Analysis of Algorithm

Amypo Technologies Pvt Ltd

Agenda

- Introduction
- Fundamentals of Algorithmic Problem Solving
- Important Problem Types

Introduction

What is an Algorithm?

- The algorithm is defined as a collection of unambiguous instructions occurring in some specific sequence and such an algorithm should produce output for given set of input in finite amount of time.

Properties of Algorithm

Non -ambiguity :

Each step in an algorithm should be non-ambiguous. That means each instruction should be clear and precise. The instruction in an algorithm should not denote any conflicting meaning. This property also indicate the effectiveness of algorithm.

How to Write Algorithm?

Let us understand some rules for writing the algorithm.

- Algorithm is a procedure consisting of heading and body. The heading consists of keyword Algorithm and name of the algorithm and parameter list.
- Then in the heading section we should write following things:

```
//Problem Description
//Input :
//Output :
```

- Then body of an algorithm is written, in which various programming constructs like if, for, while or some assignment statements may be written.
- The compound statements should be enclosed within { and } brackets.

- Single line comments are written using // as beginning of comment
- The identifier should begin by letter and not by digit. An identifier can be a combination of alphanumeric string.
- Using assignment operator an <-- assignment statement can be given.

  For instance:

  Variable <-- expression

- There are other types of operators such as Boolean operators such as true or false.Logical operators such as and, or, not. And relational operators such as <, <=, >, >=,=.
- The array indices are stored with in square brackets '[' ']'. The index of array usually start at zero. The multidimensional arrays can also be used in algorithm.

- The inputting and outputting can be done using read and write.

  For example:

  ```
  write("This message will be displayed on console");
  read(val);
  ```

- The conditional statements such as if-then or if-then-else are written in following form:

  ```
  if (condition) then statement
  if (condition) then statement else statement
  ```

If the if-then statement is of compound type then { and } should be used for enclosing block.

- while statement can be written as:

  ```
  while (condition) do
  {
  statement 1
  statement 2
  statement n
  }
  ```

- The general form for writing for loop is:

```
for variable <-- value 1 to value n do
{
statement 1
statement 2
:
:
statement n
}
```

- The repeat - until statement can be written as:

```
repeat
        statement 1
        statement 2
        :
        statement n
until (condition)
```

- The break statement is used to exit from inner loop. The return statement is used to return control from one point to another. Generally used while exiting fromfunction.

Fundamentals of Algorithmic Problem Solving

Understanding the problem

- This is the very first step in designing of algorithm.
- In this step first of all you need to understand the problem statement completely.
- While understanding the problem statements, read the problem description carefully and ask questions for clarifying the doubts about the problem.

Decision making

- After finding the required input set for the given problem we have to analyze the input.
- We need to decide certain issues such as capabilities of computational devices, whether to use exact or approximate problem solving, which data structures has to be used, and to find the algorithmic technique for solving the given problem.
- This step serves as a base for the actual design of algorithm.

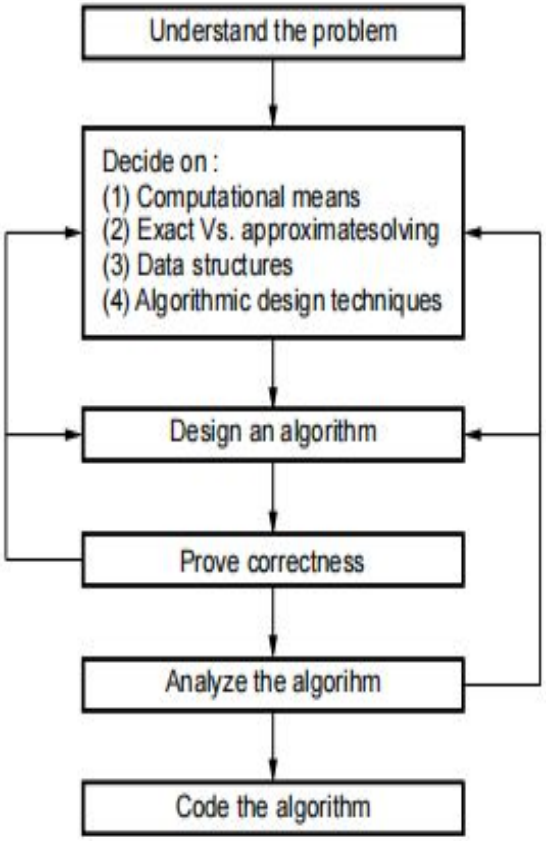

Fig. 1.2.1 Algorithmic analysis and design process

Design an algorithm

- There are various ways by which we can specify an algrithm.
  - Using natural language
  - Pseudo code
  - Flow chart

Prove correctness

- After specifying an algorithm we go for checking its corectness.
- The proof of correctness of an algorithm can be complex sometimes.

