Introduction to Computing Fundamentals

Michael Brodskiy

Professor: M. Fanaei

January 13, 2023

- An algorithm is a sequence of instructions that solve a well-defined computational problem
- The statement of the problem specifies the desired input/output relationship
- The algorithm describes a computational procedure for generating the correct output for every input instance
- Computers keep getting faster, but data keeps growing at an even faster rate
- Our algorithms must be efficient in terms of resource usage
- Resources can be:
  - Computation time: number of instructions to execute
  - Space: amount of physical memory required by the algorithm
  - Other resources: network bandwidth
- Analyzing an algorithm means predicting the resources that an algorithm requires
- Most often it is the computational time that we want to measure
- Analysis of Algorithms
  - Other important features of a computer program beside performance:
    - Modularity
    - Maintainability
    - Robustness
    - Simplicity
    - Extensibility
    - Reliability
    - Readability

- Python
  - Python is very intuitive, highly readable, and easy to learn
  - Developed by Dutch programmer Guido Van Rossum in the late 1980s
  - Python is one of the most preferred programming languages for working in data analytics and machine learning domains
  - The most popular programming language as of March 2022
  - Python is a powerful, high-level, interpreted, object-oriented programming language
  - It has a simple and easy-to-use syntax
  - Portable across different platforms and operating systems
  - Has a rich variety of native data structures such as lists and dictionaries
  - Python code is typically 1/5 to 1/3 the size of equivalent C or Java code
  - There are two major versions of the python language:
    - Python 2.x is legacy
    - Python 3.x is the present and future of the language
  - Python 3 is not backward-compatible with Python 2
- An IDE (Integrated Development Environment) provides a rich set of features that make the developer's life easier, such as:
  - Debugger
  - Code profiling
  - Unit testing
  - Integration with version control systems like Git
  - An many more
- Many IDEs exist for Python: PyCharm, VSCode, PyDev, Eclispde, Komodo, Spyder, etc.
- Categories of Programming Languages:
  1. OO and Visual Language
  2. FORTRAN, C, Pascal
  3. High-Level Language
  4. Assembly Language
  5. Machine Language
  6. Hardware

- Low-level language is a programming language that provides little or no abstraction from a computer's instruction set architecture (ISA) commands or functions in the language map close to processor instructions
  - Generally, this refers to either machine code or assembly language
  - Tend to be relatively non-portable
- High-level language is a more understandable and portable language in which each statement accomplishes substantial tasks
  - Level of abstraction closer to problem domain
  - Provides for productivity and portability
- After compilation, a high-level language (say, C) becomes an Assembly Program
  - A textual representation of instructions
  - Human-readable format instructions
- After assembly language, it becomes a Machine Program
  - Binary digits (bits)
  - Encoded instructions and data
  - Computer-readable format instructions
- Compiler is a program for converting high-level code into low-level code or binary form
- Assembler is a utility program for converting assembly language code into executable machine code (1's and 0's)
- Machine code is the only language a computer can process directly
- Compilers
  - A compiler is a program that converts the entire source code of a programming language into executable machine code for a CPU
  - Compiler takes a large amount of time to analyze the entire source code but the overall execution time of the program is significantly faster
  - Compiler generates the error message only after scanning the whole program. Hence, debugging is comparatively hard as the error can be present anywhere in the program
  - Compiler generates intermediate object code, which further requires linking; hence, needs more memory
  - Examples: C, C++, Java, Rust, Go

- Interpreters
  - An Interpreter takes a source program and runs it line by line, translating each line as it comes to it, one statement at a time
  - Interpreter takes less amount of time to analyze the source code, but the overall execution time of the program is slower
  - Interpreter's debugging is easier as it continues translating the program until the first error is met
  - No intermediate object code is generated; hence, interpreters are more memory efficient
  - Python, Perl, Ruby, PHP, JavaScript