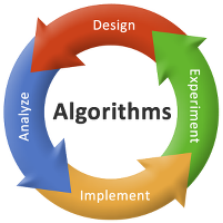
The Mathematics of Algorithms

Course: Algorithms

Faculty: Dr. Rajendra Prasath



Autumn 2018

Algorithmic Thinking

This class covers different aspects of **Algorithmic Thinking** to be considered in solving a given specific problem. This part also illustrates the effectiveness of algorithms with respect to the chosen Data Structures

Size of the Problem Instance

- Size of the Problem Instance
  - Scale the algorithm for sufficiently large n ?
  - How fast the algorithm could behave when we scale the input size n sufficiently large enough??
  - Performance degradation
  - Analysis complex properties
  - How to prepare the solution to work for the given problem with sufficiently large input size ?

Algorithm Pattern

- Nomenclature
  - Name (Descriptive Name of the algorithm)
  - Context (illustrating an essential part of the Algo)
  - Facts (Properties that could cause anyone to choose the algorithm specifically)
  - Consequences (Advantages and Disadvantages)
  - Analysis of the algorithm (understanding the behavior of the algorithm
    - why does an algorithm behave like this!!
    - Can we come up with lemmas and proofs to explain the behavior of the algorithm?
  - Alternatives (find and compare other competitive variations of solutions to the same problem)

Example - 1

- How to handle Memory Leaks in C programming?
  - malloc()
    - Does memory allocation
  - free()
    - Does memory deallocation
  - exit()
    - End of the program

- Consider the search and deletion of an element in the linked list !!

Example -2

- Task: Find the largest element in a given list L

- The given list, say L has n elements
- We have to fine the element which is the largest element than all other elements.

- Issues?
  - n elements can be unique (no repetitions)
  - There can be more than one instance of the largest element
  - The largest element can be seen in the worst case scenario

Example -2 (contd)

- Task: Find the largest number in a list of numbers of random order
- Solution: look at every number in the list

- **High-level description (Linear Scan):**
  - If there are no numbers in the list then there is no highest number
  - Assume the first number in the list is the largest number
  - For each remaining number in the list: if this number is larger than the current largest number, consider this number to be the largest number in the list
  - When there are no numbers left in the list to scan through, consider the current largest number to be the largest number of the list

Example – 2: Find the Largest

Task: Find the largest element in a given list L

Algorithm **getLargest**

Input: A list of numbers L having n elements

Output: The largest number in the list L

```
begin
    if n = 0 return null
        largest ← L[0]
    for each item in L, do
        if item > largest, then
            largest ← item
    return largest
end
```

Example – 3: Find GCD

Write an algorithm to find the greatest common divisor (GCD) of two numbers using recursion

Example:

m = 21, n = 9

GCD (21, 9) = ??

Answer: 3

Example – 3: Euclid Algorithm

Euclid Algorithm:

Find the greatest common divisor (GCD) of any two integers:

Let us take two numbers ‘a’ and ‘b’

Steps (High level Description):

1) If a<b, exchange a and b.
2) Divide a by b and get the remainder, r. If r=0, report b as the GCD of a and b.
3) Replace a by b and replace b by r.
4) Return to the previous step

Example – 3: Euclid Solution

Code:

```
int gcd (int n, int m) {
    if ( m == 0 ) return  n;
    return gcd(m, n%m);
}
```

Test Cases:

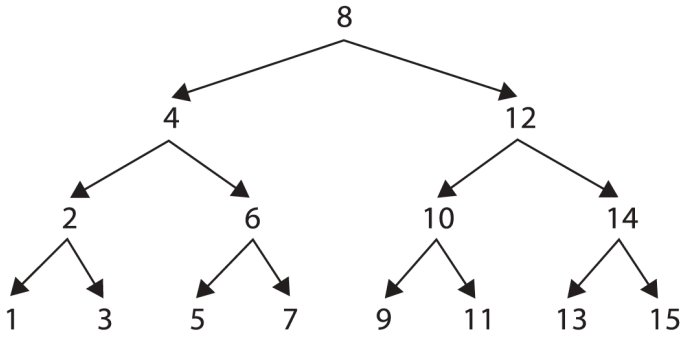
1) n = 21, m = 6, output = 3
2) n = 30, m = 95, Output = 5
3) n = 77, m = 343 Output = 7
4) n = 4, m = 11, Output = 1

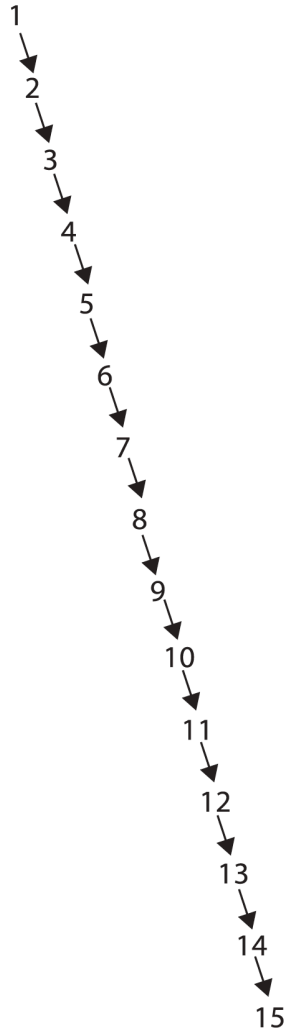
How to make Search easier?

