Introduction to Algorithms

Subhabrata Samajder

IIIT, Delhi
Winter Semester,
1st March, 2023

About Myself

Name: Subhabrata Samajder

Research Interests:

- Lattice based cryptography
- Statistical aspects of symmetric key cryptanalysis
- Searchable Symmetric Encryption (SSE)
- e-Voting
- Broadcast Encryption
- Blockchain
- Random graphs

About Myself

Name: Subhabrata Samajder

E-mail: *subhabrata@iiitd.ac.in*

Office: B-505 (R & D Block)

Office Hours: By appointment

Course Webpage: On Google classroom

- Class Code: l4o5nrw

Academic Integrity Policy

- Anyone caught cheating or copying will be penalised.
- Plagiarism cases will be *dealt strictly*.
- Take this opportunity to stay away from plagiarism forever.

Grading plan: Tentative Grading Components

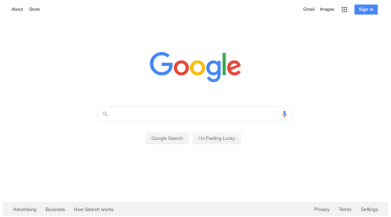
Components	Number	Weightage
MidSem Theory	1	30%
EndSem Theory	1	35%
Weekly Coding Assignments	≥ 6	20%
Quiz and/or Homework	≥ 4	15%
Bonus (Lab attendance + Lab Tests)	NA	5%

Introduction to Algorithms

Algorithms: In Our Daily Lives

Cooking

Traffic Lights



Google Search

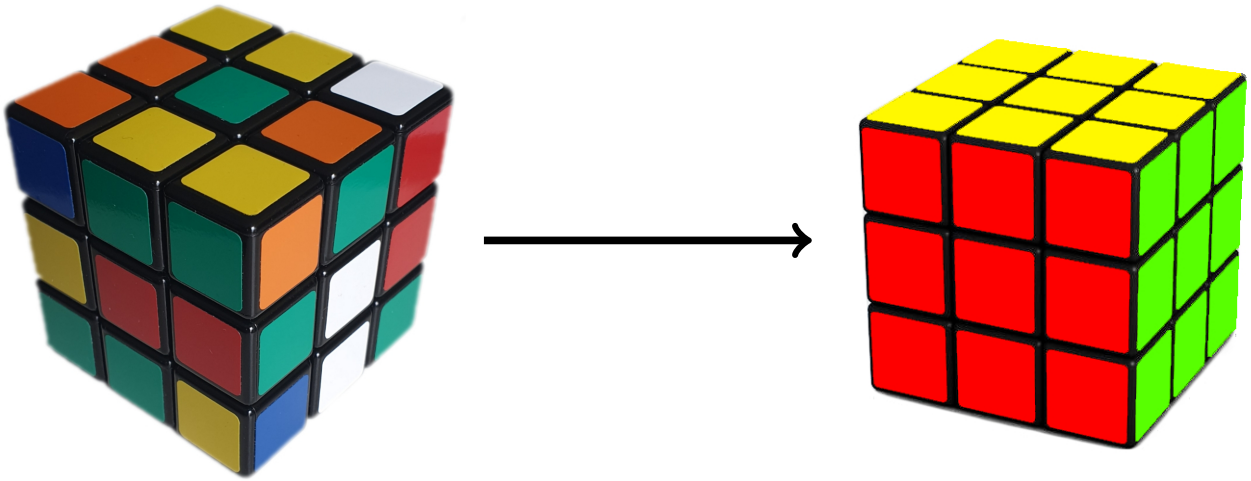
Sorting Vinyl Records

Work Commute

Online Shopping

Algorithms: Rubik's Cube

Algorithms: Rubik's Cube



Introduction of Algorithms

"What is an algorithm?"

Introduction of Algorithms

"What is an algorithm?"

Intuitive answer: It is a finite sequence of elementary operations with the objective of performing some (computational) task.

Introduction of Algorithms

"What is an algorithm?"

Intuitive answer: It is a finite sequence of elementary operations with the objective of performing some (computational) task.

Elementary Operations

"How elementary is 'elementary'?"

Elementary Operations

"How elementary is 'elementary'?"

The elementary operations that we will consider will be at a higher level and include arithmetic and logical operations.

Finiteness

Algorithms: Finiteness $\Rightarrow$ an algorithm must stop.

- **Example:** Compute $(a + b) * (c + d)$

$$t_1 = a + b; \quad t_2 = c + d; \quad t_3 = t_1 * t_2.$$

Finiteness

Algorithms: Finiteness $\Rightarrow$ an algorithm must stop.

- **Example:** Compute $(a+b)*(c+d)$

$$t_1 = a+b; \quad t_2 = c+d; \quad t_3 = t_1 * t_2.$$

Computational Method: A procedure that has all of the characteristics of an algorithm except that it possibly *lacks finiteness*.

- **Example:** $\text{while}(1)\{\}$

Sequence

$$t_1 = a + b; \quad t_2 = c + d; \quad t_3 = t_1 * t_2.$$

- We emphasise on the sequential nature of the procedure.

- Any permutation of this sequence *does not* give the same desired output.
  - **Example:** $t_3 = t_1 * t_2; \quad t_2 = c + d; \quad t_1 = a + b$, is not the same as the algorithm above.

Finite Sequence

$$t_1 = a + b; \quad t_2 = c + d; \quad t_3 = t_1 * t_2.$$

Note: Sometimes different orderings of the operations may give rise to the *same* result.

