CSC 2221: Algorithms

Lecture 2

Supta Richard Philip
Lecturer

Department of CSE
American International University(AIUB).

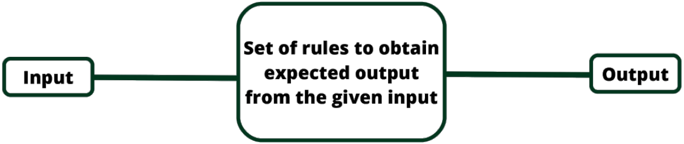
AIUB, January 2023

Table of Contents

Table of Contents

What is Algorithm?

- Informally, an algorithm is any well-defined **computational procedure** that takes some value, or set of values, as **input** and produces some value, or set of values, as **output**.[1]
- An algorithm is a sequence of computational steps that transform the input into the output.
- An algorithm as a tool for solving a well-specified computational problem.



Example of Algorithm

- Formally define the **sorting problem**:
- **Input:** A sequence of n number $(a_1, a_2, ...a_n)$
- **Output:** A permutation (reordering) $a'_1, a'_2, ..., a'_n$ of the input sequence such that $a'_1 \leq a'_2 \leq ... \leq a'_n$
- An input sequence (31, 41, 59, 26, 41, 58) is called an instance of the sorting problem.
- Sorting algorithm returns as output the sequence (26, 31, 41, 41, 58, 59)

What kinds of problems are solved by algorithms?

- **The Human Genome Project** : identifying all the 100,000 genes in human DNA. determining the sequences of the 3 billion chemical base pairs that make up human DNA(A,G,C,T) and developing tools for data analysis.
- **Network and Graph algorithm** : The Internet enables people all around the world to quickly access and retrieve large amounts of information.
- **E-commerce**: depends on the privacy of information such as credit card numbers, bank statements that depends on public-key cryptography and digital signatures , which are based on numerical algorithms and number theory.

Table of Contents

Characteristics of Algorithm

- **Unambiguity:** A perfect algorithm is defined as unambiguous, which means that its instructions should be clear and straightforward.
- **Finiteness:** An algorithm must be finite. Finiteness in this context means that the algorithm should have a limited number of instructions, i.e., the instructions should be countable.
- **Effectiveness:** Because each instruction in an algorithm affects the overall process, it should be adequate.

Table of Contents

Analyzing algorithms: RAM Model

- Analyze an algorithm, we must have a model of the implementation technology that we will use, including a model for the resources of that technology and their costs.
- We shall assume a generic one processor, random-access machine (RAM) model of computation as our implementation technology and understand that our algorithms will be implemented as computer programs.
- In the RAM model, instructions are executed one after another, with no concurrent operations.

Table of Contents

Find max from List

Algorithm 2 Find Max from List

procedure FINDMAX(A, n)
    $currentMax \leftarrow A[0]$ ▷ 2 steps
    for $i = 0; i < n; i++$ **do** ▷ i initialize 1, comparison n+1 and i increments n steps
        if $currentMax < A[i]$ **then** ▷ 2*n steps
            $currentMax = A[i]$ ▷ 2*n steps
        end if
    end for
    return $currentMax$ ▷ 1 steps
end procedure

- Sum of all, $f(n) = 2 + 1 + (n + 1) + n + 2n + 2n + 1 = 6n + 5$
- Drop lower order polynomial and coefficient.
  Time complexity is $O(n)$
- $f(n) = 2n^2 + 5n + 100$, Time complexity $= O(n^2)$

For Loop

Algorithm 1 For Loop

```
procedure ALGO1(A, n)
    for i = 0; i < n; i++ do        ▷ i initialize 1 times, comparison n+1 times
        //statement                 ▷ i increments n times and smt execute n times
    end for
end procedure
```

- Sum of all steps, $f(n) = 1 + (n + 1) + n + n$
- Time functions, $f(n) = 3n + 2$
- Time complexity of the algorithm is $O(n)$

Nested For Loop

Algorithm 5 Nested For Loop

```
procedure ALGO3(A, n)
    for i ← 0, n − 1 do                          ▷ n times
        for j ← 0, n − 1 do                      ▷ n*n times
            statement
        end for
    end for
end procedure
```

