

ACOL106: Data Structures and Algorithms

Ragesh Jaiswal, IITD

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.
 - We do an **asymptotic worst-case analysis** noting the running time in Big- (O, Ω, Θ) notation and use it to compare algorithms.

Problem

Given an integer array A with n elements output another array B such that for all i , $B[i] = \sum_{j=1}^i A[j]$. (That is find cumulative sum of elements in A .)

Algorithm

BetterCumulativeSum(A, n)	
- $sum \leftarrow 0$	$O(1)$
- for $i = 1$ to n	$O(n)$
- $sum \leftarrow sum + A[i]$	$O(n)$
- $B[i] \leftarrow sum$	$O(n)$
- return(B)	$O(1)$
Total: $O(n)$	

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.
 - We do an **asymptotic worst-case analysis** noting the running time in Big- (O, Ω, Θ) notation and use it to compare algorithms.

Problem

Sorting: Given an integer array A with n elements, sort it.

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.
 - We do an **asymptotic worst-case analysis** noting the running time in Big- (O, Ω, Θ) notation and use it to compare algorithms.

Problem

Sorting: Given an integer array A with n elements, sort it.

Algorithm

SelectionSort(A, n)

- for $i = 1$ to $n - 1$
 - $min \leftarrow \text{FindMin}(A, n, i)$
 - Swap(A, i, min)

FindMin(A, n, i)

- $min \leftarrow i$
- for $j = i + 1$ to n
 - if $(A[j] < A[min])$ $min \leftarrow j$
- return(min)

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.
 - We do an **asymptotic worst-case analysis** noting the running time in Big- (O, Ω, Θ) notation and use it to compare algorithms.

Problem

Sorting: Given an integer array A with n elements, sort it.

Algorithm

SelectionSort(A, n)

- for $i = 1$ to $n - 1$
- $min \leftarrow \text{FindMin}(A, n, i)$
- Swap(A, i, min)

FindMin(A, n, i)

- $min \leftarrow i$
- for $j = i + 1$ to n
- if ($A[j] < A[min]$) $min \leftarrow j$
- return(min)

- What is an appropriate loop-invariant for the above algorithm?

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.
 - **We do an asymptotic worst-case analysis** noting the running time in Big- (O, Ω, Θ) notation and use it to compare algorithms.

Problem

Sorting: Given an integer array A with n elements, sort it.

Algorithm

SelectionSort(A, n)

- for $i = 1$ to $n - 1$
- $min \leftarrow \text{FindMin}(A, n, i)$
- Swap(A, i, min)

FindMin(A, n, i)

- $min \leftarrow i$
- for $j = i + 1$ to n
- if ($A[j] < A[min]$) $min \leftarrow j$
- return(min)

- What is running time of the above algorithm?

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.
 - We do an **asymptotic worst-case analysis** noting the running time in Big- (O, Ω, Θ) notation and use it to compare algorithms.

Problem

Sorting: Given an integer array A with n elements, sort it.

Algorithm

BubbleSort(A, n)

- for $i = 1$ to $(n - 1)$
 - for $j = 1$ to $(n - i)$
 - if($A[j] > A[j + 1]$) Swap($A, j, j + 1$)

- What is an appropriate loop-invariant for the above algorithm?

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.
 - We do an **asymptotic worst-case analysis** noting the running time in $\text{Big-}(O, \Omega, \Theta)$ notation and use it to compare algorithms.

Problem

Sorting: Given an integer array A with n elements, sort it.

Algorithm

BubbleSort(A, n)

- for $i = 1$ to $(n - 1)$
 - for $j = 1$ to $(n - i)$
 - if($A[j] > A[j + 1]$) Swap($A, j, j + 1$)

- What is running time of the above algorithm?

End