Data Structures And Algorithms

Introduction

Phung Hua Nguyen

Department of Computer Science
University of Technology

Febuary 2020

Outline

Introduction

Data Structures

Algorithms

Lecturer In Charge

- **My name:** Nguyen Hua Phung
- **Email:** nhphung@hcmut.edu.vn
- **Website:** BKeL

References

Textbook

- ”Data Structures and Algorithm Analysis”, Clifford A. Shaffer, Dover Publications, 2013.

Reference Books

- ”C/C++: How to Program”, 7th Ed. – Paul Deitel and Harvey Deitel, Prentice Hall, 2012.
- ”Algorithms and Data Structures”, Kert Mehlhorn, Peter Sandler, Springer, 2008.
- ”Data Structures and Algorithms in C++”, A. Drozdek, Thomson Learning Inc., 2005.

Assessment

- Tutorial/Lab/Online: 10%
- Assignment: 30%
- Midterm: 10%
- Final: 50%

Note: Assignment is calculated by the following formula:

$$Assignment = 2 * \frac{A * B}{A + B}$$

where A is from some given projects and B is from some questions in midterm or final

Data Structures and Algorithms - Learning Outcomes

- ✓ Determine the complexity of simple algorithms
- ✓ Manipulate basic data structures such as list, tree and graph.
- ✓ Implement basic sorting and searching algorithms

Why Data Structures and Algorithms

- Computer Science can be defined as the study of **algorithms**, which are used to transform **data**
- Program = Data Structures + Algorithms (Niklaus Wirth)

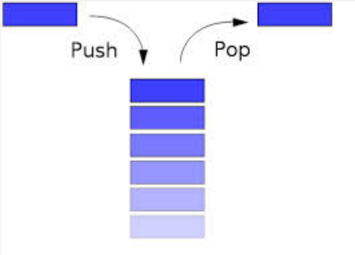
Some terms

- Type: collection of values that share similar characteristics
  - Primitive Type: int, float, char, bool, ...
  - Composite Type: struct, array, pointer,...
- Data Type: Type + operations on that type
- Abstract Data Type (ADT): the realization of a data type as a software component.
- Data Structure: implementation for an ADT.

Example of Abstract Data Type: Stack

.

- void push(E element)
- E pop()
- E top()
- bool isEmpty()
- int length()



Algorithm

Definition

An algorithm is a finite sequence of instructions to accomplish a particular task.

Algorithm 1: Greatest Common Divisor

---

input : Two positive intergers m and n
output: Greatest Common Divisor of m and n

1 Divide m by n and let r be the remainder.
2 If r = 0, the algorithm terminates. n is the output.
3 set m $\leftarrow$ n, n$\leftarrow$ r, and go back to Step 1

Properties

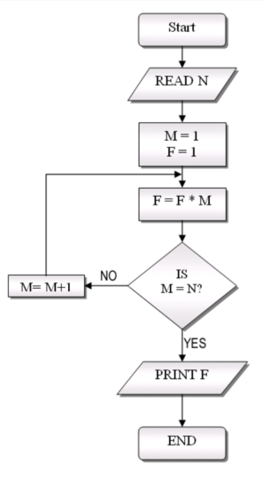
- Input: there are zero or more quantities which are externally supplied;
- Output: the quantity is produced;
- Definiteness: clear and unambiguous;
- Finiteness: terminate after a finite number of steps;
- Effectiveness: every step must be basic and essen-

Some Basic Algorithms

- Sorting
  - Bubble Sort
  - Shell Sort
  - Quick Sort
  - Heap Sort
- Searching
  - Linear Searching
  - Binary Searching
  - Hashing
- Operations on Tree
- Operations on Graph

Algorithm Representations

Flow chart



Pseudocode

Algorithm 2: Factorial Algorithm

input : non-negative integer N
output: Factorial of N

```
M ← 1;
F ← 1;
while M ⩽ N do
    F ← F * M;
    M ← M + 1;
end
return F;
```

Algorithm Efficiency

- How fast an algorithm is?
- How much memory does it consume?

For example, the running time of two different algorithms are as follows:

n	10	20	50	100	1000	5000
Algorithm 1	0.00s	0.01s	0.05s	0.47s	23.92s	47min
Algorithm 2	0.05s	0.05s	0.06s	0.11s	0.78s	14.22s

where n is the size of input

Computational Complexity

- Computational complexity: measure of the difficulty degree (time or space) of an algorithm.
- General format:

$$f(n)$$

  n is the size of a problem (the key number that determines the size of input data)

Linear Loop

Algorithm 3: Linear Loop with n=500

```
i ← 1;
n ← 500;
while i ⩽ n do
    application code;
    i ← i + 1;
end
```

The number of times the body of the loop is replicated is 500

Algorithm 4: Linear Loop with n=1000

```
i ← 1;
n ← 1000;
while i ⩽ n do
    application code;
    i ← i + 1;
end
```