- Time functions, $f(n) = n^2 + n$
- Time complexity of the algorithm is $O(n^2)$

Exercise

- Matrix Sum
- Matrix multiplication

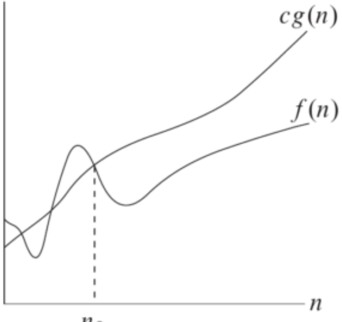
Algorithm 4 For Loop

```
procedure ALGO21(A, n, m)
    for i ← 0, n do                ▷ n times
        //statement
    end for
    for i ← 0, m do                ▷ m times
        //statement
    end for
end procedure
```

Table of Contents

O notation

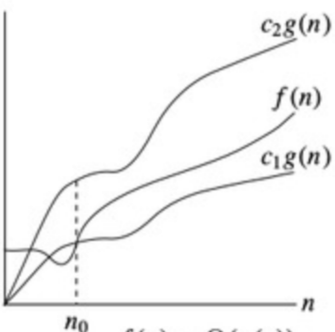
- An asymptotic upper bound(Worst Case), we use O notation. For a given function g(n), O(g(n)) is defined as: O(g(n)) = { f(n): there exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$}. As an example, let's have a look at the following figure:



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

Big Theta notation

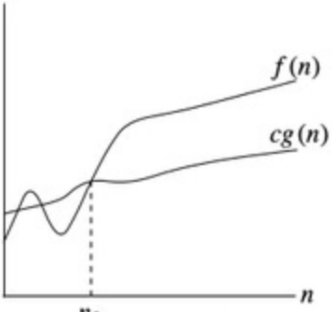
- An asymptotic tight bound(average Case), we use Θ notation. we denote by $\Theta(g(n))$, the set of functions Θ(g(n)) = { f(n): there exist positive constants c_1, c_2, and n_0 such that $0 \leq c_1.g(n) \leq f(n) \leq c_2.g(n)$ for all $n \geq n_0$}.
  As an example, let's have a look at the following figure:



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

Big Omega notation

- An asymptotic lower bound(best Case), we use Ω notation. we denote by $\Omega(g(n))$, the set of functions $\Omega(\mathrm{g(n)}) = \{$ f(n): there exist positive constants $cand\mathrm{n}_0$ such that $0 \leq c.g(n) \leq f(n)$ for all $n \geq n_0\}$.
  As an example, let's have a look at the following figure:



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

Table of Contents

Growth Rate

n $f(n)$	$\lg n$	n	$n \lg n$	n^2	2^n	$n!$
10	0.003 μs	0.01 μs	0.033 μs	0.1 μs	1 μs	3.63 ms
20	0.004 μs	0.02 μs	0.086 μs	0.4 μs	1 ms	77.1 years
30	0.005 μs	0.03 μs	0.147 μs	0.9 μs	1 sec	8.4×10^{15} yrs
40	0.005 μs	0.04 μs	0.213 μs	1.6 μs	18.3 min	
50	0.006 μs	0.05 μs	0.282 μs	2.5 μs	13 days	
100	0.007 μs	0.1 μs	0.644 μs	10 μs	4×10^{13} yrs	
1,000	0.010 μs	1.00 μs	9.966 μs	1 ms		
10,000	0.013 μs	10 μs	130 μs	100 ms		
100,000	0.017 μs	0.10 ms	1.67 ms	10 sec		
1,000,000	0.020 μs	1 ms	19.93 ms	16.7 min		
10,000,000	0.023 μs	0.01 sec	0.23 sec	1.16 days		
100,000,000	0.027 μs	0.10 sec	2.66 sec	115.7 days		
1,000,000,000	0.030 μs	1 sec	29.90 sec	31.7 years		

Growth rates of common functions measured in nanoseconds

$$n! \gg 2^n \gg n^3 \gg n^2 \gg n \log n \gg n \gg \log n \gg 1$$

Table of Contents

References

Introduction to Algorithms, Third Edition, Thomas H. Cormen, Charle E. Leiserson, Ronald L. Rivest, Clifford Stein (clrs).