Lecture 6

Time Complexity: Asymptotic Notations

Introduction

- How running time of an algorithm increases with the size of the input in the limit?
- Order of growth of the running time of an algorithm.
  - A simple characterization of algorithm’s efficiency.
- Allows to compare the relative performance of alternative algorithms.
- Algorithm that is asymptotically more efficient will be the best choice for all but very small inputs.

Example: $A[i] = A[0] + A[1] + \ldots + A[i]$

Algorithm **arrayElementSum**(A,N)

Input: An array **A** containing **N** integers.

Output: An updated array **A** containing **N** integers.

1

```
for i = 1 to N – 1 do
    sum = 0
    for j = 0 to i do
        sum = sum + A[j]
    A[i] = sum
```

Option 2 is better

2

```
for i = 1 to N – 1 do
    A[i] = A[i] + A[i – 1]
```

Analysis of Algorithms

- Identify primitive operations, i.e., low level operations independent of programming language.
- Example:
  - Data movement operations (assignment).
  - Control statements (branch, method call, return).
  - Arithmetic and Logical operations.
- Primitive operations can easily be identified by inspecting the pseudo-code.

Contd…

Algorithm	Cost	Frequency
1. **for** i = 1 to N – 1 **do**	c1	N
2. sum = 0	c2	N – 1
3. **for** j = 0 to i **do**	c3	$\sum_{i=1}^{N-1}(i+2)$
4. sum = sum + A[j]	c4	$\sum_{i=1}^{N-1}(i+1)$
5. A[i] = sum	c5	N – 1

$$c1N + c2(N-1) + c3\sum_{i=1}^{N-1}(i+2) + c4\sum_{i=1}^{N-1}(i+1) + c5(N-1)$$

$$c1N + c2N - c2 + c3\left(\frac{N(N-1)}{2}\right) + 2.c3.N - 2.c3 + c4\left(\frac{N(N-1)}{2}\right) + c4N - c4 + c5N - c5)$$

$$N^2\left(\frac{c3}{2} + \frac{c4}{2}\right) + N\left(c1 + c2 + \frac{3}{2}c3 + \frac{c4}{2} + c5\right) - (c2 + 2.c3 + c4 + c5)$$

Contd…

	Cost	Frequency
1. **for** i = 1 to N – 1 **do**	c1	N
2. A[i] = A[i] + A[i – 1]	c2	N – 1

$$c1N + c2(N-1)$$

$$N(c1 + c2) - c2$$

Asymptotic Notation

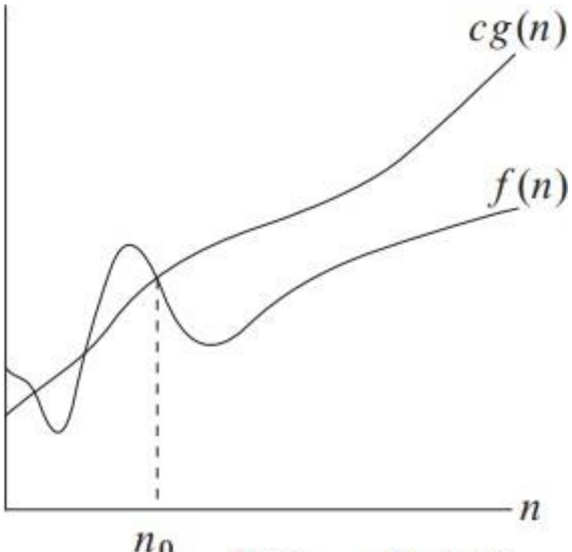
- Describes the running times of algorithms as a function of the size of its input.

- The running time of an algorithm can be
  - Worst-case running time
  - Average-case running time
  - Best-case running time

Big-Oh Notation

- Gives only an asymptotic upper bound.

- For a given function g(n), O(g(n)) (pronounced “big-oh of g of n” or sometimes just “oh of g of n”) denotes the set of functions



