1 - Data Structures + Algorithms

Joseph Afework
CS 241

Dept. of Computer Science
California Polytechnic State University, Pomona, CA

Agenda

- Data Structures/ADT
- Algorithms
- Complexity of Algorithms
- Growth Rate
- Methodology
- Testing
- Additional Resources

Reading Assignment

- Read Chapter 1 - Bags
  - Chapter 1 Intro only…

- Read Chapter 4 - Measuring an Algorithm’s Efficiency
  - Sections 4.1- 4.22

Data Structures

Data Types: A group of concrete types and operations implemented in a specific programming languages.

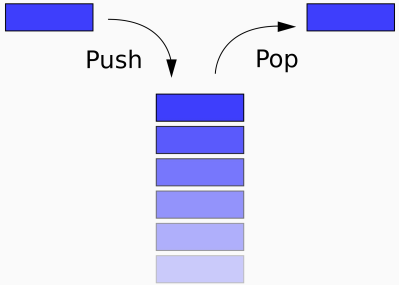
Ex. Java - int, bool, double etc…

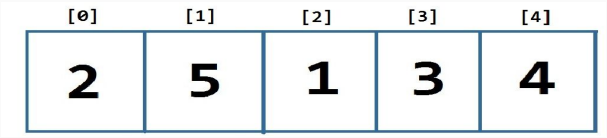
Data Structure: An implementation of data types and logic that aggregates data in specific way.

Ex. Array, list, queue, stacks, linked lists, queues, stacks, trees, graphs etc...

Linear Data Structures

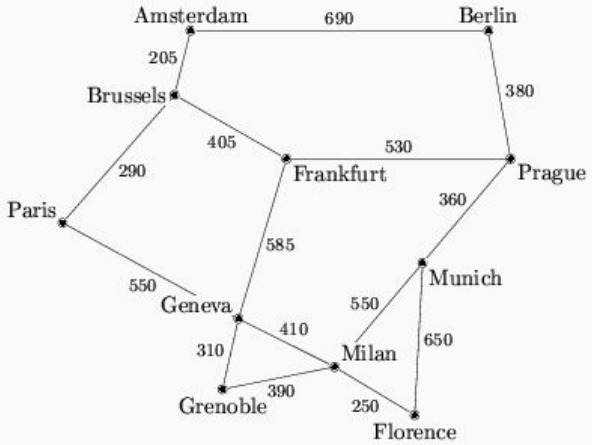
- Ex. Array, list, queue, stack etc..
- Data is stored and accessed in a linear sequence.
- Data access restriction is possible
  - FIFO (First In First Out) - queue
  - LIFO (Last in First Out) - stack

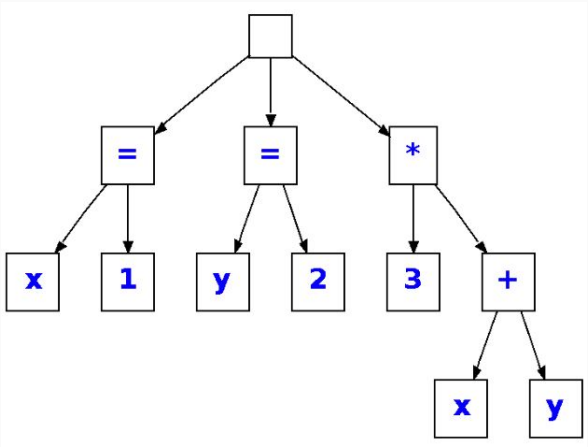




Non-linear Data Structures

- Ex. trees, graphs etc…
- Data may not be arranged or accessed sequentially
- Data access may be constrained by different rules





Abstract Data Types - ADT

Abstract Data Type (ADT): a general spec for a group of values and operations defined agnostic to a programming language.

Abstract Data Type + Implementation = Data Structure

- Ex. Stack and Queue
  - Both can be implemented using an array
  - Both can be implemented using a linked list
  - The concepts of a Stack and Queue can be separate from implementation.

Selecting the Right Data Structure

- Consider supported operations in the data structure:
  - Is the data structure limited or constrained operationally? - (Don’t fight the data)
  - Is the data structure efficient in storing the data? - Space Required
  - Is the run times of the data structure operations acceptable or time efficient?

Algorithms

Algorithm: A set of instructions for solving a problem in a defined way.

