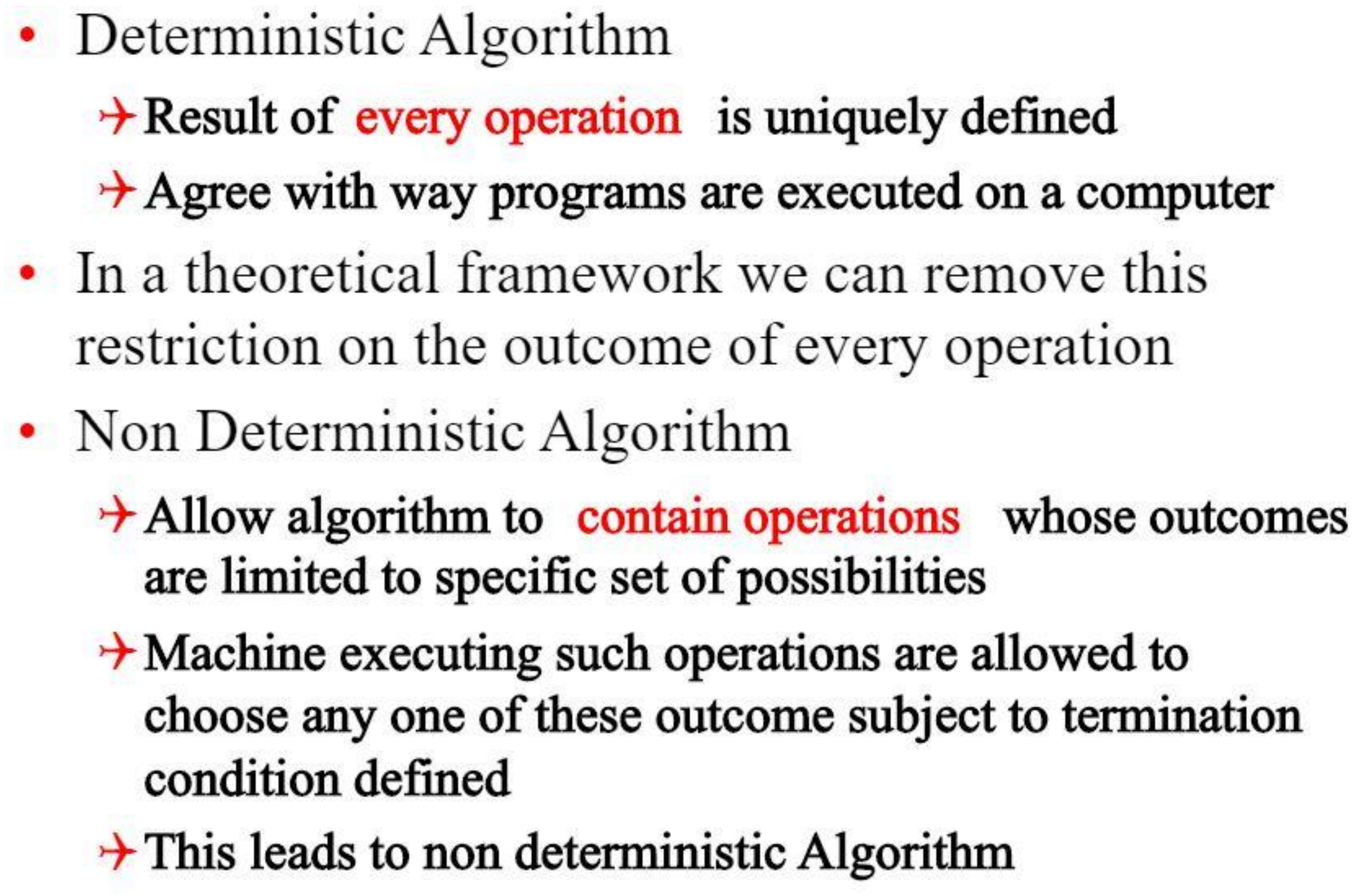
Unit-11 Algorithms

Topics

- Deterministic and Non-deterministic Algorithm
- Divide and conquer Algorithm
- Series and parallel Algorithm
- Heuristic and approximation algorithms

Deterministic and Non-deterministic Algorithm



- Deterministic Algorithm
  - Result of every operation is uniquely defined
  - Agree with way programs are executed on a computer
- In a theoretical framework we can remove this restriction on the outcome of every operation
- Non Deterministic Algorithm
  - Allow algorithm to contain operations whose outcomes are limited to specific set of possibilities
  - Machine executing such operations are allowed to choose any one of these outcome subject to termination condition defined
  - This leads to non deterministic Algorithm