$f(n) = O(g(n))$

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

Example: Big-Oh

Show that: $\mathbf{n^2/2 - 3n = O(n^2)}$

- Determine positive constants $\mathbf{c_1}$ and $\mathbf{n_0}$ such that

$$\mathbf{n^2/2 - 3n \le c_1 n^2} \text{ for all } \mathbf{n \ge n_0}$$

- Diving by $\mathbf{n^2}$

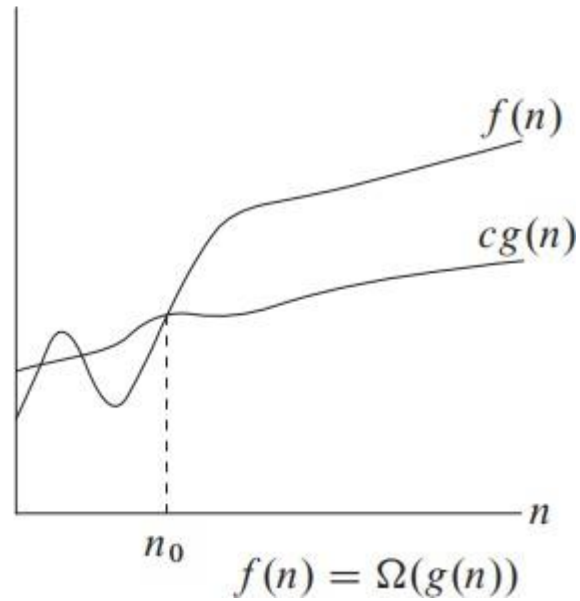
$$\mathbf{1/2 - 3/n \le c1}$$

- For: $\mathbf{n = 1}$, $\mathbf{1/2 - 3/1 \le c1}$ (Holds for $\mathbf{c1 \ge 1/2}$)

  $\mathbf{n = 2}$, $\mathbf{1/2 - 3/2 \le c1}$ (Holds and so on…)
- The inequality holds for any $\mathbf{n \ge 1}$ and $\mathbf{c_1 \ge 1/2}$.
- Thus by choosing the constant $\mathbf{c_1 = 1/2}$ and $\mathbf{n_0 = 1}$, one can verify that $\mathbf{n^2/2 - 3n = O(n^2)}$ holds.

Big-Omega Notation

- Gives only an asymptotic lower bound.
- For a given function g(n), Ω(g(n)) (pronounced "big-omega of g of n" or sometimes just "omega of g of n") denotes the set of functions



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

Example: Big-Omega

Show that: $\mathbf{n^2/2 - 3n = \Omega(n^2)}$

- Determine positive constants c_1 and n_0 such that

$$c_1 n^2 \le n^2/2 - 3n \text{ for all } n \ge n_0$$

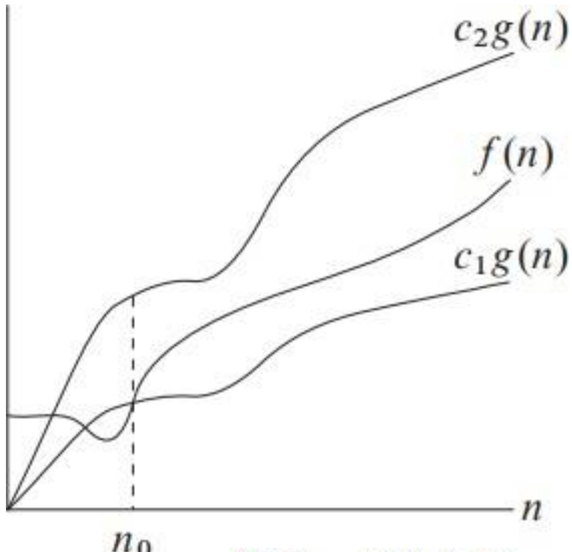
- Diving by n^2

$$c_1 \le 1/2 - 3/n$$

- For: $n = 1$, $c_1 \le 1/2 - 3/1$ (Not Holds)
  $n = 2$, $c_1 \le 1/2 - 3/2$ (Not Holds)
  $n = 3$, $c_1 \le 1/2 - 3/3$ (Not Holds)
  $n = 4$, $c_1 \le 1/2 - 3/4$ (Not Holds)
  $n = 5$, $c_1 \le 1/2 - 3/5$ (Not Holds)
  $n = 6$, $c_1 \le 1/2 - 3/6$ (Not Holds and **Equals ZERO**)
  $n = 7$, $c_1 \le 1/2 - 3/7$ or $c_1 \le (7-6)/14$ or $c_1 \le 1/14$ (Holds for $c_1 \le 1/14$)
- The inequality holds for any $n \ge 7$ and $c_1 \le 1/14$.
- Thus by choosing the constant $c_1 = 1/14$ and $n_0 = 7$, one can verify that $n^2/2 - 3n = \Omega(n^2)$ holds.