- Represented by Implementation, Pseudocode and Flowcharts

Algorithm = Data Structures + Logic + Time Complexity + Efficiency

Algorithm Efficiency

- Time Complexity
  - How long does the algorithm take to execute?

- Spacial Complexity
  - How much space or memory does an algorithm need to execute?

Analyzing an Algorithm

Big Omega (n) - Best Case Scenario

- $\Omega(f(n))$ - Lower Bound

Big Theta (n) - Average Case Scenario

- $\Theta(f(n))$

Big O(n) - Worst Case Scenario

- $O(f(n))$ - Upper Bound

Problem Size and Growth Rate

Problem Size

- A measure of the size of the problem being solved by an algorithm
- Ex. 10 iterations…. 1000 iterations… 1 million iterations...

Growth Rate

A function that describes the runtime complexity of an algorithm as it grows.

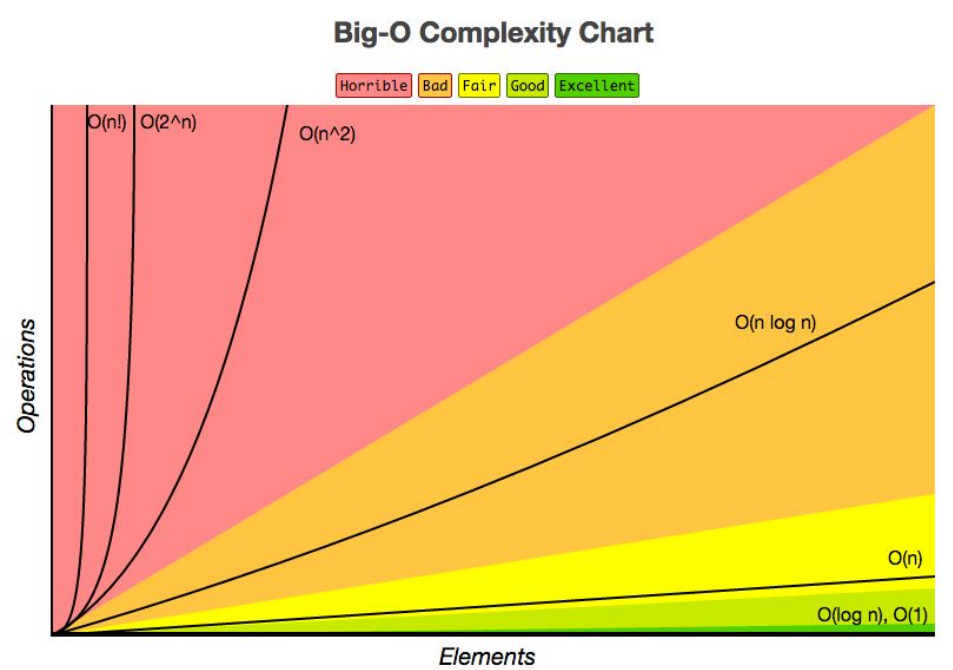
Algorithm Runtimes

FASTEST

- Constant function $f(n) = C$ -- *Constant algorithm does not depend on the input size.*
- Logarithm function $f(n) = \log n$ -- *Logarithm function gets slightly slower as n grows.*
- Linear function $f(n) = n$ -- *Whenever n doubles, so does the running time.*
- N-Log-N function $f(n) = n \log n$ -- *It grows a little faster than the linear function.*
- Quadratic function $f(n) = n^2$ -- *Whenever n doubles, the running time increases fourfold.*
- Cubic Function and Other Polynomials
- Exponential Function $f(n) = b^n$
- Factorial Function $f(n) = n!$

SLOWEST

Algorithm Runtimes Contd.



