COL702: Advanced Data Structures and Algorithms

Ragesh Jaiswal, IITD

Data Structures and Algorithms

Introduction

- Data Structure: Systematic way of organising and accessing data.
- Algorithm: A step-by-step procedure for performing some task.

Introduction

- How do we describe an algorithm?
    - Algorithms are platform independent and so should be their description.
    - This allows us to focus on the main ideas rather than spend time parsing the programming language-specific syntax and the implementation details.

Introduction

- How do we describe an algorithm?
  - Algorithms are platform independent and so should be their description.
  - This allows us to focus on the main ideas rather than spend time parsing the programming language-specific syntax and the implementation details.
  - A concise way of describing an algorithm is pseudocode.
    - Pseudocode is not an actual code.
    - It consists of:
      high-level programming constructs (if-then, for etc.) + natural language.

Introduction

- How do we describe an algorithm?
  - Algorithms are platform independent and so should be their description.
  - This allows us to focus on the main ideas rather than spend time parsing the programming language-specific syntax and the implementation details.
  - A concise way of describing an algorithm is pseudocode.
    - Pseudocode is not an actual code.
    - It consists of:
      high-level programming constructs (if-then, for etc.) + natural language.

Algorithm

FindMin(A, n)
- $min \leftarrow A[1]$
- **for** $i = 2$ to n
  - **if** $(A[i] < min)$
    - $min \leftarrow A[i]$
- **return**(min)

Introduction

- How do we describe an algorithm?
  - Algorithms are platform independent and so should be their description.
  - This allows us to focus on the main ideas rather than spend time parsing the programming language-specific syntax and the implementation details.
  - A concise way of describing an algorithm is pseudocode.
    - Pseudocode is not an actual code.
    - It consists of:
      high-level programming constructs (if-then, for etc.) + natural language.

Algorithm

`FindMin`(A, n)
- $min \leftarrow A[1]$
- **for** $i = 2$ to n
  - **if** $A[i]$ is smaller than min
    - $min \leftarrow A[i]$
- **return**(min)

Introduction

- How do we describe an algorithm?
  - Using a pseudocode.
- What are the desirable features of an algorithm?

Introduction

- How do we describe an algorithm?
  - Using a pseudocode.
- What are the desirable features of an algorithm?
  - It should be correct.
  - It should run fast.
  - It should take a small amount of space (RAM).
  - It should consume a small amount of power.
  - $\vdots$

Introduction

- How do we describe an algorithm?
  - Using a pseudocode.
- What are the desirable features of an algorithm?
  1. **It should be correct.**
  2. It should run fast.
- How do we argue that an algorithm is correct?

Introduction

- How do we argue that an algorithm is correct?
  - Proof of correctness: An argument that the algorithm works correctly for **all** inputs.
    - Proof: A valid argument that establishes the truth of a mathematical statement.
- Consider the following algorithm that is supposed to output the sum of elements of an integer array of size n.

Algorithm

FindSum(A, n)

- $sum \leftarrow 0$
- for $i = 1$ to n
  - $sum \leftarrow sum + A[i]$
- return(sum)

Introduction

- How do we argue that an algorithm is correct?
  - Proof of correctness: An argument that the algorithm works correctly for **all** inputs.
    - Proof: A valid argument that establishes the truth of a mathematical statement.
- Consider the following algorithm that is supposed to output the sum of elements of an integer array of size n.

Algorithm

FindSum(A, n)
- $sum \leftarrow 0$
- for $i = 1$ to n
  - $sum \leftarrow sum + A[i]$
- return(sum)

- To prove the algorithm correct, let us define the following loop-invariant:
  $P(i)$: At the end of the i^{th} iteration, the variable sum contains the sum of first i elements of the array A.

Introduction

- How do we argue that an algorithm is correct?
  - Proof of correctness: An argument that the algorithm works correctly for **all** inputs.
    - Proof: A valid argument that establishes the truth of a mathematical statement.
- Consider the following algorithm that is supposed to output the sum of elements of an integer array of size n.