- **Example:**

$$\begin{aligned} t_1 &= a + b; \quad t_2 = c + d; \quad t_3 = t_1 * t_2; \text{ and} \\ t_2 &= c + d; \quad t_1 = a + b; \quad t_3 = t_1 * t_2. \end{aligned}$$

Finite Sequence

$$t_1 = a + b; \quad t_2 = c + d; \quad t_3 = t_1 * t_2.$$

Note: Sometimes different orderings of the operations may give rise to the *same* result.

- **Example:**

$$\begin{aligned} t_1 &= a + b; \quad t_2 = c + d; \quad t_3 = t_1 * t_2; \text{ and} \\ t_2 &= c + d; \quad t_1 = a + b; \quad t_3 = t_1 * t_2. \end{aligned}$$

- **Note:** $t_1 = a + b$ and $t_2 = c + d$ can be executed independently of each other.
- **Single computing unit (a processor):** *Sequential execution*.
- **Two computing unit:** Can be executed simultaneously!

Finite Sequence

$$t_1 = a + b; \quad t_2 = c + d; \quad t_3 = t_1 * t_2.$$

Note: Sometimes different orderings of the operations may give rise to the *same* result.

- **Example:**

$$\begin{array}{llllllll} t_1 & = & a+b; & t_2 & = & c+d; & t_3 & = t_1 * t_2; \text{ and} \\ t_2 & = & c+d; & t_1 & = & a+b; & t_3 & = t_1 * t_2. \end{array}$$

- **Note:** $t_1 = a + b$ and $t_2 = c + d$ can be executed independently of each other.
- **Single computing unit (a processor):** *Sequential execution*.
- **Two computing unit:** Can be executed simultaneously! Give rise to the area of *parallel algorithms*.

Finite Sequence

$$t_1 = a + b; \quad t_2 = c + d; \quad t_3 = t_1 * t_2.$$

Note: Sometimes different orderings of the operations may give rise to the *same* result.

- **Example:**

$$\begin{array}{llllllll} t_1 & = & a+b; & t_2 & = & c+d; & t_3 & = t_1 * t_2; \text{ and} \\ t_2 & = & c+d; & t_1 & = & a+b; & t_3 & = t_1 * t_2. \end{array}$$

- **Note:** $t_1 = a + b$ and $t_2 = c + d$ can be executed independently of each other.
- **Single computing unit (a processor):** *Sequential execution*.
- **Two computing unit:** Can be executed simultaneously! Give rise to the area of *parallel algorithms*.

We would only be concentrating on sequential algorithms!!

Inputs and Output

Recall: The purpose of an algorithm is to perform some task.

Inputs and Output

Recall: The purpose of an algorithm is to perform some task.

Inputs: Can take *several* inputs.

- *Example:* a, b, c, d.

Output: Algorithms produce *an* output.

- *Example:* $(a + b) * (c + d)$.

Inputs and Output

Recall: The purpose of an algorithm is to perform some task.

Inputs: Can take *several* inputs.

- *Example:* a, b, c, d.

Output: Algorithms produce *an* output.

- *Example:* $(a + b) * (c + d)$.
- The relation of the output to the input defines the *computational task* of the algorithm.

Inputs and Output

Recall: The purpose of an algorithm is to perform some task.

Inputs: Can take *several* inputs.

- *Example:* a, b, c, d.

Output: Algorithms produce *an* output.

- *Example:* $(a + b) * (c + d)$.
- The relation of the output to the input defines the *computational task* of the algorithm.
- **Simplest case:** Binary valued output (*Decision problem*).

Inputs and Output

- **Simplest case:** Binary valued output (*Decision problem*).

  Example: Searching Problem
  - **I/P:** A list L of integer values and another value v.
  - **Question:** Does $v \in L$?
  - **O/P:** '*yes*' if $v \in L$; else it returns '*no*'.

Note: *Decision problems appear rather simple but much of the sophistication of the area of algorithms can be discovered by studying such algorithms*!!

Resources of an Algorithm

'Efficient algorithms?

- **High level view:** Efficiency $\equiv$ requiring little 'resources'.

Resources of an Algorithm

'Efficient algorithms?

- **High level view:** Efficiency $\equiv$ requiring little 'resources'.

- Here, resources $\equiv$ the time of execution and the space required by the algorithm.

Resources of an Algorithm

- **Time:** # steps required by the algorithm to produce its output.
  - **Assumption:** Each elementary operation requires *unit time*.
  - $\therefore$ # steps = time required by the algorithm.

Resources of an Algorithm

- **Time:** # steps required by the algorithm to produce its output.
  - **Assumption:** Each elementary operation requires *unit time*.
  - $\therefore$ # steps = time required by the algorithm.
- **Space:** # *temporary* variables.

Resources of an Algorithm

- **Time:** # steps required by the algorithm to produce its output.
  - **Assumption:** Each elementary operation requires *unit time*.
  - $\therefore$ # steps = time required by the algorithm.

- **Space:** # *temporary* variables.

  Example: Our algorithm for finding $(a + b) * (c + d)$.
  - Temporary variables = t_1, t_2 and t_3.
  - $\therefore$ space required is 3.

Resources of an Algorithm

- **Time:** # steps required by the algorithm to produce its output.
  - **Assumption:** Each elementary operation requires *unit time*.
  - $\therefore$ # steps = time required by the algorithm.
- **Space:** # *temporary* variables.

  Example: Our algorithm for finding $(a + b) * (c + d)$.
  - Temporary variables = t_1, t_2 and t_3.
  - $\therefore$ space required is 3.
- **Other resources:** For example, power consumption is important for battery operated devices.

Books Consulted

1. Chapter 2 of *A Course on Cooperative Game Theory* by Satya R. Chakravarty, Palash Sarkar and Manipushpak Mitra.

2. *Introduction to Algorithms: A Creative Approach* by Udi Manber.

Thank You for your kind attention!