INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

CVIT

# Data Structures & Algorithms for Problem Solving (CS1.304)

## Lecture # 01

Vineet Gandhi & Avinash Sharma
Center for Visual Information Technology (CVIT),
IIIT Hyderabad

# Welcome to IIIT Hyderabad

# Why DSAPS?

- Problem solving is essential to Computer Science.
- Think of some problems that we solve almost on a daily basis in digital world.
    - Go through contacts/messages that are already sorted
    - Find routes in a city from a point A to a point B
    - Find the best ticket given a set of constraints.
    - Find appropriate search keywords based on various criteria.

# Why DSAPS?

# Why DSAPS?

- Central to the examples mentioned are **data structures & Algorithms** that increase the efficiency by at least an order of magnitude.
- As the size of the data gets larger, the importance of these data structures gets critical.
  - Consider the amount of data handled by Google Maps, or the Human Genome project, or the Collider project.
- The aims of this course
  - Understand these problems and the data structures/algorithms deployed.
  - Develop solutions using appropriate data structures.

# About the course

- We will start by covering several fundamental data structures including
  - Stacks/linked-list/queues/lists
- Problem Solving and related Data Structures e.g., :
  - Search trees – especially AVL Trees and B-trees
  - Dynamic arrays and amortized data structures
  - String based problems and tries, suffix trees
  - Range querying via range trees
  - Randomization in computing via perfect hashing

# About the course

- Will introduce practical motivations to each of the considered topics.
- Several problem solving sessions to fully understand the implications of using a data structure for problem solving.
- Emphasis also on correctness and efficiency.
- Elementary analysis
  - Online Laboratory sessions are therefore very important.

# About the course - Material

- No prescribed textbook
- But you can refer to
  - "Data structures and algorithm analysis in C" by Mark Allen Weiss
  - "Introduction to Algorithms" by Thomas H. Cormen

# About the course - Structure

- Weekly 2 lecture hours.
- At least 1 Laboratory sessions every week
  - about 4-5 problems to be solved in the session (TAs to assist).
  - Credits are given for lab performance
- Several homework assignments
  - About 3-5, one every two weeks.
  - Each set to have about 6-7 problems

- Actively use Courses (Moodle) as platform for course content/news/assignments
- Email communication
  - Vineet Gandhi, vgandhi@iiit.ac.in
  - Avinash Sharma, asharma@iiit.ac.in
- Very important: Seek help early enough.
- Strictly, no plagiarism Any detected case of plagiarism to be taken seriously.

# About the course – Grading Policy

- Assessment (Credit Distribution**)
  - Quiz (21%) + Exam (30%)
  - Assignments* (24%)
  - Lab Test (25%)

* If copying is detected, you will get 0 marks for the assignment

**This policy might slightly vary as course evolve over the semester

# Organization (today's lecture)

1. Number Representation

**UNDERSTAND BASICS**

2. Number Operations

**UNDERSTAND BASICS**

3. Finding Greatest Common Divisor (GCD)

**HOW TO FORMULATE ?**

# Number System

- A number system is a way to represent numbers.
- Several known number systems in practice today.
  - Hindu/Decimal/Arabic system
  - Roman system
  - Binary, octal, hexa-decimal.
  - Unary system
  - ...

Classifiable as

- positional
- non-positional

# Number System

- Hindu/Decimal system
  - Numbers represented using the digits in {0, 1, ,..., 9}. E.g., 8,732,937,309
- Roman System
  - Numbers represented using the letters I, V, X, L, C, D, and M. E.g., X represents 10, L represents 50.
  - LX stands for 60, IV stands for 4, what is MMXIX?

|  | I | V | X | L | C | D | M |
|---|---|---|---|---|---|---|---|
| Value | 1 | 5 | 10 | 50 | 100 | 500 | 1000 |

  - MMMDCCCLLLXXXVVIX – largest numbers without any overlines/subtractions. What is this number?
- Binary system
  - Numbers represented using the digits 0 and 1.
  - 10111 represents 23.

