

Why Should We Study Data Structures and Algorithms?

Every single thing that we do or see around us can be thought of as a program that can and should be optimised. Let's take the very simple example of building a house. Simplistically, what is the input in this case? Well, they are the 'bricks' and 'cement'. And what would be the output? The 'house'.

But how would we go about with building the house? Do we just randomly lay bricks? Or do we work on laying the foundation first? Do we want to construct one wall at a time or raise all the walls together bit by bit?

All of these decisions will impact the strength of the house being built. The process that we choose to follow, in this case, is known as the '**Algorithm**'.

Why Should We Study Data Structures and Algorithms?

Next, the inputs - ‘bricks’ and ‘cement’ that we will use are, of course, inanimate objects that cannot move on their own from the point of manufacture to the building under construction. We will have to figure out how to transport the raw material and then move them.

Now, a truck for transporting the raw materials (bricks & cement) makes sense; manually carrying them does not! These temporary structures that we use to transport the raw material are analogous to our ‘Data Structure’, where the raw material is the ‘**Data**’.

In order to make the entire program efficient, both the Data Structure and the Algorithm will have to be efficient.

Why Should We Study Data Structures and Algorithms?

Can you guess what would be the Data, the Data Structures and the Algorithms to use in the following examples?

- Baking a cake
- An Amazon delivery associate delivering 10 packages
- Looking for a word in a dictionary

In the world of software development, we are faced with a large number of problems where the choice of Data Structure and Algorithm is of very high importance.

How would a Google search work?

How does Netflix figure out what TV shows it should recommend for us?

Why Should We Study Data Structures and Algorithms?

We have certain common paradigms in Data Structures and Algorithms that form the foundation for a lot of specific algorithms:

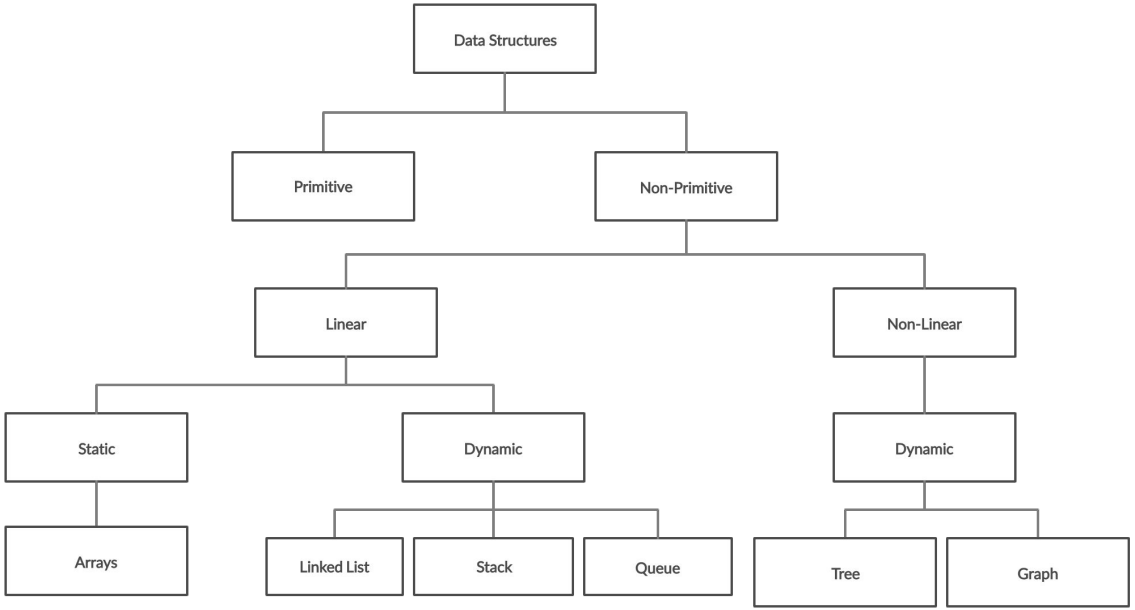
- **Recursion,** method of solving a problem where the solution depends on solutions to smaller instances of the same problem. Such problems can be solved easily using recursion.
- **Divide and Conquer**, where we try and 'divide' the data into smaller chunks, which are easier to 'conquer'. For example, Merge Sort, Quick Sort, etc.

We shall look into these paradigms later in detail when we study specific algorithms.

Why Should We Study Data Structures and Algorithms?

Data structures can be treated as complex data types. We have earlier looked at the primitive datatypes in Java (boolean, char, short etc).

One way of studying data structures is by exploring them as non-primitive data types.



Poll 1 (15 sec.)

Data Structure intends to achieve

1. Optimal storage of data
2. Optimal query of data
3. Both optimal storage and optimal query
4. Neither optimal storage nor optimal query

Poll 1 (Answer)

Data Structure intends to achieve

1. Optimal storage of data
2. Optimal query of data
3. **Both optimal storage and optimal query**
4. Neither optimal storage nor optimal query

Poll 2 (30 sec.)