- Introduce new functions
  - Choice (set S) – arbitrarily choose one element of S
  - Failure() – signals an unsuccessful completion
  - Success()- signals a successful completion
  - X = choice(1,n) results in x being assigned any one of integer in the range of 1 to n, but no rule how this choice is to be made !
- Failure and success – just to define a computation of algorithm and cannot be used to effect a return
- Whenever there is a set of choices that leads to a successful completion, then one such set of choices is always made and algorithm terminates successfully

- NDA terminates unsuccessfully if and only if there is no set of choices leading to success signal
- Hence depending upon order in which choices are made – affects the successful / unsuccessful termination of NDA
- Computing time of Choice(), Success(),failure() is O(1) – Constant
- Machine capable of handling NDA is called as nondeterministic Machine (NDM)
- Although, NDM do not exist in practice, there are certain problems that can not be solved by deterministic algorithm.

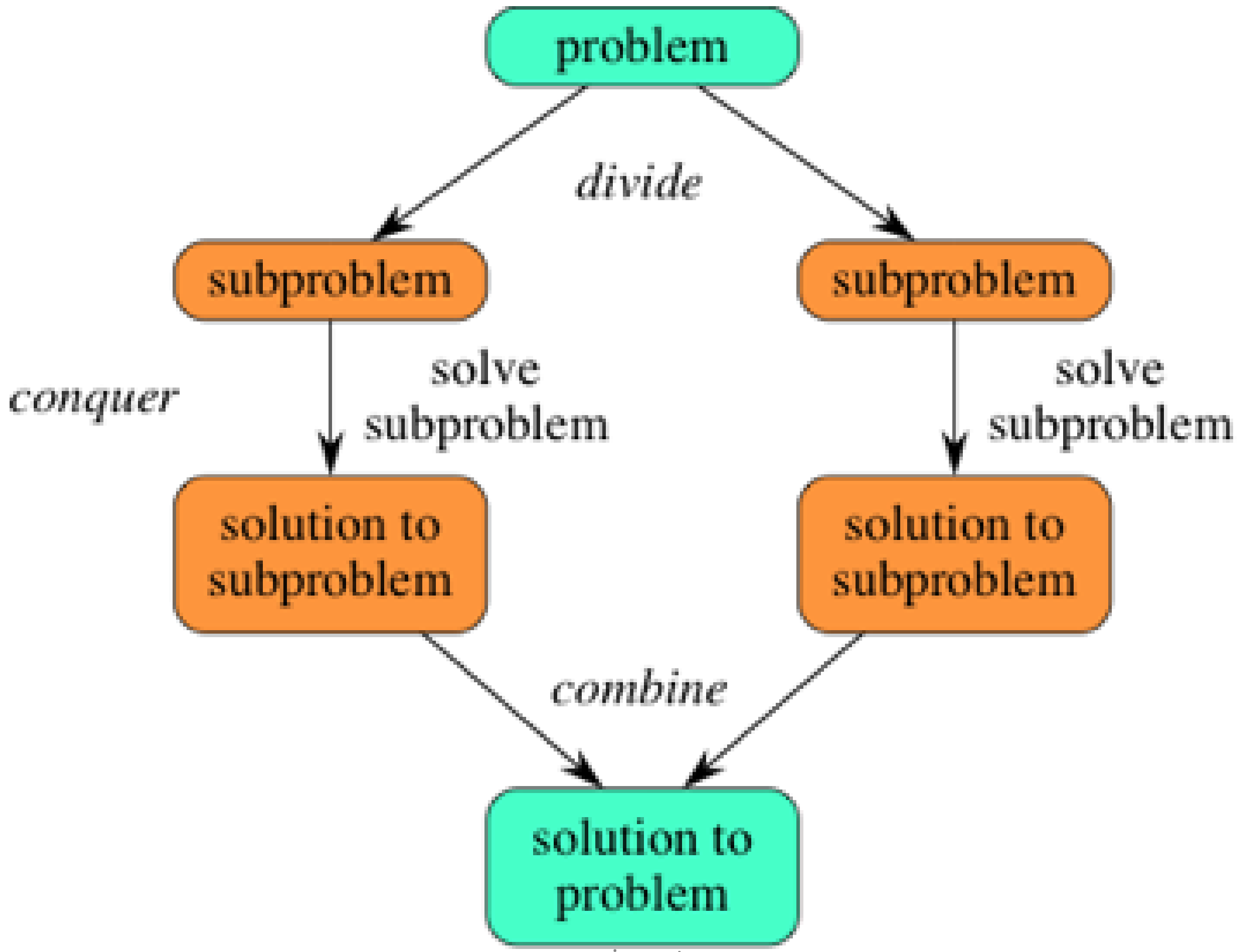
P and NP

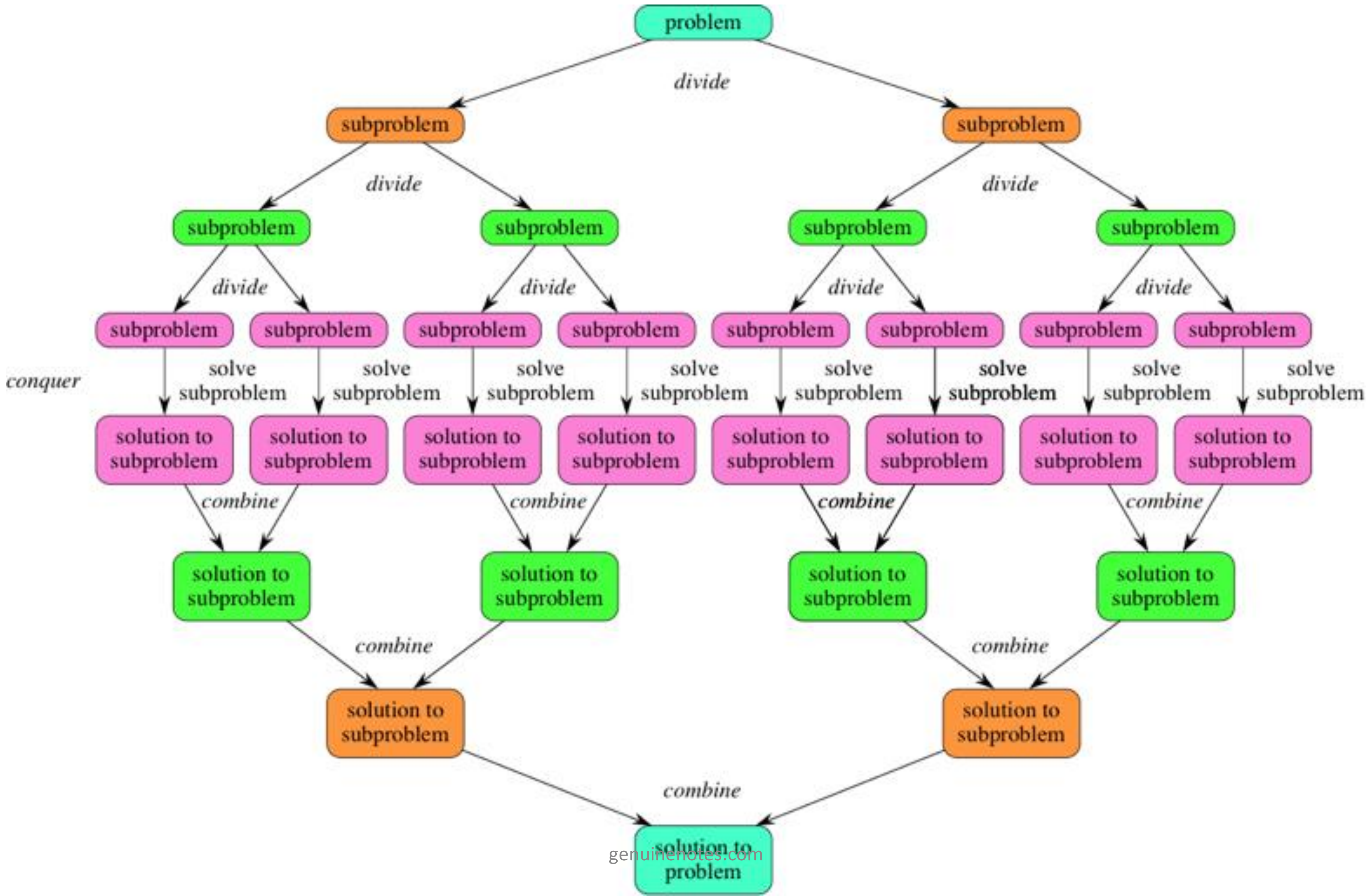
- Let us now define some terms
  - **P:** The set of all problems that can be solved by deterministic algorithms in polynomial time
- By *deterministic* we mean that at any time during the operation of the algorithm, there is only one thing that it can do next
- A nondeterministic algorithm, when faced with a choice of several options, has the power to "guess" the right one.
- Using this idea we can define NP problems as,
  - **NP:** The set of all problems that can be solved by nondeterministic algorithms in polynomial time.