# Number System

- Positional (aka value based) number systems associate a value to a digit based on the its position.
  - Example: Decimal, binary, ...
- Non-positional do not have such an association.
  - Example: Unary

# Operation on Numbers

- Let us consider operations addition and multiplication.
- Hindu/Decimal system
  - Add digit wise
  - Carry of $x$ from digit at position k to position k+1 equivalent to a value of $x.10^{k+1}$, $k \geq 0$.
  - Example: Adding 87 to 56 gives 143.
- Unary system
  - Probably, the first thing we learn.
  - To add two numbers $x$ and $y$, create a number that contains the number of 1's in both $x$ and $y$.
  - Example: Adding 1111 to 11111 results in 111111111.

# Operation on Numbers

- Roman system
  - A bit complicated but possible.
  - Follow the following three steps:
    - Write the numbers side by side.
    - Arrange the letters in decreasing order of value.
    - Simplify.
  - Example: to add 32 and 67:
    - 32 = XXXII, 67 = LXVII.
    - XXXIILXVII
    - LXXXXVIIII – LXLIX – XCIX
    - Simplified as: XCIX

| | | | | | |
|---|---|---|---|---|---|
| 1 | I | 11 | XI | 50 | L |
| 2 | II | 12 | XII | 100 | C |
| 3 | III | 13 | XIII | 500 | D |
| 4 | IV | 14 | XIV | 1000 | M |
| 5 | V | 15 | XV | | |
| 6 | VI | 16 | XVI | | |
| 7 | VII | 17 | XVII | | |
| 8 | VIII | 18 | XVIII | | |
| 9 | IX | 19 | XIX | | |
| 10 | X | 20 | XX | | |

- Rules such as:
  - If there are 4I's, write it as IV.
  - If there are 4X's, write it as XL.
  - Write LXL as XC.
  - Similar rules apply.
- Careful when starting with numbers such as LXIV.
  - Can replace IV with IIII initially.

# Operation on Numbers

- Let us now consider multiplication.
- Typically, multiplication is achieved by repeated addition.
- Decimal system
  - Known approach.
- Roman system
  - How to multiply?
  - Much complicated, but is possible.

# Multiplication in Roman Numbers

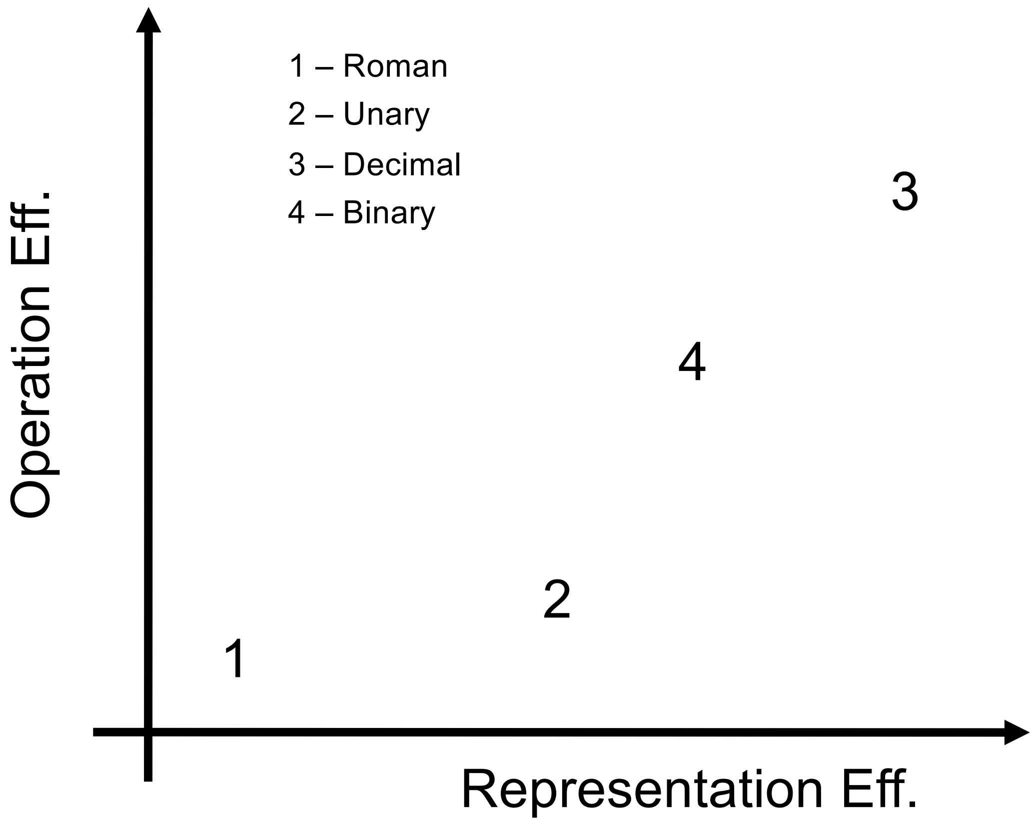
- Easy to imagine the following approach.
  - Multiplication is repeated addition
- Plus, think of a Roman number as the addition of 1000's + 100's + 50's + 10's + 5's + 1's.
- Multiply by each of these, and add as earlier.
- Example: LXII x XXXVII (62 x 37)
  - Multiply each of LXII by II. Meaning, make 2 copies of each symbol in LXII as LLXXIIII
  - Simplify using the rules of addition to CXXIV.
  - Now, multiply each of LXII by V. Start with LLLLLXXXXXIIIIIIIIII, simplify to CCCX.
  - Multiply LXII by XXX. That can be done in two ways. Either multiply by 3 followed by 10, or directly.

