CS303 - Algorithms and Data Structures

Lecture 4

- Analysis of Algorithms-

Professor : Mahmut Unan – UAB CS

Agenda

- Time Complexity
- Heap Sort

The RAM Model of Computation

- Random Access Machine
  - Each simple operation (+ - * / + if …) takes exactly one time step
  - Loops are not simple operations
  - Each memory access takes one step
- We will calculate best, worse and average case complexities

If-Then-Else

```
if (cond) then
   block 1 (sequence of statements)
else
   block 2 (sequence of statements)
end if;
```

Here, either block 1 will execute, or block 2 will execute. Therefore, the worst-case time is the slower of the two possibilities:

$$\max(\text{time(block 1)}, \text{time(block 2)})$$

If block 1 takes O(1) and block 2 takes O(N), the if-then-else statement would be O(N).

http://web.mit.edu/16.070/www/lecture/big_o.pdf

Statements with function/ procedure calls

When a statement involves a function/ procedure call, the complexity of the statement includes the complexity of the function/ procedure. Assume that you know that function/ procedure *f* takes constant time, and that function/procedure *g* takes time proportional to (linear in) the value of its parameter *k*. Then the statements below have the time complexities indicated.

f(k) has O(1)
g(k) has O(k)

When a loop is involved, the same rule applies. For example:

```
for J in 1 .. N loop
    g(J);
end loop;
```

Order of Growth

- It is how the time of execution depends on the length of the input.

Big – O Notation

- Big O notation is used in Computer Science to describe the performance or complexity of an algorithm.
- Big O specifically describes the worst-case scenario, and can be used to describe the execution time required

O-notation:

To denote asymptotic upper bound, we use O-notation. For a given function $g(n)$, we denote by $O(g(n))$ (pronounced "big-oh of g of n") the set of functions:

$O(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that $0 \leq f(n) \leq c * g(n)$ for all $n \geq n_0 \}$

Ω-notation:

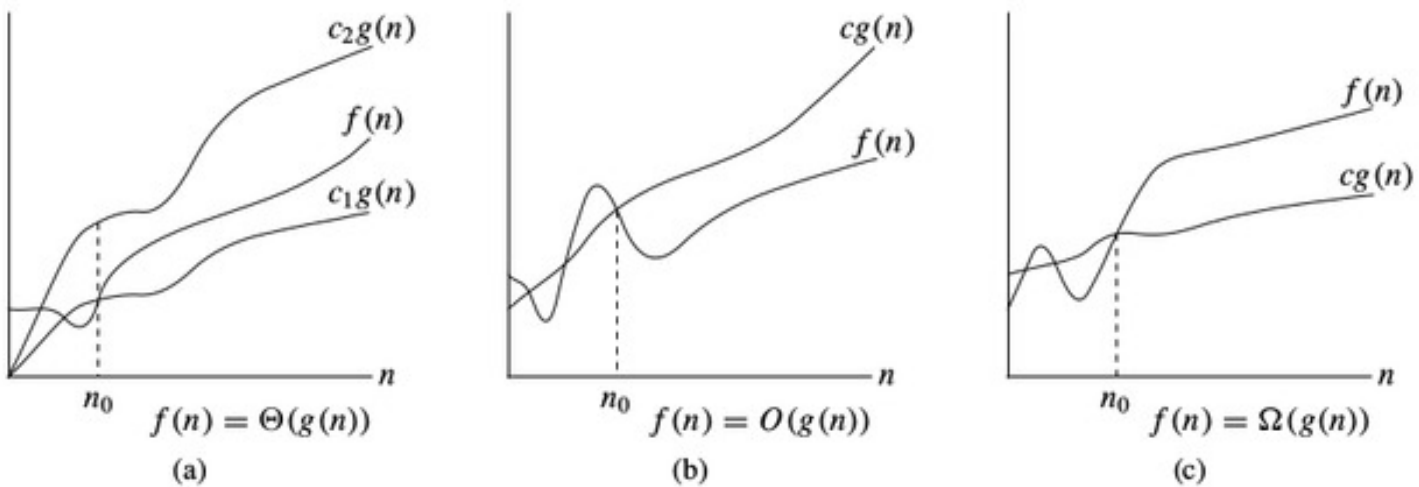
To denote asymptotic lower bound, we use Ω-notation. For a given function $g(n)$, we denote by $\Omega(g(n))$ (pronounced "big-omega of g of n") the set of functions:

$\Omega(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that $0 \leq c * g(n) \leq f(n)$ for all $n \geq n_0 \}$

Θ-notation:

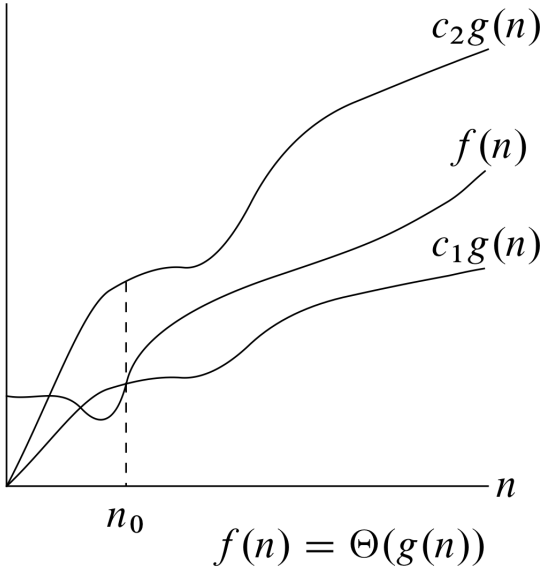
To denote asymptotic tight bound, we use Θ-notation. For a given function $g(n)$, we denote by $\Theta(g(n))$ (pronounced "big-theta of g of n") the set of functions:

$\Theta(g(n)) = \{ f(n)$: there exist positive constants c_1, c_2 and n_0 such that $0 \leq c_1 * g(n) \leq f(n) \leq c_2 * g(n)$ for all $n > n_0 \}$

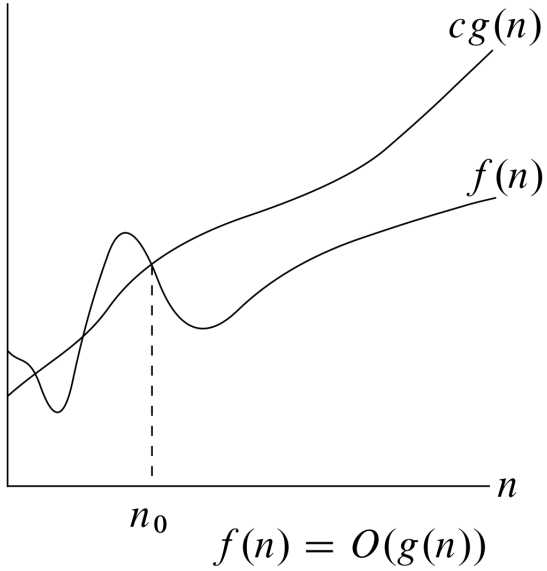


https://www.hackerearth.com/practice/basic-programming/complexity-analysis/time-and-space-complexity/tutorial/

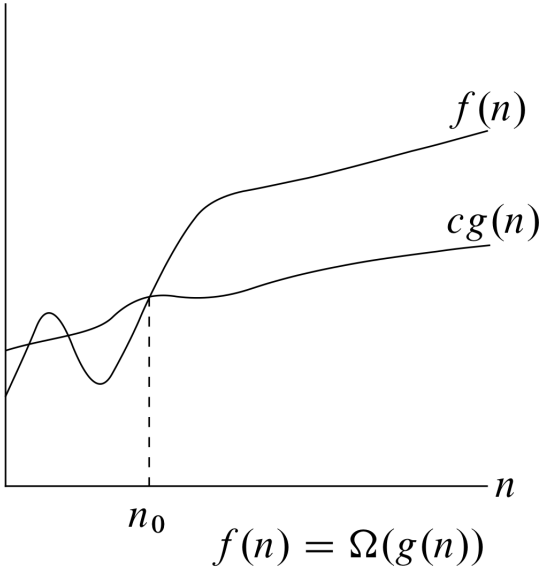
Asymptotic Analysis



(a)



(b)



(c)

$\Theta(g(n)) = \{f(n)$: there exist positive constants c_1, c_2, and n_0 such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0\}$.