Analysis of algorithm

- while analysis an algorithm we should consider follwing factors-
  - Time complexity
  - Space complexity
  - Simplicity
  - Generality
  - Range of inputs

Analysis of Algorithm

- Efficiency of an algorithm can be in terms of time or space. Thus, checking whether the algorithm is efficient or not means analyzing the algorithm.
- There is a systematic approach that has to be applied for analyzing any given algorithm. This systematic approach is modelled by a framework called as analysis framework.
- Let us understand the analysis framework in detail.

Space Complexity

- The space complexity can be defined as amount of memory required by an algorithm to run.
- To compute the space complexity we use two factors: Constant and instance characteristics. The space requirement S(p) can be given as :

  S(p) = C + Sp

- where C is a constant i.e. fixed part and it denotes the space of inputs and outputs. This space is an amount of space taken by instruction, variables and identifiers.
- And Sp is a space dependent upon instance characteristics.

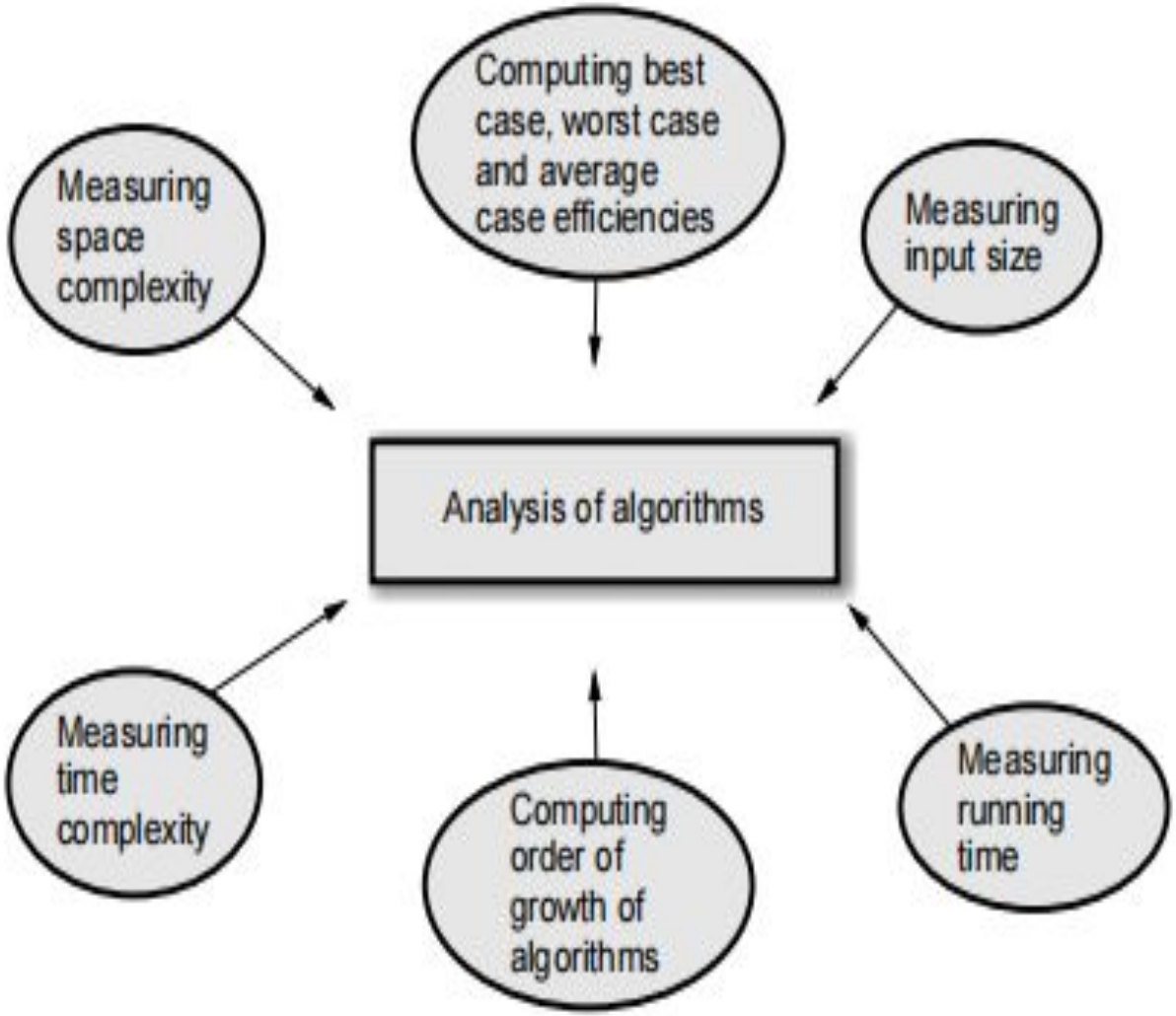

Fig. 1.5.1 Analysis of algorithms

Important Problem Types

- Sorting
- Searching
- Numerical problems
- Geometric problems
- Combinatorial problems
- Graph problems
- String processing problems

1.Sorting

- Sorting means arranging the elements in increasing order or in decreasing order (also called as ascending order or descending order respectively).
- The sorting can be done on numbers, characters (alphabets), strings or employees record.
- For sorting any record we need to choose certain piece of information based on which sorting can be done.
- For instance : For keeping the employees record in sorting order we will arrange the employees record as per employee ID. Similarly one can arrange the library books according to the title of the books.

What is the need for sorting ?

- The usefulness of sorting is that we can search any desired element efficiently. That is why the telephone directory, dictionary or any database is always in sorted order.
  Properties of Sorting
- In computer science there are numerous sorting algorithms that are available. All can be analyzed by two important properties : Stable property and in-place property.
- The algorithm is said to be stable if it preserves the relative ordering of items with equal values. For example : Consider a following list -

Roll Number	Name
10	Padma
20	Jayashri
30	Sarika
40	Sarika
50	Poonam

2. Searching

- Searching is an activity by which we can find out the desired element from the list.
