Fundamentals of Algorithmic problem solving:

The following are the steps to be followed in designing and analyzing an algorithm.

understand the problem

↓

decide on : Computational means,
exact Vs approximate problem solving
DS
algorithm design technique

↓

Design an algorithm

↓

Prove the correctness

↓

Analyze the algorithm

↓

code the algorithm

Understanding the problem :-

↳ i/p to the problem is vital for algorithm to specify an instances to solve a problem.

↳ have to specify the range of instance.

↳ All before, we need to understand the problem.

Ascertaining the capability of computational Device.

↳ have to analyze the capability of machine in which algorithm is processed.

↳ von-Neumann architecture runs by RAM.

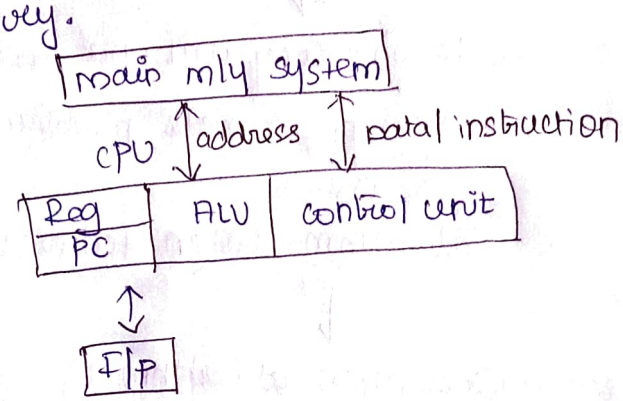
↳ here instructions are executed one after other.

↳ Algorithm operations on Non Neumann is called sequential algorithm.

↳ Algorithm operates on "Harvard architecture" are called "parallel algorithms".

(x)

Von Neumann architecture - stored program concept where program data and instruction are stored in same memory.



Harvard architecture - pigital computer architecture where there is seperate storage and seperate buses for instruction and data.

Bus - computer architecture or communication system which transpers data b/w components inside the computer or b/w computers.

↳ data bus and address bus., control bus.

Choosing between exact and approximate problem solving

Based on solution, the solution are classified as exact and approximation algorithm.

issues in approximation algorithm

↳ for sqrt, non linear equation

↳ reduces operation performance

↳ It is sophosticated when algorithm is exact.

Deciding on data structure

Algorithm + Data Structure = Program.

Algorithm Design techniques:

↳ approach to solve problem algorithmically.

This is vital for

↳ to provide better guidance for designing for new problems

↳ They are cornerstone for computer programs.

Methods for specifying an algorithm

-> pseudocode -> mix of natural language and programming language

-> flowchart are base for writing algorithms.

Proving an algorithm's correctness.

↳ Correctness is very essential for an algorithm to use.

↳ we can use mathematical induction -> for algorithms iteration steps.

↳ But there is only instance is enough for proving that an algorithm fails.

↳ for approximation algorithm - correctness is less.

Analyzing an algorithm.

Two kind of algorithm efficiency

↳ time complexity -> how fast algorithm is.

↳ Space complexity -> how much extra m/y requires.

-> requires simplicity -> easy to debug.

-> generality - general algorithm does not applicable for all similar type of problems.

->verification -> is algo designed as per requirements.

->validation - by testing and debugging

↳ Compiler for code optimization - sped up programs.

->optimality and -> ambiguity.