Home Work

- Difference between deterministic and non-deterministic algorithm with their performance and examples.

Divide and conquer Algorithm

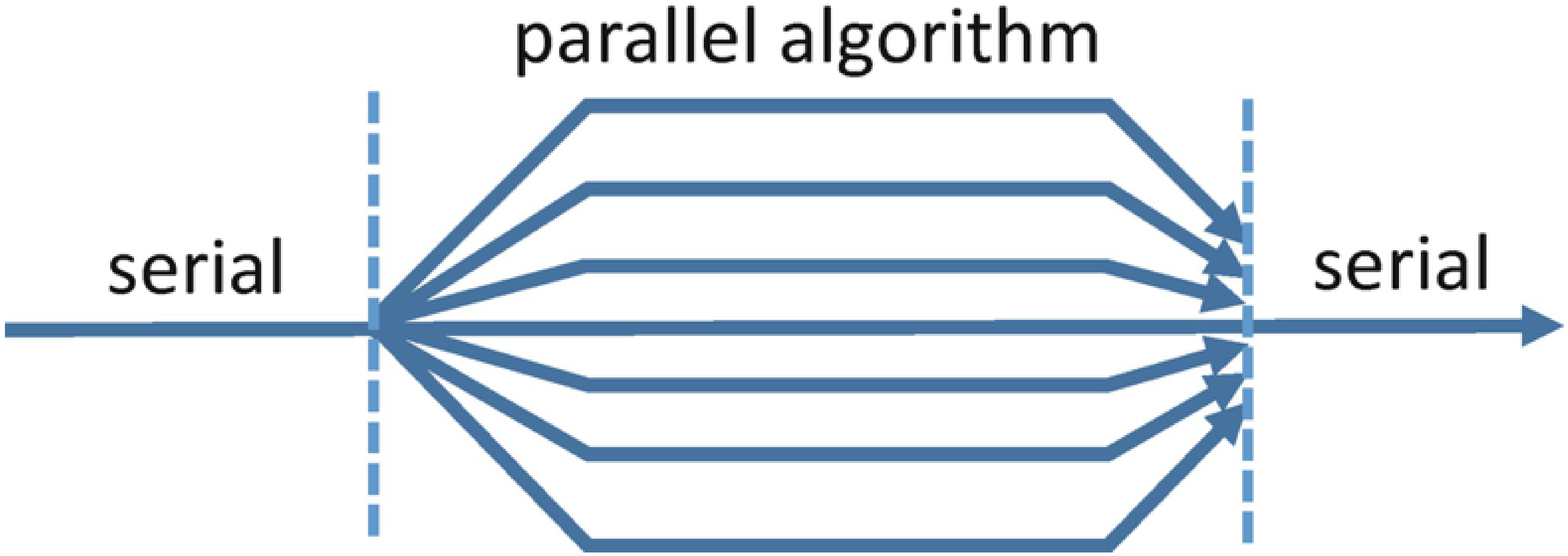



problem
divide
subproblem
subproblem
divide
divide
subproblem
subproblem
subproblem
subproblem
divide
divide
divide
divide
subproblem
subproblem
subproblem
subproblem
subproblem
subproblem
subproblem
subproblem
conquer
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solve subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
combine
combine
combine
combine
solution to subproblem
solution to subproblem
solution to subproblem
solution to subproblem
combine
combine
solution to subproblem
solution to subproblem
combine
solution to problem

