Algorithm and Algorithm Analysis

- Algorithm - a well defined set of instructions to complete a task
- Algorithm analysis - estimate the time/space required by an algorithm
  - Optimization
  - Comparison

Algorithm != Code

- The same algorithm can be implemented by different people, in different languages, under different conditions
- To analyze algorithm, we need to abstract away the implementation details

The Model

- A computer with infinite amount of memory
- Simple instructions only (addition, multiplication, assignment etc.)
- Sequential execution
- Each instruction takes one unit of time

Running Time

- Given size of the input N
  - T_{best}(N) – best-case running time
  - T_{worst}(N) – worst-case running time
  - T_{avg}(N) – average running time

Example 1: Max

```java
int max( int a[] )
{
    int max=a[0];
    for( int i=1 ; i < a.length ; ++i )
        if( max < a[i] ) max = a[i];
    return max;
}
```

- N??, T_{best}(N)??, T_{worst}(N)??, T_{avg}(N)??
Example 2: Search

```java
int search( int x, int a[] )
{
    for( int i=0 ; i < a.length ; ++i )
        if( a[i] == x ) return 1;
    return -1;
}
```

◆ $T_N$, $T_{best}(N)$, $T_{worst}(N)$, $T_{avg}(N)$?

The Big-O Notation

◆ $T(N) = O(f(N))$ if there are positive constants $c$ and $n_0$ such that $T(N) \leq cf(N)$ when $N \geq n_0$

◆ $O(f(N))$ – Order of $f(N)$

Time Complexity

◆ In the Big-O notation

<table>
<thead>
<tr>
<th>Running Time $T(N)$</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>$2057 + N$</td>
<td>$O(N)$</td>
</tr>
<tr>
<td>$100N^2$</td>
<td>$O(N^2)$</td>
</tr>
<tr>
<td>$1321312 + 23244255N + N^2$</td>
<td>$O(N^2)$</td>
</tr>
</tbody>
</table>

Two Simple Rules

◆ Constants do not matter
◆ Higher-order terms dominate lower-order terms

Growth Rate of Some Functions

<table>
<thead>
<tr>
<th>$N$</th>
<th>$\log N$</th>
<th>$N \log N$</th>
<th>$N^2$</th>
<th>$2^n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>24</td>
<td>64</td>
<td>256</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>64</td>
<td>256</td>
<td>65536</td>
</tr>
<tr>
<td>32</td>
<td>5</td>
<td>160</td>
<td>1024</td>
<td>4294967296</td>
</tr>
</tbody>
</table>

Growth Rate of $N^2$ and $N^2+4N+20$

<table>
<thead>
<tr>
<th>$N$</th>
<th>$N^2$</th>
<th>$N^2+4N+20$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>100</td>
<td>130</td>
</tr>
<tr>
<td>50</td>
<td>2500</td>
<td>3220</td>
</tr>
<tr>
<td>100</td>
<td>10000</td>
<td>10420</td>
</tr>
<tr>
<td>1000</td>
<td>1000000</td>
<td>1004020</td>
</tr>
<tr>
<td>10000</td>
<td>10000000</td>
<td>100040020</td>
</tr>
</tbody>
</table>
Understand the Definition

How do we show $2057+N = O(N)$?

$T(N) = 2057+N$ and $f(N) = N$, and based on the definition, we need to find $c$ and $n_0$ such that $2057+N \leq cN$ for $N>n_0$.

We can pick $c=2$ and $n_0=2057$. Because $2057 < N$ for $N>2057$ and $N \leq N$ we have $2057+N \leq 2N$ for $N>2057$.

Typical Complexities

<table>
<thead>
<tr>
<th>Time Complexity</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(1)$</td>
<td>Constant</td>
</tr>
<tr>
<td>$O(\log N)$</td>
<td>Logarithmic</td>
</tr>
<tr>
<td>$O(\log^2 N)$</td>
<td>Log-squared</td>
</tr>
<tr>
<td>$O(N)$</td>
<td>Linear</td>
</tr>
<tr>
<td>$O(N\log N)$</td>
<td>Log-linear</td>
</tr>
<tr>
<td>$O(N^2)$</td>
<td>Quadratic</td>
</tr>
<tr>
<td>$O(N^3)$</td>
<td>Cubic</td>
</tr>
<tr>
<td>$2^n$</td>
<td>Exponential</td>
</tr>
</tbody>
</table>

Example 3: Binary Search

```java
int binarySearch( int x, int a[] )
{
    int index = -1, left = 0, right = a.length-1, mid;
    while( left <= right ){
        mid = (left+right)/2;
        if( a[mid] > x ) right = mid-1;
        else if( a[mid] < x ) left = mid+1;
        else { index = mid; break; }
    }
    return index;
}
```

Time complexity: best-case?, worst-case?, average case?

Beyond Basics

- $\Omega$, $\Theta$, and $\Omega$
- Proofs
- Complex complexities, e.g.

$$T(N) = T(N-1) + T(N-2) + 2$$