CALIFORNIA STATE UNIVERSITY
FULLERTON

CPSC 335: Algorithm Engineering

Department of Computer Science, College of Engineering and Computer Science

Instructor: Dr. Shah, Adjunct Faculty, Computer Science https://www.hassanonline.us/

Lecture # 1 Notes: Introduction to Pseudocode

Pseudocode is a method used to describe algorithms using a combination of natural language and high-level programming constructs. It is not bound by the syntax rules of any specific programming language, making it a versatile tool for algorithm design and communication among programmers. The primary goal of pseudocode is to focus on the algorithm's logic without getting bogged down in syntactic details.

Characteristics of Pseudocode

1. **Readability**: Pseudocode should be easily understandable by anyone with basic programming knowledge.
2. **Simplicity**: It abstracts away complex code syntax, making algorithms easier to read and write.
3. **Flexibility**: There is no strict standard for pseudocode, allowing for variations that best suit the algorithm's explanation.

Writing Pseudocode

- Use clear and concise statements resembling high-level programming languages.
- Employ common programming constructs like **if-else**, **while**, **for**, but in plain English.
- Indicate blocks of code with indentations or keywords like **begin** and **end**.

Example 1: Finding the Maximum Value in a List

Algorithm FindMax

Input: A list of numbers L

Output: The maximum number in the list L

```
maxValue <- L[0]
for each number n in L starting from the second element
    if n > maxValue then
        maxValue <- n
end for
return maxValue
```

Example 2: Calculating the Factorial of a Number

Algorithm CalculateFactorial

Input: A positive integer N

Output: The factorial of N

```
factorial <- 1
for i from 1 to N
    factorial <- factorial * i
end for
return factorial
```

Example 3: Linear Search for an Element in a List

Algorithm LinearSearch
Input: A list of elements L, and a target element T
Output: The index of T in L, or -1 if T is not found

```
for index from 0 to length of L - 1
    if L[index] = T then
        return index
end for
return -1
```

Example 4: Bubble Sort Algorithm

Algorithm BubbleSort
Input: A list of numbers L
Output: The list L sorted in ascending order

```
for i from 0 to length of L - 1
    for j from 0 to length of L - i - 2
        if L[j] > L[j + 1] then
            swap L[j] and L[j + 1]
        end if
    end for
end for
return L
```

Example 5: Binary Search Algorithm (on a sorted list)

Algorithm BinarySearch

Input: A sorted list of elements L, and a target element T

Output: The index of T in L, or -1 if T is not found

```
left <- 0
right <- length of L - 1
while left <= right
    mid <- (left + right) / 2
    if L[mid] = T then
        return mid
    else if L[mid] < T then
        left <- mid + 1
    else
        right <- mid - 1
    end if
end while
return -1
```

Pseudocode is a powerful tool for algorithm design, allowing for clear communication of complex ideas without the constraints of programming language syntax. Through examples, we've seen how algorithms can be conceptualized in pseudocode and then translated into executable Python code. As you continue to learn and work with algorithms, developing strong skills in writing and interpreting pseudocode will be invaluable.