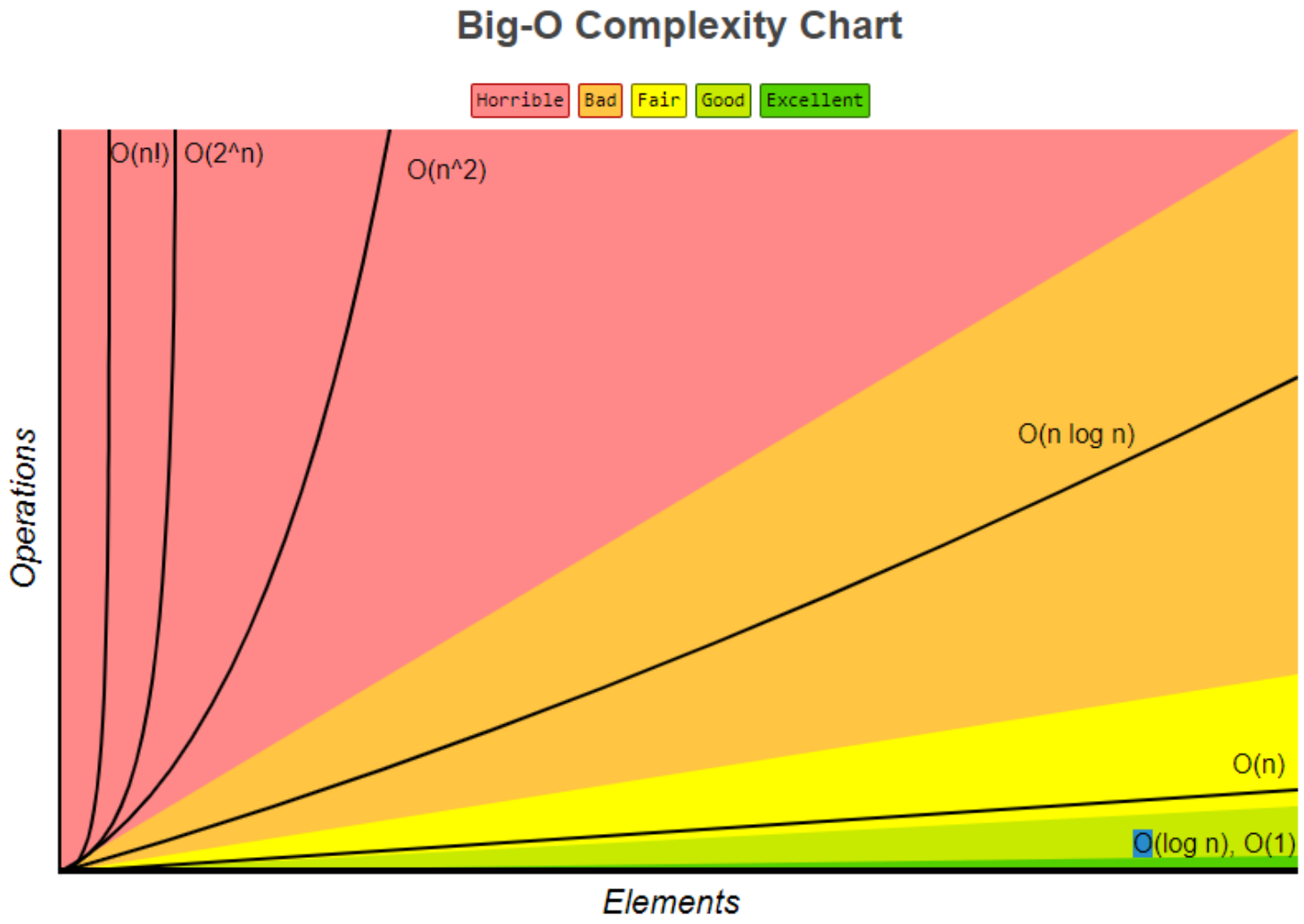
$O(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0\}$.