Algorithm

FindSum(A, n)
- $sum \leftarrow 0$
- for $i = 1$ to n
  - $sum \leftarrow sum + A[i]$
- return(sum)

- To prove the algorithm correct, let us define the following loop-invariant:
  $P(i)$: At the end of the i^{th} iteration, the variable *sum* contains the sum of first i elements of the array A.
- How do we prove statements of the form $\forall i, P(i)$?

Introduction

- How do we argue that an algorithm is correct?
  - Proof of correctness: An argument that the algorithm works correctly for **all** inputs.
    - Proof: A valid argument that establishes the truth of a mathematical statement.
- Consider the following algorithm that is supposed to output the sum of elements of an integer array of size n.

Algorithm

FindSum(A, n)
- $sum \leftarrow 0$
- for $i = 1$ to n
  - $sum \leftarrow sum + A[i]$
- return(sum)

- To prove the algorithm correct, let us define the following loop-invariant:
  $P(i)$: At the end of the i^{th} iteration, the variable *sum* contains the sum of first i elements of the array A.
- How do we prove statements of the form $\forall i, P(i)$?Induction

Introduction

- Proof: A valid argument that establishes the truth of a mathematical statement.
  - The statements used in a proof can include axioms, definitions, the premises, if any, of the theorem, and previously proven theorems and uses rules of inference to draw conclusions.
- A proof technique very commonly used when proving the correctness of Algorithms is *Mathematical Induction*.

Definition (Strong Induction)

To prove that $P(n)$ is true for all positive integers, where $P(.)$ is a propositional function, we complete two steps:

- Basis step: We show that $P(1)$ is true.
- Inductive step: We show that for all k, if $P(1), P(2), ..., P(k)$ are true, then $P(k+1)$ is true.

Definition (Strong Induction)

To prove that $P(n)$ is true for all positive integers, where $P(.)$ is a propositional function, we complete two steps:

- Basis step: We show that $P(1)$ is true.
- Inductive step: We show that for all k, if $P(1), P(2), ..., P(k)$ are true, then $P(k+1)$ is true.

- Question: Show that for all $n > 0$, $1 + 3 + ... + (2n - 1) = n^2$.

Introduction

- Question: Show that for all $n > 0$, $1 + 3 + ... + (2n-1) = n^2$.

Proof

- Let $P(n)$ be the proposition that $1 + 3 + 5 + ... + (2n-1)$ equals n^2.
- Basis step: $P(1)$ is true since the summation consists of only a single term 1 and $1^2 = 1$.
- Inductive step: Assume that $P(1), P(2), ..., P(k)$ are true for any arbitrary integer k. Then we have:

$$\begin{aligned} 1 + 3 + ... + (2(k+1)-1) &= 1 + 3 + ... + (2k-1) + (2k+1) \\ &= k^2 + 2k + 1 \quad (\text{since } P(k) \text{ is true}) \\ &= (k+1)^2 \end{aligned}$$

This shows that $P(k+1)$ is true.
- Using the principle of Induction, we conclude that $P(n)$ is true for all $n > 0$. $\square$

- How do we describe an algorithm?
  - Using a pseudocode.
- What are the desirable features of an algorithm?
  1. It should be correct.
     - We use proof of correctness to argue correctness.
  2. **It should run fast.**

Introduction

- How do we describe an algorithm?
  - Using a pseudocode.
- What are the desirable features of an algorithm?
  1. It should be correct.
     - We use proof of correctness to argue correctness.
  2. **It should run fast.**
- Given two algorithms `A1` and `A2` for a problem, how do we decide which one runs faster?

Introduction

- Given two algorithms A1 and A2 for a problem, how do we decide which one runs faster?
  - Idea#1: Implement them on some platform, run and check.

Introduction

- Given two algorithms A1 and A2 for a problem, how do we decide which one runs faster?
  - Idea#1: Implement them on some platform, run and check.
  - The speed of programs P1 (implementation of A1) and P2 (implementation of A2) may depend on various factors:
    - Input
    - Hardware platform
    - Software platform
    - Quality of the underlying algorithm

End