Nile University

Data Structures and Algorithms

Episode Seven (Complexity Analysis)

Agenda for Today

- What is Asymptotic Notations?
- Code Analysis
- Big-O
- Omega
- Theta

Remember before Mid-term

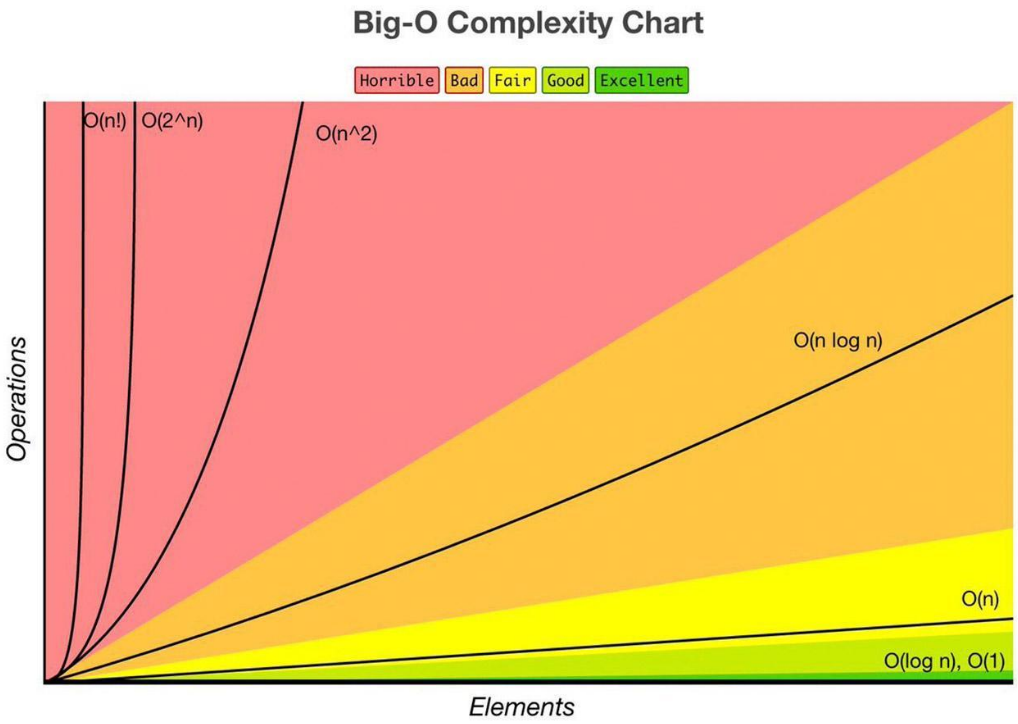
Singly Linked List Insertion O(n)
Doubly Linked List Insertion O(1)

What Does this mean ?!