Which of the following is true about an algorithm?

1. It’s a step-by-step process to solve a problem
2. Out of all the options available, only the most efficient way of solving a problem is known as an algorithm
3. Algorithms help in optimal storage of data
4. None of these

Poll 2 (Answer)

Which of the following is true about an algorithm?

1. **It's a step-by-step process to solve a problem**
2. Out of all the options available, only the most efficient way of solving a problem is known as an algorithm
3. Algorithms help in optimal storage of data
4. None of these

Time Complexity

The ‘running time’ of our algorithm is the most important metric for us, and we usually go to any lengths to make sure our program's ‘running time’ is kept in check. To get an idea of the ‘running time', we use a technique known as **Asymptotic Analysis**.

We will learn about the **Big-O notation** which represents the worst case running time of an algorithm.

The Big-O notation

The Big - O notation represents a function that is like an **upper bound** to the actual running time of the program. It represents the **worst-case** performance of the program.

Mathematically, given two non-negative functions f(n) and g(n), there exists an integer n_o and a constant c > 0, such that for all integers $n \geq n_o$, if $|f(n)| \leq c. |g(n)|$, then f(n) is O(g(n)).

This means that if the running time of the program grows with the function g(n), then we can say that the worst-case time complexity of the program is 'O(f(n))'. In case the function f(n) has multiple terms, then we consider only the **leading** term. For example, if the function is ' $5x^2 + 4x$', then we consider the time complexity to be $O(x^2)$.

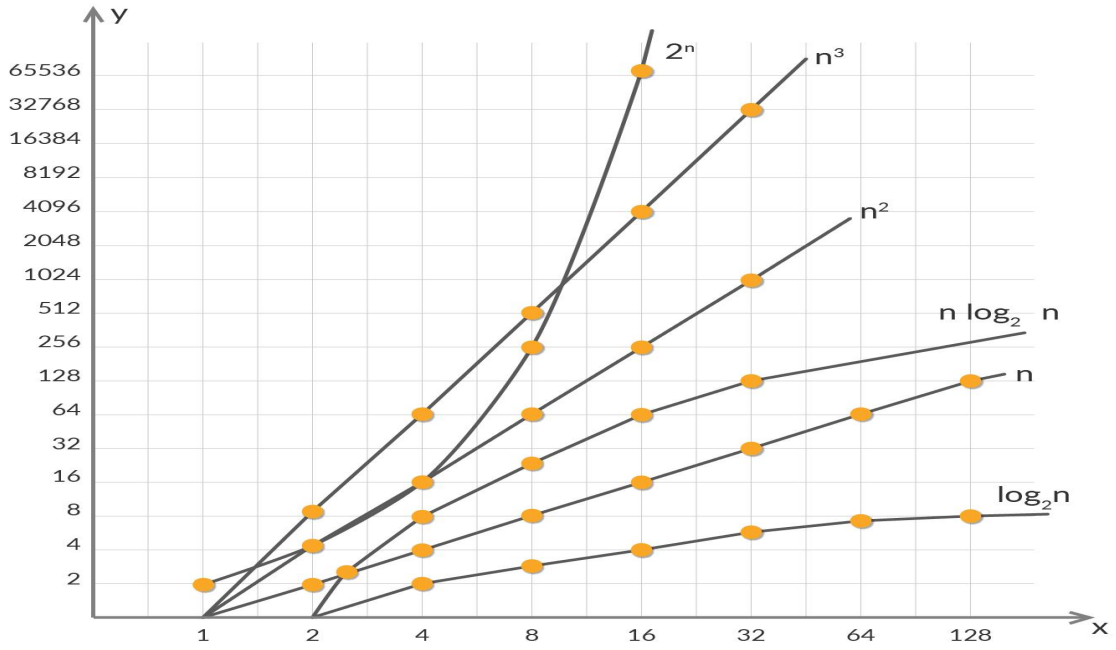
upGrad

Comparing a Few Common Time Complexities

Efficiency:

$\log_2 n > n > n*\log_2 n > n^2 > n^3 > 2^n$

2^n being the worst.



