COL106: Data Structures and Algorithms

Ragesh Jaiswal, IITD

Administrative Slide

- URGENT: Register on gradescope.
  - Use course code 9Z547M to add COL106.
  - Use your IIT Delhi email address.
  - Do this before the lecture tomorrow (Fri).
- Quiz 1 and 2 in the lecture tomorrow (Fri).

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?
  - 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.

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.)

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

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

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

CumulativeSum(A, n)	
- for $i = 1$ to n	$3n$ operations
- $sum \leftarrow 0$	n operations
- for $j = 1$ to i	$3 \cdot (1 + 2 + 3 + \ldots + n)$ operations
- $sum \leftarrow sum + A[j]$	$2 \cdot (1 + 2 + 3 + \ldots + n)$ operations
- $B[i] \leftarrow sum$	n operations
- return(B)	1 operation (assuming that only reference to the array is returned)
	Total: $\frac{1}{2} \cdot (5n^2 + 15n + 2)$ operations

- 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

CumulativeSum(A, n)	
- for $i = 1$ to n	$2n$ operations
- $sum \leftarrow 0$	n operations
- for $j = 1$ to i	$2 \cdot (1 + 2 + 3 + \ldots + n)$ operations
- $sum \leftarrow sum + A[j]$	$2 \cdot (1 + 2 + 3 + \ldots + n)$ operations
- $B[i] \leftarrow sum$	n operations
- return(B)	1 operation (assuming that only reference to the array is returned)
	Total: $\frac{1}{2} \cdot (5n^2 + 15n + 2)$ operations

- So, the asymptotic worst-case running time of the above algorithm is $O(n^2)$. Note that we can also say the running time is $\Omega(n^2)$ and $\Theta(n^2)$.

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

CumulativeSum(A, n)	
- for $i = 1$ to n	$2n$ operations
- $sum \leftarrow 0$	n operations
- for $j = 1$ to i	$2 \cdot (1 + 2 + 3 + \ldots + n)$ operations
- $sum \leftarrow sum + A[j]$	$2 \cdot (1 + 2 + 3 + \ldots + n)$ operations
- $B[i] \leftarrow sum$	n operations
- return(B)	1 operation (assuming that only reference to the array is returned)
	Total: $\frac{1}{2} \cdot (5n^2 + 15n + 2)$ operations

- So, the asymptotic worst-case running time of the above algorithm is $O(n^2)$. Note that we can also say the running time is $\Omega(n^2)$ and $\Theta(n^2)$.
- Can you design a better $O(n)$ (linear-time) algorithm for this problem?

- 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 ← FindMin(A, n, i)
    - Swap(A, i, min)
```

```
FindMin(A, n, i)
  min ← i
  - for j = i + 1 to n
    - if (A[j] < A[min]) min ← 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 ← FindMin(A, n, i)
      - Swap(A, i, min)
```

```
FindMin(A, n, i)
  min ← i
  - for j = i + 1 to n
      - if (A[j] < A[min]) min ← 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 ← FindMin(A, n, i)
      - Swap(A, i, min)
```

```
FindMin(A, n, i)
  min ← i
  - for j = i + 1 to n
      - if (A[j] < A[min]) min ← 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 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 running time of the above algorithm?

- How do Data Structures play a part in making computational tasks efficient?

Introduction

- How do Data Structures play a part in making computational tasks efficient?

Example problem

Maintain a record of students and their scores on some test so that queries of the following nature may be answered:

- Insert: Insert a new record of a student and his/her score.
- Search: Find the score of a given student.

- Suppose we maintain the information in a 2-dimensional array.
  - How much time does each insert operations take?
  - How much time does each search operation take?

Introduction

- How do Data Structures play a part in making computational tasks efficient?

Example problem

Maintain a record of students and their scores on some test so that queries of the following nature may be answered:

- Insert: Insert a new record of a student and his/her score.
- Search: Find the score of a given student.

- Suppose we maintain the information in a 2-dimensional array.
  - How much time does each insert operations take? $O(1)$
  - How much time does each search operation take? $O(n)$
  - So, if the majority of the operations performed are search operations, then this data structure is perhaps not the right one.

- How do Data Structures play a part in making computational tasks efficient?

Example problem

Maintain a record of students and their scores on some test so that queries of the following nature may be answered:

- Insert: Insert a new record of a student and his/her score.
- Search: Find the score of a given student.

- Suppose we maintain the information in a 2-dimensional array such that the array is sorted based on the names (dictionary order).
  - How much time does each insert operations take?
  - How much time does each search operation take?

Introduction

- How do Data Structures play a part in making computational tasks efficient?

Example problem

Maintain a record of students and their scores on some test so that queries of the following nature may be answered:

- Insert: Insert a new record of a student and his/her score.
- Search: Find the score of a given student.

- Suppose we maintain the information in a 2-dimensional array such that the array is sorted based on the names (dictionary order).
  - How much time does each insert operations take? $O(n)$
  - How much time does each search operation take? $O(\log n)$ using **Binary Search**
  - In this case, if the majority of the operations performed are insert operations, then the previous one is better.

End