Asymptotic Notations

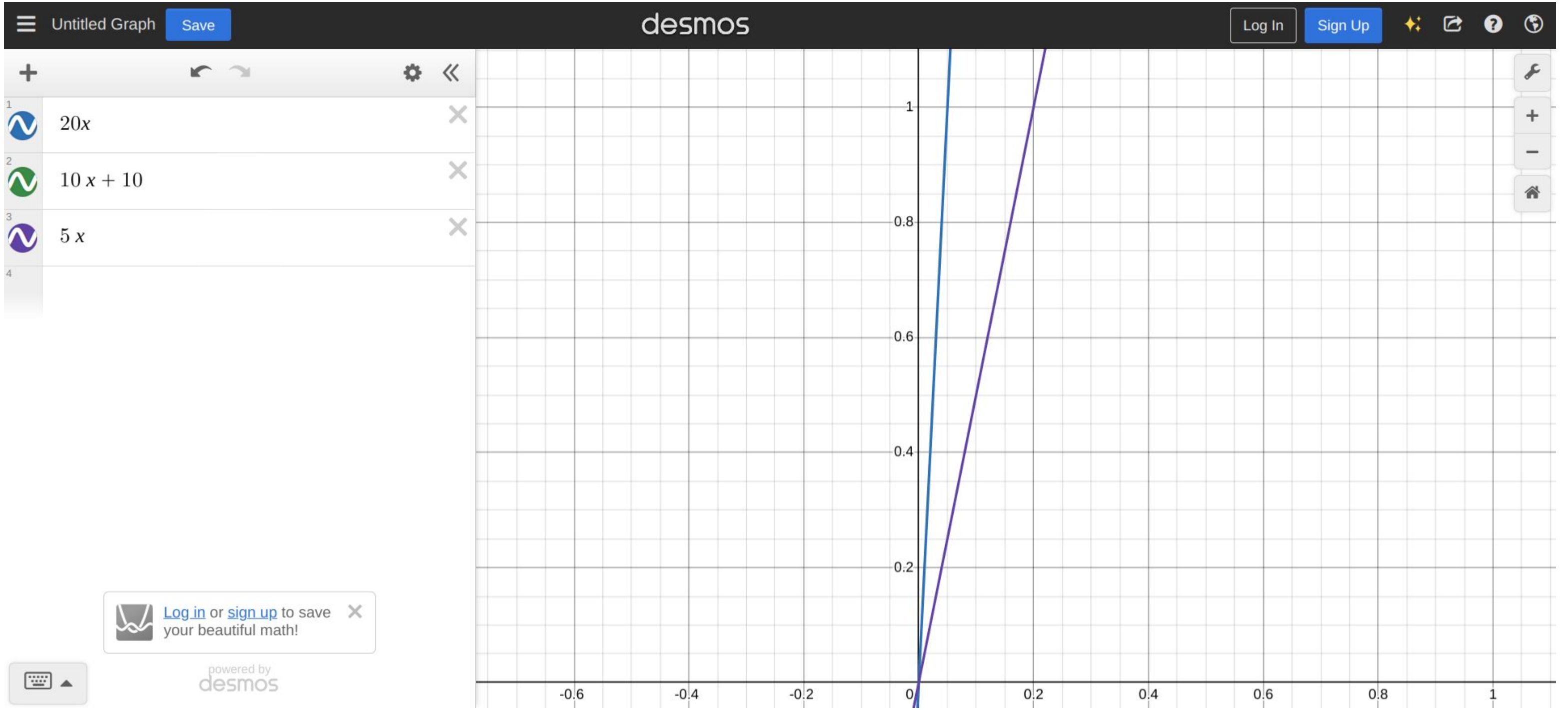
1. *Big-O Notation (O-notation)* ***Worst case***
2. *Omega Notation (Ω-notation)* ***Best case***
3. *Theta Notation (Θ-notation)* ***Average case***

Big-O complexity chart



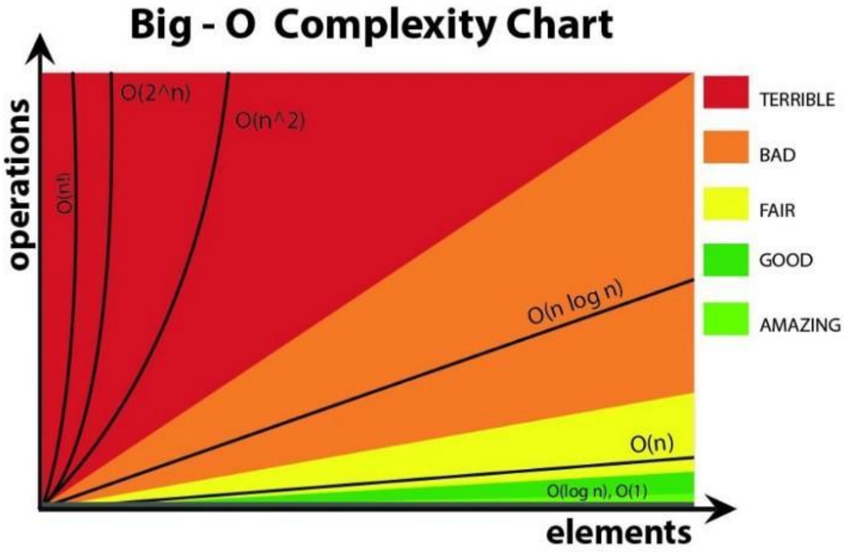
Big-O Chart: http://treeindev.net/article/algorithm-complexity-analysis

Open Desmos



Big-O Chart: http://treeindev.net/article/algorithm-complexity-analysis

Growth orders



O(1) — constant time
O(log n) — logarithmic
O(n) — linear time
O(n²) — quadratic
O(2^n) — exponential
O(n!) — factorial