  The element which is to be searched is called search key. There are many searching
  algorithms such as sequential search, binary search, Fibonacci search and many more.

How to choose best searching algorithm ?

- One can not declare that a particular algorithm is best searching algorithm because some algorithms work faster, but then they may require more memory. Some algorithms work better if the list is almost sorted. Thus efficiency of algorithm is varying at varying situations.

Searching in dynamic set of elements

- There may be a set of elements in which repeated addition or deletion of elements
  occur. In such a situation searching an element is difficult. To handle such lists
  supporting data structures and algorithms are needed to make the list balanced
  (organized).

3. Numerical Problems

- The numerical problems are based on mathematical equations, systems of equations, computing definite integrals, evaluating functions and so on.
- These mathematical problems are generally solved by approximate algorithms.
- These algorithms require manipulating of the real numbers;
- hence we may get wrong output many times.

4. Geometric Problems

- The geometric problems is one type of problem solving area in which various operations can be performed on geometric objects such as points, line, polygon.
- The geometric problems are solved mainly in applications to computer graphics or in robotics.

5. Combinatorial Problems

- The combinatorial problems are related to the problems like computing permutations
  and combinations.
- The combinatorial problems are most difficult problems in computing
  area because of following causes
- As problem size grows the combinatorial objects grow rapidly and reach to a huge value.
- There is no algorithm available which can solve these problems in finite amount of time.
- Many of these problems fall in the category of unsolvable problems.
- However there are certain problems which are solvable.

6. Graph Problems

- Graph is a collection of vertices and edges. The graph problems involve graph traversal algorithms, shortest path algorithms and topological sorting and so on.
- Some graph problems are very hard to solve. For example travelling salesman problem, graph coloring problems.

7. String Processing Problems

- String is a collection of characters. In computer science the typical string processing algorithm is pattern matching algorithm.
- In these algorithms particular word is searched from the text.
- The algorithms lying in this category are simple to implement.