# Multiplication in Roman Numbers

- Continuing the example,
  - Let us multiply by LXII by 3 as LLLXXXIIIIII and simplified as CLXXXVI.
  - Now, multiply CLXXXVI by 10 as CCCCCCCCCCLLLLLLLLLLXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXVVVVVVVVVVIIIIIIIIII and simplified as MDCCCLX.
  - Add all the constituents as CXXIV + CCCX + MDCCCLX =CDXXXIV + MDCCCLX = MMCCXCIV.
  - What is this number ?

# Lesson Learnt

- Representation scheme for numbers influences the ease of performing operations.
- Roman system quite difficult to use.
- There are other such systems not in use today.



# Further Operations

- Let us now fix the decimal system as the representation scheme.
- We will now focus on the efficiency of operations.
- Let us see further operations such as finding the GCD of two numbers.

# Greatest Common Divisor (GCD)

- Given two positive numbers, x and y, the largest number that divides both x and y is called the greatest common divisor of x and y. Denoted gcd(x,y).
- Several approaches exist to find the gcd.
- Approach 1 : List all the divisors of both x and y. Find the common divisors, and the largest among the common divisors.
- Example for Approach 1: x = 24, y = 42,
  - divisors of 24 are {1, 2, 3, 4, 6, 8, 12, 24}.
  - divisors of 42 are {1, 2, 3, 6, 7, 14, 21, 42}.
  - Common divisors are {1, 2, 3, 6}. Hence, gcd(24, 42) = 6.

# Are There Other Representation Formats?

- Yes, recall the fundamental theorem of arithmetic.
- Any number can be expressed uniquely as a product of primes.
- So, a product of primes representation is also possible.
- Not easy to add though.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
| 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 |
| 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 |
| 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |
| 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |
| 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 |

# GCD – Approach II

- Use the fundamental theorem of arithmetic, write x and y as:
  - $x = p_1^{a1} . p_2^{a2} \dots . p_k^{ak}$ , $y = p_1^{b1} . p_2^{b2} \dots . p_r^{br}$
  - It holds that $gcd(x,y) = p_1^{\min\{a1,b1\}} . p_2^{\min\{a2,b2\}} \dots p_r^{\min\{ar,br\}}$.
- Example Approach II, let x = 24, y = 42.
  - $x = 2^3.3$ , $y = 2.3.7$
  - $gcd(x,y) = 2.3 = 6$.

