Fundamentals of algorithm & problem Solving:

The essence of this architecture is captured by RAM.

→ In this instruction executed by one after another. one operation at time

Understand the problem

↓

Decide on computational means Exact Vs approximate Solving data structures. Algorithm design techniques.

↓

Design an algorithm.

↓

Prove correctness

↓

Analyze the alg

↓

Code the alg

seq ↓ parallel

→ Seq. Algorithm designed to be executed on such machine

→ Alg that takes adv of this capability are called parallel

→ Begin
I/P A
I/P B
Compute SUM A+B
End.

1. Understanding the problem:

→ To dirign an alg
→ Understand the problem completely
→ Ask questions to clarify the doubts.
→ Divide the problem into smaller problems until they becomes managable size.

2. Computational devices:

→ Machine instruction executed at one operation at a time.
→ Executed only sequential alg.
→ The operation executed in parallel.

3. Choosing blw exact & appropriate problem Solving:

→ The Decision is to choose blw solving the problem exactly & approximately.

Exact alg:

4. Solving Data structure:

→ OOps, DS is important for both DAA.

5 Alg Design Technique:

Def: Its a general approach to solving problems algorithmically that is applicable to a variety of problems from different areas of computing.

6. Methods of specifying an alg:-

1. Euclid alg ← simple eng statment. step by step format.
2. Pseudo code. 3. Flow chart.

4. proving an alg correctness:

→ Alg has been specified, then its must be correct.

→ If the alg is incorrect, we need to either redesign it.

→ The correctness for an approximation alg, is if we are able to show that the error produced.

Analyze an alg:

→ An alg implemented, uses the computer primary memory & CPU.

→ We analyze the amount of resources (time & space).

→ Analyze is done at 2 stage { Prior, Posterior }

→ Qualities of alg (i) correctness, Efficiency, Simplicity (iv) Generality.

→ Alg efficiency { Time : how fast the alg runs; space : how much extra memory the alg needs.

→ Alg should be simple, because its easy to understand & easy to program.

Coding an alg:

→ The correctness of the pgm is done by proving it mathematically.

→ The validity of the program is done by Testing.

→ Using code optimization technique.

Important problem types:

- → Sorting
- → Searching
- → String processing
- → Graph problem
- → Combinatorial problem
- → Geometric problems
- → Numerical problems.

Fundamentals of analysis of Alg efficiency:-

→ When alg implemented, uses the computer primary memory & CPU.

→ Analysing the amount of resource (time / space) for solving problems.

→ Alg efficiency depends on resource { Running Time, memory space.

Analyze Framework:

① Alg efficiency { Time, space.

② Measuring an Input size

③ Units for measuring Running Time.

④ Orders of Growth

⑤ Approaches in Analysis of alg. { Theortical, Empirical.

⑥ Worst, Best, Average case Efficiencies.

Measuring Space Complexity:-

→ An amount of storage space or memory required to execute.

$$S(P) = C + SP$$

Asymptotic Notations:

where

S(P) → Space complexity of a pgm 'P'

C → Constants for I/P & O/P.

Sp → Instance characteristics of a pgm

(2) Meaning Time complexity:

→ An amount of computer time required to completion.

→ Multiuser s/y execution time depends on

- ↳ s/y load.
- ↳ Instruction set used
- ↳ No. of other pgm running.
- ↳ speed of underlying h/w.

→ Time complexity is given in terms of Frequency count.

→ Denoted by Big oh notation (O).

(3) Meaning an I/P size:-

→ The efficiency of an alg can be computed as a function

→ I/P size is passed as a parameter.

→ It can be ↳ exact value

↳ Approximate value.

Ex:

Spell-check Alg.

→ No of char } I/P size
→ No of words }

④ Measuring Running Time:

⇒ Identify basic operation.
⇒ Understand the concept of basic operation.
⇒ Compute Total no. of time taken by basic operation.

$$T(n) = C_{op} \cdot C(n)$$

$T(n)$ → Running time of basic operation.
C_{op} → Time take basic "
$C(n)$ → No. of times the operation needs to be executed.

⑤ Computing Order of Growth:

⇒ Measuring the performance of an alg in relation with I/P size.

Eg:

$\log n$, $n \log n$, n^2, 2^n ...

⑥ Worst, Best, Average case efficiency

Worst case

| 10 | 20 | 30 | ←

$C_w(n) = n$.

Best

→ | 10 | 20 | 30 |

$C_B(n) = n$.

Average

↓

| 10 | 20 | 30 |

$C_{Avg}(n) = n$.