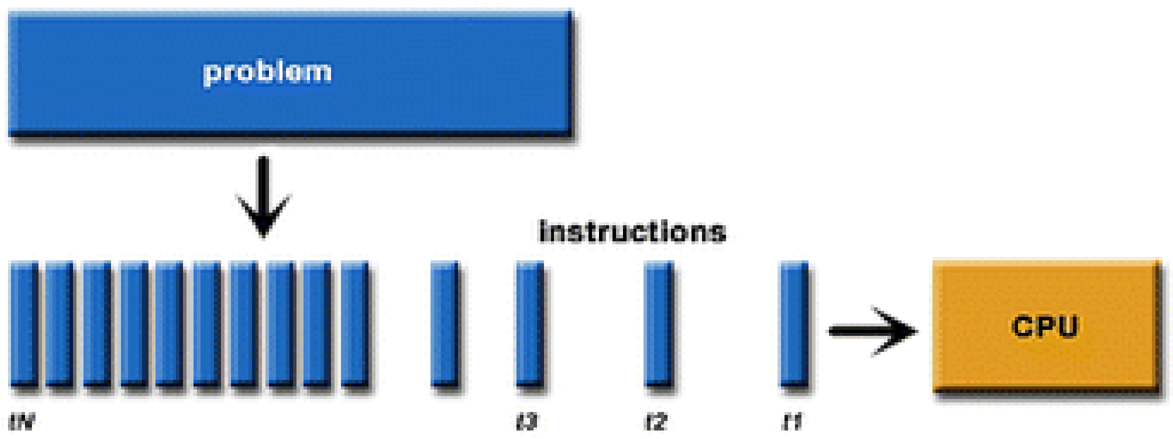
Examples

- Quick sort
- Merge Sort
- Binary Search
- Etc.