# Which approach is better?

- Both are actually bad from a computational point of view.
- Both require a number to be factorized.
  - a computationally difficult task.
- For fairly large numbers, both approaches require a lot of computation.
  - Try for yourself by writing a simple program. Check what is the number at which things start getting real slow.
- Is there a better approach?
  - Indeed there is, given by the Greek mathematician Euclid.
  - Celebrated as a breakthrough.

# Euclid's algorithm for GCD

- Based on the following lemma.
- Lemma : Let x, y be two positive integers. Let q and r be integers such that x = y.q + r. Then, gcd(x,y) = gcd(y, r).
  - Argue that the common divisors of x and y are also common divisors of y and r.
  - Let d divide both x and y. Then, d divides x – yq = r.
  - The converse also applies in a similar fashion.
- The above lemma suggests the following algorithms for gcd.
  - Apply the above lemma repeatedly till the remainder is 0.
  - Let $r_1$, $r_2$, ..., be the remainders.

# Euclid's algorithm for GCD

- Let $r_2$, $r_3$, ..., be the remainders with $r_0 = x$ and $r_1 = y$.

- We have that:

$$r_0 = r_1 q_1 + r_2$$
$$r_1 = r_2 q_2 + r_3$$
$$r_2 = r_3 q_3 + r_4$$

and so on, till $r_{n-1} = r_n\, q_n + 0$

- By the result of the above lemma, it also holds that:

$$\begin{aligned} \gcd(r_0, r_1) &= \gcd(r_1, r_2) \\ &= \gcd(r_2, r_3) \\ &= \dots \\ &= \gcd(r_{n-1}, r_n) \\ &= \gcd(r_n, 0) = r_n \end{aligned}$$

- Notice that $r_n$ is the last nonzero remainder in the process.

# Euclid's algorithm for GCD

```
Algorithm GCD-Euclid(a,b)
    x := a, y := b;
    while (y is not  0)
        r := x mod y;x := y;y := r;
    end-while
    Return y;
End-Algorithm.
```

- Example, x = 42 and y = 24.
- Iteration 1: r = 18, x = 24; y = 18
- Iteration 2: r = 6, x = 18, y = 6
- Iteration 3: r = 0.

# Euclid's algorithm for GCD

- Why is this efficient?
- It can be shown that given numbers x and y, the algorithm requires only about log min{x,y} iterations.
  - Compared to about sqrt{x} for Approach I.
  - Why does approach 1 takes sqrt{x} iterations?
- There is indeed a difference for large numbers.
- The example suggests that also efficient ways to perform operations are of interest.

# Tentative Course Plan

| # | Date | Topic | Assignments |
|---|---|---|---|
| 01 | 16/08 | Introduction | |
| 02 | 23/08 | Linked List | |
| 03 | 26/08 | Stack and Queue | |
| 04 | 30/08 | Recursion | |
| 05 | 02/09 | Intro to Trees | Assign #1 |
| | | Quiz 1 | |
| 06 | 09/09 | Binary Search Trees | |
| 07 | 13/09 | AVL Trees | |
| 08 | 16/09 | R&B Trees, Splay Trees | Assign #2 |
| 09 | 20/09 | B+ Trees, Heaps | |
| 10 | 23/09 | Hashing | |
| 11 | 27/09 | Searching in Higher Dimensions | |
| 12 | 30/09 | Range Trees | |
| | | NO CLASS WEEK | |
| 13 | 07/10 | Trie | Assign #3 |
| 14 | 11/10 | Suffix Tree | |
| 15 | 14/10 | Graph Traversal (BFS) | |
| 16 | 18/10 | Shortest Path on Graphs | |
| 17 | 21/10 | Graph Traversal (DFS) | Assign #4 |
| 18 | 25/10 | Minimum Spanning Trees | |
| | | Quiz 2 | |
| 19 | 01/11 | Searching in Integer Data | |
| 20 | 08/11 | Van Emde Boas tree | |
| 21 | 11/11 | Amortized Analysis | |
| 22 | 15/11 | Sorting | |
| 23 | 18/11 | Algorithm Design | |
| 24 | 22/11 | Buffer Class | |

Thank You !