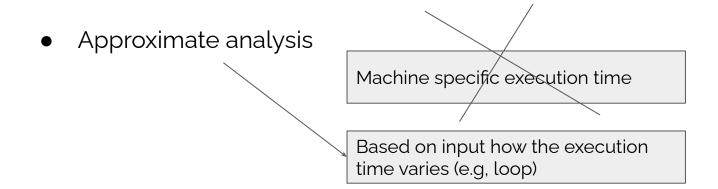
CSE 246: Algorithms

Analysis of Algorithms

Asymptotic Analysis using Asymptotic Notation



Asymptotic Notation

- Θ (Theta notation)
- O (Big O notation)
- Ω (Omega notation)

O (Big O notation)

A Solution's upper bound

```
void f4(int n){
    if (n%2 == 0)
        return;
    for(int i=0;i<n;i++){
        cout<<i<<endl;
        for(int j=0; j<n;j++){
            cout<<j<<endl;
            for(int k=0; k<n;k++){
                 cout<<k<<endl;
            }
        }
    }
    return;
}</pre>
```

```
void f1(int n){
    return;
}
```

```
void f2(int n){
   for(int i=0;i<n;i++){
      cout<<i<endl;
   }
   return;
}</pre>
```

```
void f3(int n){
    for(int i=0;i<n;i++){
        cout<<i<<endl;
        for(int j=0; j<n;j++){
            cout<<j<<endl;
        }
    }
    return;
}</pre>
```

O (Big O notation)

Definition: Let **g** and **f** be functions from the set of natural numbers to itself. The function **f** is said to be O(g), if there is a constant c > 0 and a natural number n_a such that $f(n) \le c * g(n)$ for all $n >= n_a$.

```
O(g(n)) = { f(n): there exist positive constants c and n<sub>o</sub> such that o <= f(n) <= c*g(n) for all n >= n<sub>o</sub>}
```

```
void f4(int n){
   if (n%2 == 0)
        return;
    for(int i=0;i<n;i++){
        cout<<i<<endl:
        for(int j=0; j<n; j++){
            cout<<j<<endl;
            for(int k=0; k<n; k++){
                cout<<k<<endl;
                for(int l=0; l<5; l++) {
                    cout<<l<<endl;
    return;
```

O (Big O notation)

- Constant Multiplication: If $f(n) = c^* k(n)$, then O(f(n)) = O(k(n)); where c is a nonzero constant.
- Polynomial Function: If $f(n) = a_0 + a_1 \cdot n + a_2 \cdot n^2 + --- + a_m \cdot n^m$, then $O(f(n)) = O(n^m)$.
- Summation Function: If $f(n) = f_1(n) + f_2(n) + --+ f_m(n)$ and $f_i(n) ≤ f_{i+1}(n) ∀ i=1, 2,..., m$, then $O(f(n)) = O(\max(f_1(n), f_2(n), ..., f_m(n)))$.
- Logarithmic Function: If $f(n) = \log_a n$ and $g(n) = \log_b n$, then O(f(n)) = O(g(n))

Ω (Omega Notation)

→ Lower Bound Calculation

```
\Omega (g(n)) = { f(n): there exist positive constants c and n<sub>o</sub> such that 0 <= c*g(n) <= f(n) for all n >= n<sub>o</sub>}.
```

```
void f4(int n){
    if (n%2 == 0)
        return;
    for(int i=0;i<n;i++){</pre>
        cout<<i<<endl;
        for(int j=0; j<n; j++){
            cout<<j<<endl;
            for(int k=0; k<n;k++){
                 cout<<k<<endl;
                 for(int l=0; l<5; l++) {
                     cout<<l<<endl;
    return;
```

Θ (Theta notation)

Merges Upper bound and lower bound

```
\Theta(g(n)) = {f(n): there exist positive constants c1, c2 and n_o such that 0 <= c1*g(n) <= f(n) <= c2*g(n) for all n >= n_o}
```

```
void f4(int n){
    if (n%2 == 0)
        return;
    for(int i=0;i<n;i++){
        cout<<i<<endl;
        for(int j=0; j<n;j++){
            cout<<j<<endl;
            for(int k=0; k<n; k++){
                cout<<k<<endl;
                for(int l=0; l<5; l++) {
                    cout<<l<<endl;
    return;
```

Sources

https://www.geeksforgeeks.org/analysis-of-algorithms-set-1-asymptotic -analysis/

Thank You