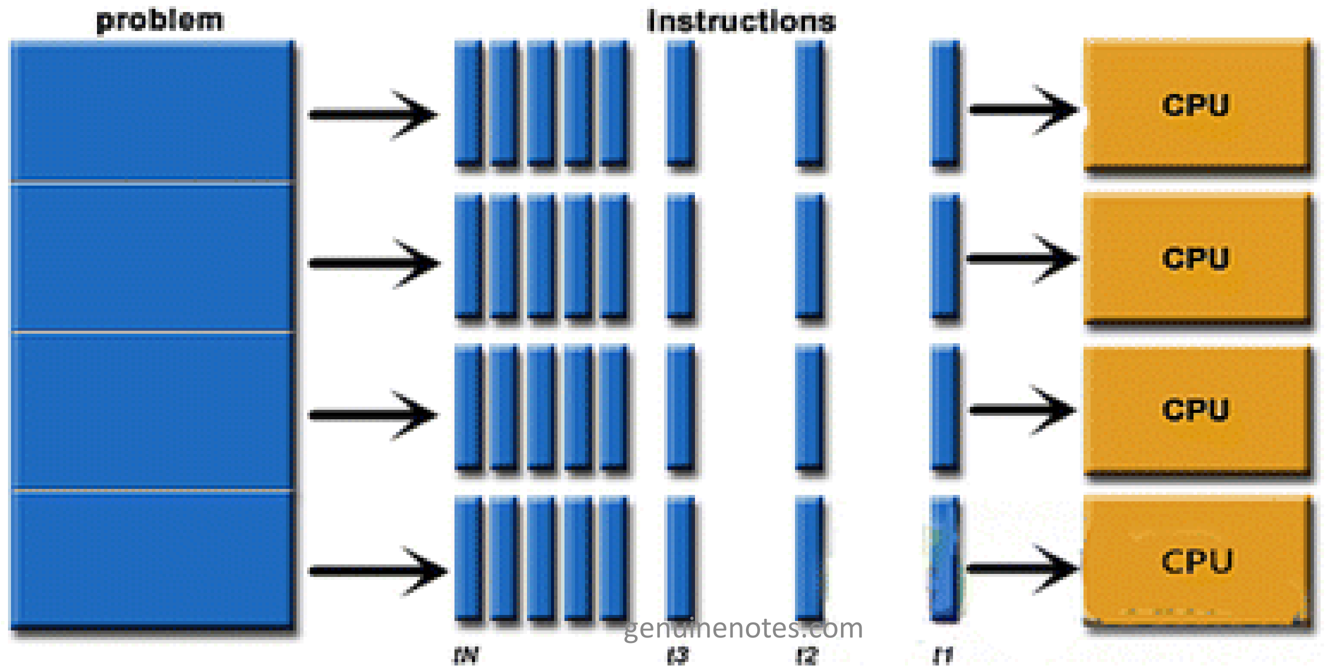
Series and parallel Algorithm



Serial operation schematic diagram



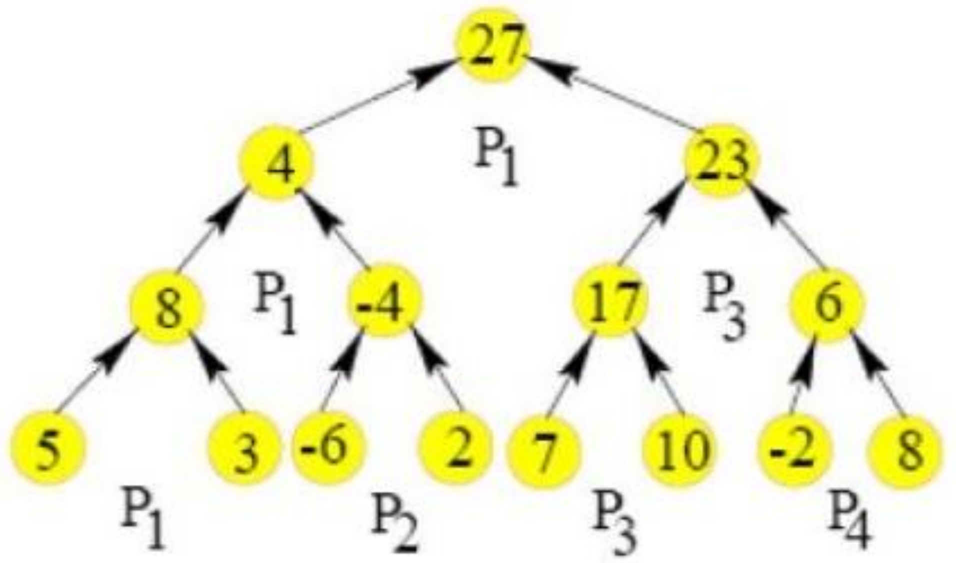
Parallel computing



What is parallel Algorithm?

- Imagine you need to find a lost child in the woods.
- Even in a small area, searching by yourself would be very time consuming.
- Now if you gather some friends and family to help you, you could cover the woods much faster.

A simple parallel algorithm



$T(n)=\log(n)$

Adding n numbers in parallel

A parallel algorithm or concurrent algorithm, as opposed to a traditional sequential algorithm, is an algorithm which can be executed a piece at a time on many different processing devices, and then combined together again at the end to get the correct result.

Heuristic and approximation algorithms

1. In mathematics and computer science, an optimization problem is the problem of finding the best solution from all feasible solutions.
2. The objective may be either min. or max. depending on the problem considered.
3. A large number of optimization problems which are required to be solved in practice are NP-hard.
4. For such problems, it is not possible to design algorithms that can find exactly optimal solution to all instances of the problem in polynomial time in the size of the input, unless $P = NP$.

- Brute-force algorithms
  1. Develop clever enumeration strategies.
  2. Guaranteed to find optimal solution.
  3. No guarantees on running time.
- Heuristics
  1. Develop intuitive algorithms.
  2. Guaranteed to run in polynomial time.
  3. No guarantees on quality of solution.
- Approximation algorithms
  1. Guaranteed to run in polynomial time.
  2. Guaranteed to get a solution which is close to the optimal solution (a.k.a near optimal).
  3. *Obstacle : need to prove a solution's value is close to optimum value, without even knowing what the optimum value is !*

Approximation Algorithm

Given an optimization problem $\mathcal{P}$, an algorithm $\mathcal{A}$ is said to be an *approximation algorithm* for $\mathcal{P}$, if for any given instance I, it returns an approximate solution, that is a feasible solution.

Types of Approximation

$\mathcal{P}$	An optimization problem
$\mathcal{A}$	An approximation algorithm
I	An instance of $\mathcal{P}$
$\mathcal{A}^*(I)$	Optimal value for the instance I
$\mathcal{A}(I)$	Value for the instance I generated by $\mathcal{A}$

1. Absolute approximation
   - $\mathcal{A}$ is an *absolute approximation algorithm* if there exists a constant k such that, for every instance I of $\mathcal{P}$, $|\mathcal{A}^*(I) - \mathcal{A}(I)| \leq k$.
   - For example, Planar graph coloring.
2. Relative approximation
   - $\mathcal{A}$ is an *relative approximation algorithm* if there exists a constant k such that, for every instance I of $\mathcal{P}$, $\max\{\frac{\mathcal{A}^*(I)}{\mathcal{A}(I)}, \frac{\mathcal{A}(I)}{\mathcal{A}^*(I)}\} \leq k$.
   - Vertex cover.