What is the time complexity?

upGrad

```
void func(int n){
 for(int i=1 ; i*i<n ; i++){
   System.out.println("upGrad");
 }
}
```

O(sqrt(n))

```
void func(int n){
  for(int i=1 ; i<=n ; i++){
     for(int j=1 ; j<=n ; j++){
        System.out.println("Find my TC");
        break;
    }
  }
}
```

O(n)

What is the time complexity?

upGrad

```
void func(int n){
 for(int i=1 ; i<n ; i=i*2){
   System.out.println("upGrad");
 }
}
```

O(logn)

```
void func(int n){
 for(int i=n ; i>0 ; i=i/2){
  System.out.println("upGrad");
 }
}
```

O(logn)

What is the time complexity?

upGrad

O(1)

```
void func(int n){
if(n==1){
   for(int i=1 ; i<=n ; i++){
     for(int j=1 ; j<=i ; j++){
     System.out.println("Find my TC");
     break;
     }
   }
  }
}
```

What is the time complexity?

upGrad

```
int a = 0;
for(i = 0; i < n*n; i++)
    for(j = 0; j < 100000; j++)
      a++;
```

$O(n^2)$

```
int m = 0, n = 0;
for (i = 0; i < a; i++)
  m++;
for (j = 0; j < b; j++)
  n++;
```

O(a+b)

What is the time complexity?

upGrad

O((n^3) * (logn))

```
for(double i = n; i >= 0; i /= 2) {
  for(double j = 1; j <= n; j += 2) {
    System.out.println("Hello World!");
  }
  for(double j = 1; j <= n*n; j += 1) {
    System.out.println("Hello World Again!");
  }
  for(double j = 1; j <= n*n*n; j += 2) {
    System.out.println("Hello World Again!");
  }
}
```

Poll 3 (30 sec.)

Which symbol is used to represent the worst-case time complexity of an algorithm?

1. **Θ**
2. **Ω**
3. **O**
4. None of these

Poll 3 (Answer)

Which symbol is used to represent the worst-case time complexity of an algorithm?

1. **Θ**
2. **Ω**
3. **O**
4. None of these

Poll 4 (20 sec.)

Which of the following is true about time complexity?

1. It’s independent of the input size
2. It’s independent of the programming language
3. It’s independent of the operating system
4. Both 2 and 3

Poll 4 (Answer)

Which of the following is true about time complexity?

1. It’s independent of the input size
2. It’s independent of the programming language
3. It’s independent of the operating system
4. **Both 2 and 3**

Poll 5 (30 sec.)

Which of the following is true about time complexity?

1. Time complexity is dependent on how much memory is occupied
2. Time complexity quantifies the variable number of operations with respect to input size
3. Time complexity is dependent on the programming language that is used to write the algorithm
4. None of these

Poll 5 (Answer)

Which of the following is true about time complexity?

1. Time complexity is dependent on how much memory is occupied
2. **Time complexity quantifies the variable number of operations with respect to input size**
3. Time complexity is dependent on the programming language that is used to write the algorithm
4. None of these

Poll 6 (30 sec.)

What is the meaning of O(n)?

1. In the worst case, the number of operations will be equal to n
2. In the average case, the number of operations will be equal to n
3. In the best case, the number of operations will be equal to n
4. None of these

Poll 6 (Answer)

What is the meaning of O(n)?

1. In the worst case, the number of operations will be equal to n
2. In the average case, the number of operations will be equal to n
3. In the best case, the number of operations will be equal to n
4. **None of these**

Space Complexity

Whenever a program is written, some memory is required to execute it. For any algorithm, memory may be used for the following:

- Variables (including constant values, temporary values)
- Program Instructions
- Execution

Space complexity is the amount of memory used by the algorithm to execute the program and produce the result.

What is the space complexity?

upGrad

```
public int[] reverse(int[] arr) {
 int i=0;
 int j=arr.length-1;
 while(i<j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
    i++;
    j--;
}
```

Poll 7 (15 sec.)

What is space complexity?

1. The amount of time taken to solve a problem
2. The amount of storage needed to solve a problem
3. Both 1 and 2
4. Neither 1 nor 2

Poll 7 (Answer)

What is space complexity?

1. The amount of time taken to solve a problem
2. **The amount of storage needed to solve a problem**
3. Both 1 and 2
4. Neither 1 nor 2

Poll 8 (20 sec.)

Space complexity depends on

1. The language used to write the algorithm
2. The operating system used for executing the code
3. Both 1 and 2
4. Neither 1 nor 2

Poll 8 (Answer)

Space complexity depends on

1. The language used to write the algorithm
2. The operating system used for executing the code
3. Both 1 and 2
4. **Neither 1 nor 2**

Hands-On Exercise 1

upGrad

What is the time and space complexity of the below code snippet?

```
float x = 0;
for(a = 0; a < n; a++)
    for(b = 0; b < n; b++)
        for(c = 0; c < 100; c++)
            x++;
```

Hands-On Exercise 2

upGrad

What is the time and space complexity of the below code snippet?

```
public void function(int n) {
      int j = 1;
      while (j < n){
          j = j*2;
     }
}
```

Homework

1. Explain what is the Big-O Notation?
2. Explain the difference between time and space complexity.

Tasks to complete after the session

Homework Questions
MCQs

In the next class...

- Bit Manipulation

upGrad

#RahoAmbitious

Thank You!