Algorithms (CS 2443)

Fahad Panolan

Department of Computer Science and Engineering
Indian Institute of Technology Hyderabad, India

5 January 2022

Course Information

- 3 credits, C-slot
- Instructors: Fahad Panolan and Nitin Saurabh

Course Information

- 3 credits, C-slot
- Instructors: Fahad Panolan and Nitin Saurabh
- Piazza page for communication:
  `piazza.com/iit_hyderabad/fall2021/cs24432022`
  (Access code: cs2443iith)

Course Information

- 3 credits, C-slot
- Instructors: Fahad Panolan and Nitin Saurabh
- Piazza page for communication: piazza.com/iit_hyderabad/fall2021/cs24432022 (Access code: cs2443iith)
- Reference: **Algorithms** (First Edition, Jun 2019) by Jeff Erickson

Course Information

- 3 credits, C-slot
- Instructors: Fahad Panolan and Nitin Saurabh
- Piazza page for communication: piazza.com/iit_hyderabad/fall2021/cs24432022 (Access code: cs2443iith)
- Reference: **Algorithms** (First Edition, Jun 2019) by Jeff Erickson
- Grading:
  - Exams: 15, 15, 40
  - Class participation: 10
  - Quizzes and assignments: 20

Introduction

- Designing algorithms
- Analyse algorithms: Formally prove that it is correct. Formally prove its efficiency.
- Various algorithmic techniques will be covered.

Examples of algorithmic question

- Schedule time table for IITH
- Given a number n, check whether it is a prime number
- Given a road network and two cities, find a shortest path between them
- Mutifly two matrices, find rank of a matrix etc
- Find two strings P and Q, check whether P is a substring (subsequence) of Q.

Sorting

Input: A sequence of number $(a_1, \ldots, a_n)$
Output: A permutation $(a'_1, \ldots, a'_n)$ of those numbers where $a'_1 \leq a'_2 \leq \ldots \leq a'_n$

Insertion Sort

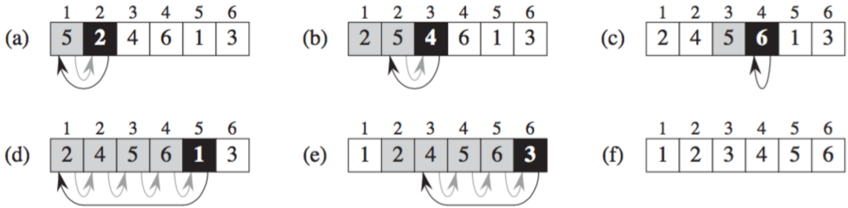

Figure from "Introduction to Algorithms" by CLRS

Insertion Sort : Pseudocode

INSERTION-SORT(A)

```
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1 .. j - 1].
    i = j - 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i - 1
    A[i + 1] = key
```

Correctness Proof

- Use mathematical induction

Correctness Proof

- Use mathematical induction

Loop invariant: for loop

At the start of each iteration of the for loop, the subarray $A[1 \ldots j-1]$ consists of the elements originally in $A[1 \ldots j-1]$, but in non-decreasing order.

- To prove the above statement, we need to write a loop invariant for the **while** loop and prove it.

Run Time Analysis (Time Complexity)

- The number of instructions executed by the algorithms. This may be different for different inputs.
- Each execution of each instruction takes **one unit** of time. (We assume RAM model)

Input 1: $[1, 2, \ldots, n]$

INSERTION-SORT(A)

	times
for $j = 2$ **to** $A.length$	$n-1$
$key = A[j]$	$n-1$
// Insert $A[j]$ into the sorted sequence $A[1\,..\,j-1]$.	0
$i = j - 1$	$n-1$
while $i > 0$ and $A[i] > key$	$n-1$
$A[i+1] = A[i]$	0
$i = i - 1$	0
$A[i+1] = key$	$n-1$

$$R([1, 2, \ldots n]) = 5(n-1)$$

Input 2: $[n, n-1, \ldots, 1]$

INSERTION-SORT(A)

```
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted sequence A[1 .. j - 1].
    i = j - 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i - 1
    A[i + 1] = key
```

Times.

$n-1$

$n-1$

0

$n-1$

$\sum_{j=2}^{n} j$

$\sum_{j=2}^{n} j-1$

$\sum_{j=2}^{n} j-1$

$n-1$

$$4(n-1)+3\sum_{j=2}^{n} j-\sum_{j=2}^{n} 2$$

$$= 4(n-1)+\frac{3(n+1)(n-2)}{2}-2(n-2)$$

$$R([n, n-1, \ldots 1]) = \frac{3}{2}n^2-\frac{5}{2}n+3$$

Worst-case and best-case running time

- Worst-case running time of INSERTION-SORT is the function $T\colon \mathbb{N} \mapsto \mathbb{N}$:

$$T(n) = \max_{A\colon |A|=n} R(A).$$

Worst-case and best-case running time

- Worst-case running time of INSERTION-SORT is the function $T\colon \mathbb{N} \mapsto \mathbb{N}$:
$$T(n) = \max_{A\colon |A|=n} R(A).$$
That is, for any input B of "length n", INSERTION-SORT on B executes at most $T(n)$ instructions.

Worst-case and best-case running time

- Worst-case running time of Insertion-Sort is the function $T\colon \mathbb{N} \mapsto \mathbb{N}$:

  $$T(n) = \max_{A\colon |A|=n} R(A).$$

  That is, for any input B of "length n", Insertion-Sort on B executes at most $T(n)$ instructions.

- Best-case running time of Insertion-Sort is the function $T_b\colon \mathbb{N} \mapsto \mathbb{N}$:

  $$T_b(n) = \min_{A\colon |A|=n} R(A).$$

- $T(n) = \frac{3}{2}n^2 - \frac{5}{2}n + 3$
- $T_b(n) = 5n - 5$

Comparing two algorithms

- Suppose we have another algorithm (say $\mathcal{A}$) for SORTING with worst-case running time is $1000n \log n + 100n$.
- Which algorithm is best among these two?

Thank You.