- Given an array of n elements
- List

1	4	9	15	7	12	13	6

- Linked list
- Trees and its variants





Sequential Search

- Let us consider the list of n elements

Solution:

Algorithm **findNumber**(list, num)

```
    begin
        for each element t in the list, do
            if (list[i] = t) then
                    return true
        return false;
    end
```

Search - Complexity

- Best
  - O(1) – Constant time
- Average
  - O(n) – Linear time
- Worst
  - O(n) – Linear time
- Which Data Structure is used … ??
- Choice of the data structure makes this complexity different

Compute the Sum

- Given: n – an integer
- **Task:**
  - How to compute the sum of first n natural numbers?

- **Solutions:**
  A) Simply use a for loop
  B) Have two pointers at the end point of the list of first n natural numbers and do the sum until they cross over each
  C) Gauss Technique
  D) Arithmetic Progression $(s + (n-1)d)/2$

Help among Yourselves?

- **Perspective Students** (having CGPA above 8.5 and above)
- **Promising Students** (having CGPA above 6.5 and less than 8.5)

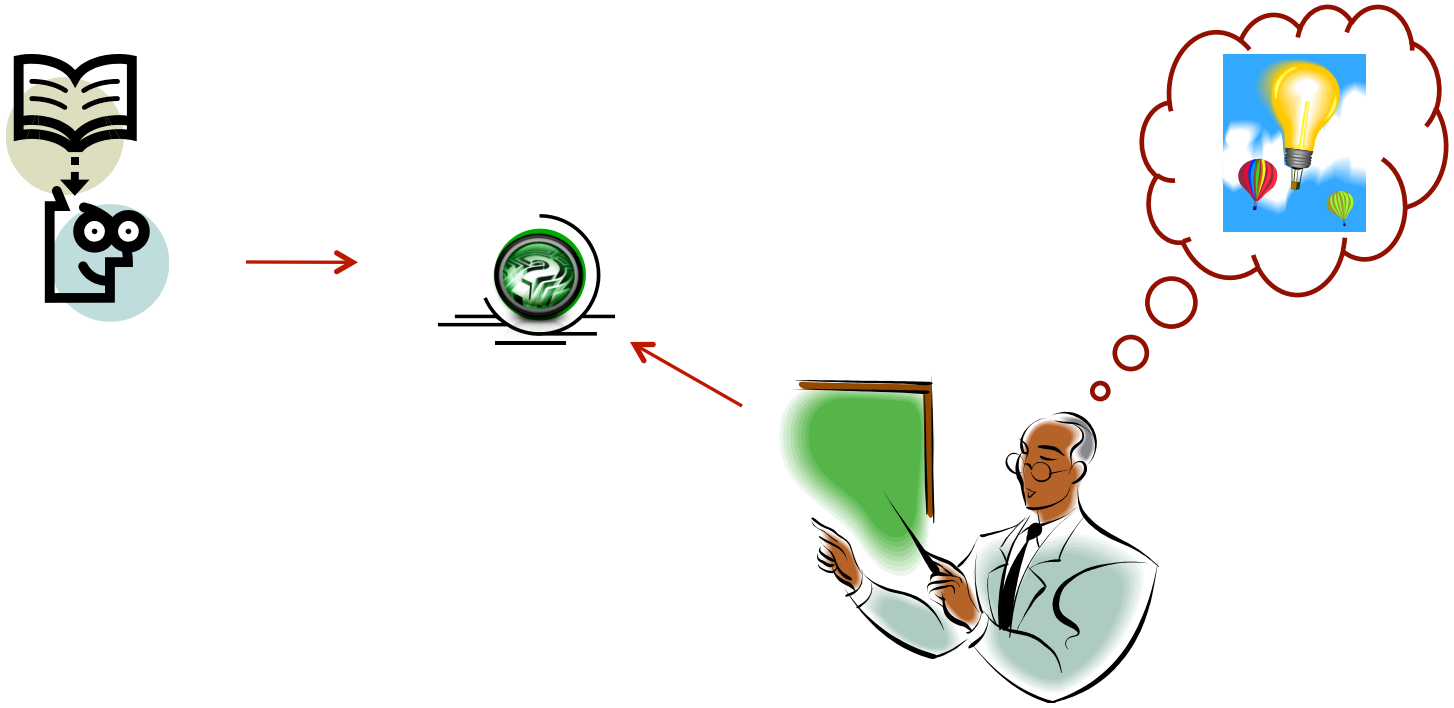
- **Needy Students** (having CGPA less than 6.5)
  - Can the above group help these students? (Your work will also be rewarded)

- You may grow a culture of **collaborative learning** by helping the needy students

Assistance

- You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- TA s would assist you to clear your doubts.
- You may leave me an email any time (email is the best way to reach me faster)

Thanks …



… Questions ???