Theta Notation

- Gives an asymptotic tight bound.
- Function f(n) belongs to the set θ(g(n)) if there exist positive constants c_1 and c_2 such that it can be "sandwiched" between $c_1g(n)$ and $c_2g(n)$, for sufficiently large n.



$f(n) = \Theta(g(n))$

- For a given function g(n), θ(g(n)) denotes the set of functions:

$$\Theta(g(n)) = \{ f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \} .^1$$

Example: Theta

- Show that: $\mathbf{n^2/2 - 3n = \theta(n^2)}$
- Determine positive constants $\mathbf{c_1}$, $\mathbf{c_2}$, and $\mathbf{n_0}$ such that

$$c_1 n^2 \le n^2/2 - 3n \le c_2 n^2 \text{ for all } n \ge n_0$$

- Diving by $\mathbf{n^2}$

$$c_1 \le 1/2 - 3/n \le c_2$$

- Right Hand Side Inequality holds for any $\mathbf{n \ge 1}$ and $\mathbf{c_2 \ge 1/2}$.
- Left Hand Side Inequality holds for any $\mathbf{n \ge 7}$ and $\mathbf{c_1 \le 1/14}$.
- Thus by choosing the constants $\mathbf{c_1 = 1/14}$ and $\mathbf{c_2 = 1/2}$ and $\mathbf{n_0 = 7}$, one can verify that $\mathbf{n^2/2 - 3n = \theta(n^2)}$ holds.


1200
1000
800
600
400
200
0
200
400
600
800
1000
1200
1400
1600
1800
2000
2200
1
$y = \log n$
2
$y = \sqrt{n}$
3
$y = n$
4
$y = n \log n$
5
$y = 2^n$

o-Notation

- o-notation denotes an upper bound that is not asymptotically tight.
- Formally o(g(n)) ("little-oh of g of n") is defined as the set

$$o(g(n)) = \{ f(n) : \text{ for any positive constant } c > 0, \text{ there exists a constant } n_0 > 0 \text{ such that } 0 \le f(n) < cg(n) \text{ for all } n \ge n_0 \} .$$

- For example, $2n = o(n^2)$, but $2n^2 \mathrel{!=} o(n^2)$.
- Intuitively, in o-notation, the function f(n) becomes insignificant relative to g(n) as n approaches infinity; that is, $\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$

ω-Notation

- ω-notation denotes a lower bound that is not asymptotically tight.
- One way to define it is as f(n) ∈ ω(g(n)) if and only if g(n) ∈o(f(n)).
- Formally, ω(g(n)) ("little-omega of g of n") is defined as the set

$$\omega(g(n)) = \{f(n) : \text{for any positive constant } c > 0 \text{, there exists a constant } n_0 > 0 \text{ such that } 0 \le cg(n) < f(n) \text{ for all } n \ge n_0\} .$$

- For example, $n^2/2 = \omega(n)$, but $n^2/2 \mathrel{!=} \omega(n^2)$.
- The relation f(n) ∈ω(g(n)) implies that $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, if limit exists.

Reference

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to algorithms. MIT press.

Asymptotic Analysis Examples

1. Sequence of statements :

```
statement 1;
statement 2;
-
-
statement n;
```

Complexity: O(1)

2. If – else statements:

```
if (condition)
        { statement 1;
        -- }
else
        { statement 2;
        -- }
```

Complexity: O(1)

Asymptotic Analysis Examples

3. For and While Loops

```
for (i = 0; i < N; i++)
{ sequence of statements }
```

Complexity: O(N)

```
for (i = 0; i < N; i++) {
        for (j = 0; j < M; j++) {
                sequence of statements }}
```

Complexity: O(N*M)

```
for (i = 0; i < N; i++) {
        for (j = i+1; j < N; j++) {
                sequence of statements  }}
```

Complexity: O(N^2)

```
for (i = 0; i < N; i++) {sequence of statements}
for (j = 0; j < M; j++) {sequence of statements}
```

Complexity: O(N+M)

Asymptotic Analysis Examples

```
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) { sequence of statements }}

for (k = 0; k < N; k++) { sequence of statements }
```

Complexity: O(N^2)
Max(O(N^2),O(N))

```
for (i = 0; i < N; i++) {
    for (j = N; j > i; j--) {
        sequence of statements } }
```

Complexity: O(N^2)

```
for (k = 1; k <= N; )
        {       k = k * 2;      }
```

Complexity: O(log N)

```
for (i = N; i > 0; )
        {       i = i / 2;      }
```

Complexity: O(log N)

Thank You