ICE 1.1 Algorithm Runtime

```
public void function(int x)
{
	for(int i = 0; i < this.list.length(); i++)
	{
		if(this.list[i] == x)
		{
			System.out.println(i);
			break;
		}
	}
}
```

Best :
Worst :
Avg:

```
public void function2(int x)
{
	for(int i = 0; i < this.list2.length; i++)
	{
		for(int j=0; j < this.list2[i].length(); j++)
		{
			if(this.list2[i][j] == x)
			{
				System.out.println(i + " " + j);
				break;
			}
		}
	}
}
```

Best :
Worst :
Avg:

ICE 1.1 Algorithm Runtime

```
public void function(int x)
{
	for(int i = 0; i < this.list.length(); i++)
	{
		if(this.list[i] == x)
		{
			System.out.println(i);
			break;
		}
	}
}
```

Best : O(1) - Constant Time
Worst : O(n) - Linear Time, Have to go through full list

```
public void function2(int x)
{
	for(int i = 0; i < this.list2.length; i++)
	{
		for(int j=0; j < this.list2[i].length(); j++)
		{
			if(this.list2[i][j] == x)
			{
				System.out.println(i + " " + j);
				break;
			}
		}
	}
}
```

Best : O(1) - Constant Time
Worst : O (N^2) - Quadratic Time - Slow

Analysing an Algorithm

- Go line by line to evaluate the complexity of the instruction
- Look for loops:
  - Nested loops = multiplication
  - Loops in scope = addition
- Drop least significant polynomial terms
- Drop coefficients

Ex. $O(4n^3+n^2+1) \sim= O(n^3)$

Methodology

1. Understand the problem
2. Design a solution
3. Implement the solution (potentially it's an algorithm)
4. Analyze the solution (Complexity, Correctness etc..)
5. Test and debug the algorithm
6. Maintenance

Testing

Testing a Program Tips

- **Understanding Program Behavior**: What are the possible results for your program?
- **Identify Boundary Values:** Test your algorithm by selecting test input that account for different error states **- Boundary Values** ex.) Array out of bounds etc..
- **Test Branching Logic:** Identify code paths that have conditional logic, and ensure your test values test these states.

ICE 1.2 FizzBuzz

"Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”."

- *http://wiki.c2.com/?FizzBuzzTest*

Ex. Output: 1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, Fizz Buzz, 16, 17, Fizz, 19, Buzz, Fizz, 22, 23, Fizz, Buzz, 26, Fizz, 28, 29, Fizz Buzz, 31, 32, Fizz, 34, Buzz, Fizz, ...

Testing Contd.

```
public String fizzbuzz(int input)
{
	if(input %3 == 0 && input %5 == 0)
	{
		return "FizzBuzz"
	}
	else if(input % 3 == 0)
	{
		return "Fizz";
	}
	else if(input % 5 == 0)
	{
		return "Buzz"
	}
	else
	{
		return "" + input;
	}
}
```

```
public void run()
{
	for(int i == 1; i < 101; i++)
	{
		String output = fizzbuzz(i);
		System.out.println( output );
	}
}
```

"Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”." - *http://wiki.c2.com/?FizzBuzzTest*

Testing Contd.

```
public String fizzbuzz(int input)
{
	if(input %3 == 0 && input %5 == 0)
	{
		return “FizzBuzz”
	}
	else if(input % 3 == 0)
	{
		return “Fizz”;
	}
	else if(input % 5 == 0)
	{
		return “Buzz”
	}
	else
	{
		return “” + input;
	}
}
```

```
public void run()
{
	for(int i == 1; i < 101; i++)
	{
		String output = fizzbuzz(i);
		System.out.println( output );
	}
}
```

What Values Should you test?

Testing Contd.

```
public String fizzbuzz(int input)
{
	if(input %3 == 0 && input %5 == 0)
	{
		return “FizzBuzz”
	}
	else if(input % 3 == 0)
	{
		return “Fizz”;
	}
	else if(input % 5 == 0)
	{
		return “Buzz”
	}
	else
	{
		return “” + input;
	}
}
```

```
public void run()
{
	for(int i == 1; i < 101; i++)
	{
		String output = fizzbuzz(i);
		System.out.println( output );
	}
}
```

What Values Should you test?

i = 2, 3, 5, 15, 20, 22

Learning Outcomes

- Understanding the analysis of algorithms

Additional Resources

- http://bigocheatsheet.com
- http://wiki.c2.com/?FizzBuzzTest

Image Attribution

Slide 5 - https://simple.wikipedia.org/wiki/Stack_(data_structure)#/media/File:Data_stack.svg

Slide 5 - http://cforbeginners.com/CSharp/SelectionSort.html

Silde 6 - https://cordis.europa.eu/tmr/src/res970302.htm

Slide 6 - http://stackoverflow.com/questions/5026517/whats-the-difference-between-parse-tree-and-ast