Example 1

Code Analysis

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

int main()
{
     int x = 0;  //    T(1)
     int y = 0;  //    T(1)
     int N = 4;  //    T(1)
     int l = 4;  //    T(1)


     for (int i = 0; i < N; i++) {   //T(3)
          x = x + 10;  //T(N)
     }

     for (int i = 0; i < l; i++) {  //T(3)
          y = y + 40; //T(N)
     }
```

Big-O Analysis

- **We have to deduce the Big-O complexity**

1. To prove that: $1+1+1+1+3+5N+3+5N = O(n)$
2. Solve for $C=15$ and $N_o>=2$
3. $15N>=1+1+1+1+3+5N+3+5N$
4. $15N>=10+10N$

$O(n)$

Omega Notation (Ω) Analysis

1. To prove that: $10+10N = \Omega(n)$
2. Solve for $C=3$ and $N_0>=1$
3. $3N<=10+10N$

$\Omega(n)$

Theta Notation(Θ) Analysis

- We have to deduce the Θ

1. To prove that: $10+10N = \Theta(n)$
2. Solve for $C1=3$, $N_o>=2$ and $C2=15$
3. $c1 .g(N) <= f(N) <= c2 .g(N)$
4. $3N<=10+4N<=15 N$

$\Theta(n)$

Example 2

Code Analysis

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


int main()
{
    int a = 0;     //T(1)
    int N = 4;     //T(1)

    for (int i = 0; i < N; i++) {  //T(3)
        for (int j = 0; j < N; j++) { //T(3)
            a = a + j;   //T(2)
        }

    }

    for (int k = 0; k < N; k++) {  //T(3)
        a = a + k;     //T(2)
    }

}
```

Big-O Analysis

- **We have to deduce the Big-O complexity**

1. To prove that: 2+N(3+N(3+2))+3N= O(n^2)
2. Solve for C=8 and No>=2 || g(n)= $8N^2 + N$
3. $8N^2 + N$ >= 2+N(3+N(3+2))+3N
4. $8N^2 + N >= 5N^2 + 6N + 2$

O(n^2)

Omega Notation (*Ω)* Analysis

1. To prove that: 2+N(3+N(3+2))+3N= $\Omega(n^2)$
2. Solve for C=1 and N_0>=1 || g(n)= $N^2 + N$
3. $N^2 + N$ =< 2+N(3+N(3+2))+3N
4. $N^2 + N$ <= $5N^2 + 6N + 2$

$$\Omega(n^2)$$

Theta Notation(Θ) Analysis

- We have to deduce the Θ

1. To prove that: $5N^2 + 6N + 2 = \Theta(n^2)$

1. Solve for C1=3 , N_o>=2 and C2=8

2. c1 .g(n) <= f(n) <= c2 .g(n)

3. $N^2 + N <= 5N^2 + 6N + 2 <= 8N^2 + N$

$$\Theta(n^2)$$

Note

- The Big O can be upper-bounded with any larger degree e.g (n^3, n^4, n^5) but we always pick the closest value for the given equation which was (n^2) in our example.

- The Omega can be lower-bounded with smaller degrees as well. E.g(n,log(n)) but we always pick the closest value for the equation as well.

Example 3

Code Analysis

```
#include <iostream>

int main() {

    int N=20;

   for (int j = 0; j <= N; j = j * 2) {

    cout << k << ' ';
}

}
```

Big-O Analysis

- We have to deduce the Big-O complexity

- $g(n)=c1.f(n)=6+\log(N)$ $c1=6$ $N_o>=1$
- $6+\log(N)>=1+5\log(N)$

$O(\log(n))$

Omega Notation (Ω) Analysis

- c1.g(n) = log(n)   No>=1 c1=1
- 1+5 log(n) >= log(n)

Ω(log(n))

Theta Notation(Θ) Analysis

- We have to deduce the Θ

- $\log(n) \le 1 + 5\log(n) \le 6 + 6\log(n)$

c1=1 c2=6 No >=1

Θ(log(n))

Task 1

- Analyze the code and calculate the asymptotic functions.

```
using namespace std;

int main() {

    int N=30;
    int j=20;
    for(int i=0;i<N;i+=5){
        cout<<i;
        cout<<j;
    }
}
```

Task 2

- Analyze the code and calculate the asymptotic functions.

```
int main() {

    int N=30;
    int M=20;
    int Q=40;
    for(int i=0;i<N;i++){
        for(int j=0;i<M;j++){
            for(int l=0;i<N;l++){
                cout<<i;
                cout<<j;
                cout<<l;
    }
```

Thank you.