Design and Analysis of Algorithms

Lecture - 1

Success is always inevitable with Hard Work and Perseverance

N. Ravitha Rajalakshmi

Course Objective / Outcome

- Improve the programming skills by introducing the techniques / means to derive solutions for complex problems in an easier way.

Outcomes:

1. Analyze the complexity of algorithm
2. Design optimized algorithms for well defined problems
3. Improvise the algorithm by incorporating best practices

Assessment

- Every Week 2 programming assignments will be hosted in the hacker rank. Problem statements will be provided earlier that week.
  - Challenge: Solve all the test cases
  - Test cases will be hidden
- Quiz / program debugging / coding problems will be conducted regularly
  - Formative assessment
- One project (Assignment Presentation) will be provided at the end of the course.

Classroom Code

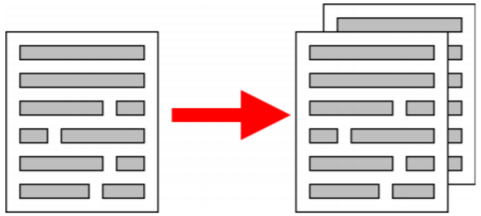
s4ajjgi

Learning Objective

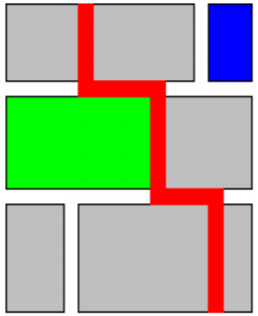
- Understand what type of problems will be discussed in the course.
- Understand efficient algorithms
- Learn mechanisms to effectively debug the programs

Problem Types

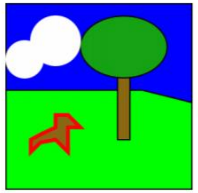
Copy a File



Find the Shortest Path Between Locations



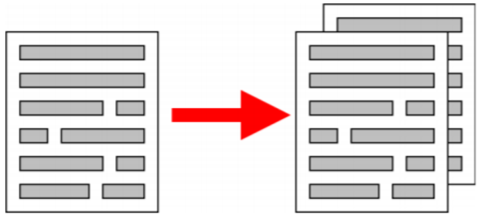
Identify Objects In Photographs



Problem Types

- Problem statement is clearly defined (or) not
- Is there a scope of improvement in the way the solutions are derived

Copy a File



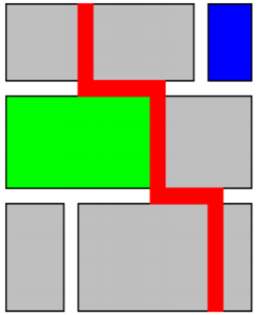
Problem statement : Clearly defined

Scope of Improvement : Straightforward solution

Problem Types

- Problem statement is clearly defined (or) not
- Is there a scope of improvement in the way the solutions are derived

Find the Shortest Path Between Locations



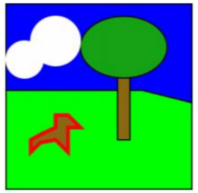
Problem statement : Clearly defined

Scope of Improvement : possibility to improvise

Problem Types

- Problem statement is clearly defined (or) not
- Is there a scope of improvement in the way the solutions are derived

Identify Objects In Photographs



Problem statement : Not clear

Scope of Improvement : ?

Pause & Think

Can you find the problems where there is a scope for devising efficient algorithms?

1. Search for a keyword in the document
2. Speech Recognition system
3. Given a set of activities , find the maximum number of non-overlapping activities [Assume for each activity start time and finish time is provided]

Finding GCD of two numbers

- Problem Statement
- Naïve Algorithm
- Efficient Algorithm

Problem Statement

Input : Two positive numbers a and b , $a, b \geq 0$

Output: gcd (a, b)

Definition

For any two integers a and b, their greatest common divisor is the largest integer d which divides both a and b.

Application - Cryptography

GCD Example

GCD (9,0)		9
GCD (13,45)	1	1
GCD (35,15)	1, 5	5

Finding GCD of two numbers

- Problem Statement
- Naïve Algorithm
- Efficient Algorithm

Naïve Algorithm

```
Function GCD (a, b)
best = 0
for $d$ from 1 to $a+b$:
        if $a|d$ and $b|d$ :
                best = $d$
return best
```

Not Optimal for large numbers with more number of digits

Runtime : No of operations

$a+b$

Example

gcd(3918848, 1653264)

Extremely slow !!