$\Omega(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 \le cg(n) \le f(n)$ for all $n \ge n_0\}$.

Working with the Big –O notation

- $O(f(n)) + O(g(n)) = O(\max(f(n), g(n)))$
- $O(c*f(n)) = O(f(n))$
- $O(f(n)) * O(g(n)) = O(f(n) * g(n))$
- ….

Big 0 Complexity Chart



Ref: http://bigocheatsheet.com/

Examples

1. What is the time, space complexity of following code:

```
int a = 0, b = 0;
for (i = 0; i < N; i++) {
    a = a + rand();
}
for (j = 0; j < M; j++) {
    b = b + rand();
}
```

https://www.geeksforgeeks.org/practice-questions-time-complexity-analysis/

1. What is the time, space complexity of following code:

```
int a = 0, b = 0;
for (i = 0; i < N; i++) {
    a = a + rand();
}
for (j = 0; j < M; j++) {
    b = b + rand();
}
```

O(N + M) ➔ O(n) time, O(1) space

2. What is the time complexity of following code:

```
int a = 0;
for (i = 0; i < N; i++) {
    for (j = N; j > i; j--) {
        a = a + i + j;
    }
}
```

1. O(N)
2. O(N*log(N))
3. O(N * Sqrt(N))
4. O(N*N)

2. What is the time complexity of following code:

```
int a = 0;
for (i = 0; i < N; i++) {
    for (j = N; j > i; j--) {
        a = a + i + j;
    }
}
```

1. O(N)
2. O(N*log(N))
3. O(N * Sqrt(N))
4. O(N*N)

O(n^2)

3. What is the time complexity of following code:

```
int i, j, k = 0;
for (i = n / 2; i <= n; i++) {
    for (j = 2; j <= n; j = j * 2) {
        k = k + n / 2;
    }
}
```

1. O(n)
2. O(nLogn)
3. O(n^2)
4. O(n^2Logn)

3. What is the time complexity of following code:

```
int i, j, k = 0;
for (i = n / 2; i <= n; i++) {
    for (j = 2; j <= n; j = j * 2) {
        k = k + n / 2;
    }
}
```

1. O(n)
2. O(nLogn)
3. O(n^2)
4. O(n^2Logn)

O(n logn)

```
int a = 0, i = N;
while (i > 0) {
    a += i;
    i /= 2;
}
```

1. O(N)
2. O(Sqrt(N))
3. O(N / 2)
4. O(log N)

```
int a = 0, i = N;
while (i > 0) {
        a += i;
        i /= 2;
}
```

O(logn)

1. O(N)
2. O(Sqrt(N))
3. O(N / 2)
4. O(log N)

What are Divide and Conquer algorithms? Describe how they work. Can you give any common examples of the types of problems where this approach might be used?

What are the key advantages of Insertion Sort, Quicksort, Heapsort and Mergesort? Discuss best, average, and worst case time and memory complexity.

https://www.interviewbit.com/courses/programming/topics/time-complexity/

1) Performance of traversing given integer array with the help of for loop ..

Code:

```
int a[] = {23,45,66,67,78,90};

for(int i=0;i<a.length;i++ {
    System.out.println("  value :"+a[i]);
}
```


http://javainterviewquestionswithanswer.blogspot.com/2016/11/calculate-time-complexity-of-below-code.html

```
function getMinMaxRange( array ) {
    let max = -Infinity;
    let min = Infinity;

    for ( let value of array ) {
        if ( value > max ) max = value;
    }
    for ( let value of array ) {
        if ( value < min ) min = value;
    }

    return max - min;
}
```

http://www.zsoltnagy.eu/big-o-notation-for-beginner-tech-interview-candidates/

```
for ( let i = 0; i < array.length; ++i ) {
    for ( let j = i; j < array.length; ++j ) {
        for ( let k = j; k < array.length; ++k ) {
            for ( let l = k; l < array.length; ++l ) {
                // O(N**4) algorithm
            }
        }
    }
}
```

http://www.zsoltnagy.eu/big-o-notation-for-beginner-tech-interview-candidates/

Next Class

- Heap Sort
- Heap Sort Time complexity