The number of times the body of the loop is replicated is 1000

$$f(n) = n.T$$

Logarithmic Loops

Algorithm 5: Multiply Loop

```
i ← 1;
n ← 1000;
while i ⩽ n do
    application code;
    i ← i * 2;
end
```

Algorithm 6: Divide Loop

```
n ← 1000;
i ← n;
while i ⩾ 1 do
    application code;
    i ← i / 2;
end
```

The number of times the body of the loop is replicated is

$log_2 n$

$f(n) = (log_2 n).T$

Independent Nested Loops

Algorithm 7: Independent Nested Loops

```
n ← 10;
i ← 1;
while i ⩽ n do
    j ← 1;
    while j ⩽ n do
        application code;
        j ← j * 2;
    end
    i ← i + 1;
end
```

Iterations = Outer loop iterations × Inner loop iterations

$$f(n) = (n log_2 n).T$$

Dependent Nested Loops

Algorithm 8: Dependent Nested Loops

```
n ← 10;
i ← 1;
while i ⩽ n do
    j ← 1;
    while j ⩽ i do
        application code;
        j ← j + 1;
    end
    i ← i + 1;
end
```

The number of times the body of the loop is replicated is

$$1 + 2 + ... + n = n(n+1)/2$$
$$f(n) = (n(n+1)/2).T$$

Asymptotic Complexity

- Asymptotic complexity is used when comparing algorithm efficiency.
- Algorithm efficiency is considered with only big problem sizes.
- An exact measurement of an algorithm's efficiency is not necessary.

Complexity Notation

- Big-O: notation for upper bound

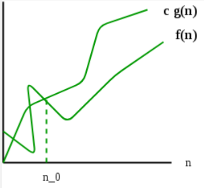
$$f(n) \in O(g(n)) \iff \begin{array}{c} \exists C > 0, \exists N_0 > 0, \forall n \geq N_0 : \\ |f(n)| \leq C * |g(n)| \end{array}$$

- Big-Ω: notation for lower bound

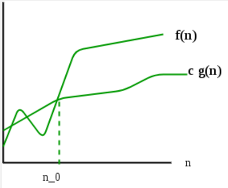
$$f(n) \in \Omega(g(n)) \iff \begin{array}{c} \exists C > 0, \exists N_0 > 0, \forall n \geq N_0 : \\ |f(n)| \geq C * |g(n)| \end{array}$$

- Big-Θ: notation for lower and upper bounds.

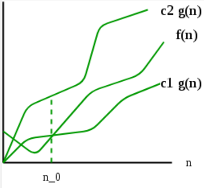
$$f(n) \in \Theta(g(n)) \iff \begin{array}{c} f(n) \in O(g(n)) \text{ and} \\ f(n) \in \Omega(g(n)) \end{array}$$



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



f(n) = Omega(g(n))



f(n) = theta(g(n))

Big-O Notation

- Set the coefficient of the term to one.
- Keep the largest term and discard the others.

Algorithm 9: Linear Logarithmic

```
n ← 10;
i ← 1;
while i ⩽ n do
    j ← 1;
    while j ⩽ n do
        application code;
        j ← j * 2;
    end
    i ← i + 1;
end
```

$f(n) = (nlog_2 n).T \in O(nlog_2 n)$

Big-O Notation

Algorithm 10: Quadratic

```
n ← 10;
i ← 1;
while i ⩽ n do
    j ← 1;
    while j ⩽ i do
        application code;
        j ← j + 1;
    end
    i ← i + 1;
end
```

f(n) = $(n(n+1)/2)$.T = $\frac{T}{2}n^2 + \frac{T}{2}.n \in O(n^2)$

Standard Measures of Efficiency

Efficiency	Big-O	Iterations	Est. Time
logarithmic	$O(log_2 n)$	14	microseconds
linear	$O(n)$	10,000	.1 seconds
linear logarithmic	$O(nlog_2 n)$	140,000	2 seconds
quadratic	$O(n^2)$	$10,000^2$	15-20 min
polynomial	$O(n^k)$	$10,000^k$	hours
exponential	$O(2^n)$	$2^{10,000}$	intractable
factorial	$O(n!)$	$10,000!$	intractable

Assume instruction speed of 1 microsecond and 10 instructions in loop.
n = 10,000

Time Costing Operations

- The most time consuming: data movement to/from memory/storage.
- Operations under consideration:
  - Comparisons
  - Arithmetic operations
  - Assignments

Best, Average, Worst Cases

- Best case: when the number of steps is smallest.
- Worst case: when the number of steps is largest.
- Average case: in between.

Summary

- Data structures and algorithms are 2 important basic aspects of computer science
- They are combined in Abstract Data Types
- Algorithms are evaluated by the complexity

For Further Reading I

- Clifford A. Shaffer
  Data Structures and Algorithm Analysis, chapter 3.
  Free E-Book, 2012.
- Kert Mehlhorn and Peter Sandler
  Algorithms and Data Structures, chapter 2.
  Springer, 2008.