Need for an algorithm which can speed up the process

Finding GCD of two numbers

- Problem Statement
- Naïve Algorithm
- Efficient Algorithm

Euclidean algorithm

Lemma

If a' is a remainder when a is divided by b , then

$$gcd(a, b) = gcd(b, a')\ a \geq b$$

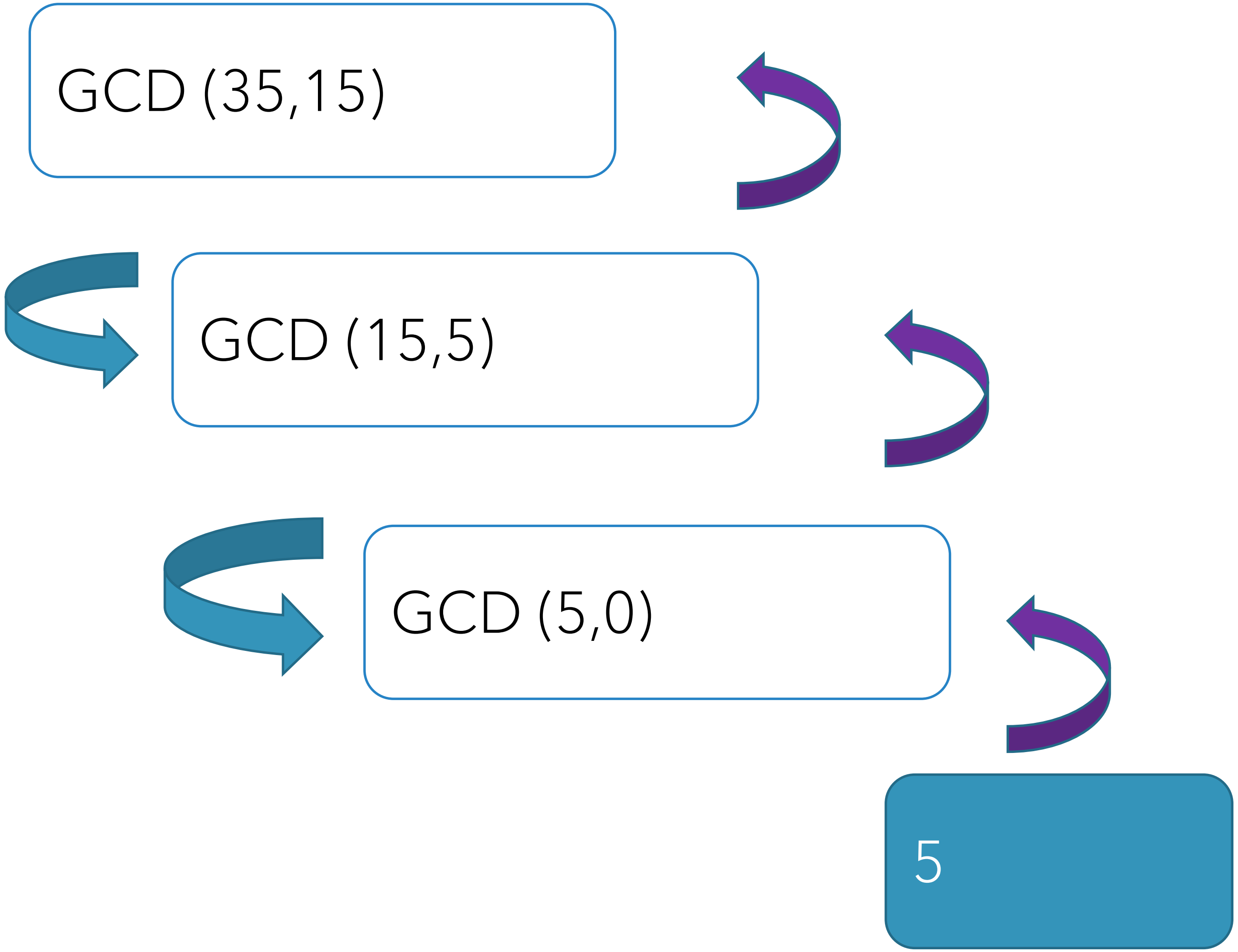
Proof

- If a is divided by b, then $\boldsymbol{a = a' + bq}$
- $\boldsymbol{d}$ divides a and b , then it should also divide $\boldsymbol{a'}$

Euclidean algorithm

```
Function EuclideanGCD (a, b)
If(b==0):
        return a
else
        return EuclideanGCD(b, a%b)
```

Example



GCD (35,15)

GCD (15,5)

GCD (5,0)

5

Example

GCD (35446,1510)

GCD (35446,1510)

GCD (1510,716)

GCD (716,78)

GCD (78,14)

GCD (14,8)

GCD (8,6)

GCD (6,2)

GCD (2,0)

Runtime

- Number of operations depends on a and b
- At every iteration, problem is reduced by half of the numbers

  ab → ab /2 → ab/4 → ab/8 → 1

  $a\,b/2^{\mathrm{i}}$ = 1

  $\mathrm{i} = \log_2(ab)$
- For 100 digit numbers, this algorithm will take just 600 steps

Pause & Think

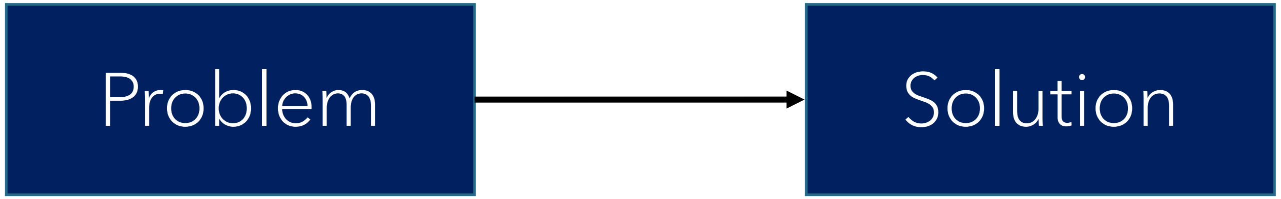
What would be the GCD of numbers (3918848, 1653264)

Summary

- Naïve Algorithm is slow
- Euclidean is more efficient
- To create efficient solutions, some interesting properties can be exploited

Problem Solving

How do we devise an algorithm ? (or) What is the primary criteria for an algorithm?



Problem

Solution

Correctness is important
Solution should be universal
For any problem instance, it should yield correct solution

Debugging

- Finding logical errors are always difficult
- Errors happen as the user is concerned with known test cases
- Corner cases (or) edge cases [input cases that lie at the extreme of the problem space] are not looked upon

Stress Test

- Random generation of test cases
- Comparing the output of the algorithm with some well known method

Thank You
Happy Learning

Success is always inevitable with Hard Work and Perseverance