

Computational Complexity:

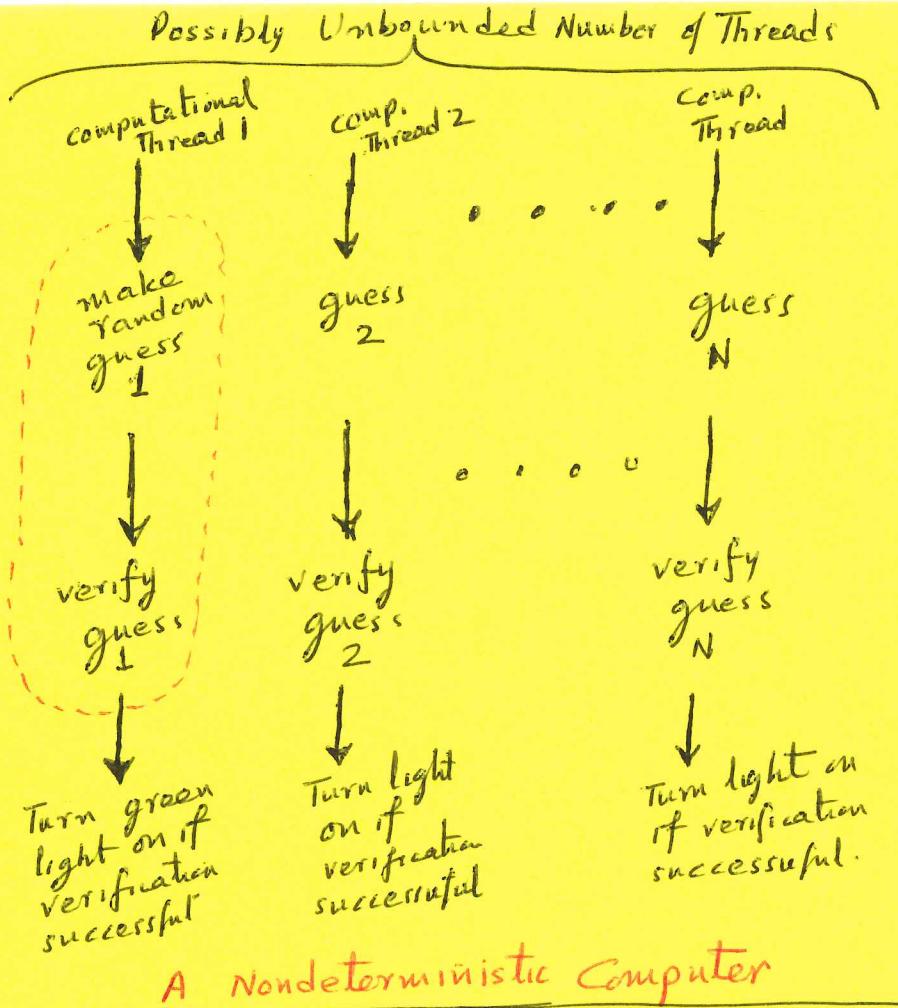
Some Basic Definitions

- ① **Problem (Π) :** A problem is a general question to be answered. The question will possess several parameters (or free variables) whose values are left unspecified in the formulation of the question.
- ② **Problem Instance (I) :** An instance of a problem is obtained by specifying values for all the parameters of the question.
- ③ **Algorithm :** An algorithm is said to solve a problem Π if that algorithm can be applied to any instance I of Π and is guaranteed always to produce a solution.
- ④ **Input Length :** The time requirements of an algorithm are frequently expressed in terms of a single variable — **the size of the problem instance**. The size is measured by the **input length** that is defined to be the number of symbols in the description of I under a reasonable encoding scheme.
- ⑤ **Time Complexity Function :** It is a function of the input length. For each possible value of the input length, the Time Complexity Function returns **the longest time** needed by the algorithm for a problem of that size. In other words, the Time Complexity Function is **the worst-case measure of the time required by the algorithm**.
- ⑥ **Polynomial-Time Algorithm :** It is an algorithm whose Time Complexity Function is $O(p(n))$ where n is the input length and $p(n)$ is a polynomial in n . [We say $f(n)$ is $O(g(n))$ if there exists a constant c such that $|f(n)| \leq c|g(n)|$ for all $n \geq 0$.]
- ⑦ **Exponential-Time Algorithm :** If the Time Complexity Function cannot be bounded by a polynomial, we may refer to the algorithm as an exponential-time algorithm (although that is an oversimplified characterization of such problems, as you will see later in this class).

- It is generally believed that a problem is not considered well-solved until a polynomial-time algorithm is known for it. Most exponential-time algorithms are merely variations on exhaustive search.

THE NONDETERMINISTIC COMPUTER

- Several important concepts of the complexity theory are best understood with the help of the computational model known as **The Nondeterministic Computer**.
- A nondeterministic computer solves a problem by guessing a solution and then verifying that guess. **All possible guesses are tried simultaneously, each in a separate thread of computation.**
- The threads must execute independently — there can be no communication between the threads.
- A thread that carries out successful verification turns on its green light to report that fact.
- There are no reports from the threads that are unsuccessful at verification.



A Nondeterministic Computer

- For a decision problem to belong to class NP, all of the computations in the dashed red oval shown above (in the thread that turns on the green light) must take place in polynomial time.

NP-class of Decision Problems

- A decision problem belongs to class NP if it can be solved in polynomial time by a nondeterministic computer. (A decision problem is one whose answer is yes or no.)

NP : Nondeterministic Polynomial

Example :

The Traveling Salesman Problem (TS) :

INSTANCE : A finite set $C = \{c_1, \dots, c_m\}$ of cities and the distances $d(c_i, c_j) \in \mathbb{Z}^+$ between every pair $c_i, c_j \in C$ of the cities, and a bound $B \in \mathbb{Z}^+$

QUESTION : Is there a tour of all the cities having a total length of B or less?

(\mathbb{Z}^+ denotes +ve integers)

- TS belongs to class NP. Why?

- Now consider the complement of the Traveling Salesman Problem (TSC^c) : "Is it true that no tour of all the cities has length B or less?" TSC^c , although obviously solvable, does NOT belong to class NP? Why?

- We can therefore entertain the following hierarchy of intractability :

Intractable because unsolvable

Ex.: The word prob. for a phrase-structure grammar. The Halting Problem.

Intractable although solvable

Also called nondeterministically intractable. Ex.: TSC^c

Intractable due to exponential complexity

This is where the most difficult of the class NP problems belong.

Polynomial Time Reducibility

- Often a seemingly simpler problem can be "reduced" to a seemingly harder problem. (The harder problem will usually have more parameters. An instance of the simpler problem can be mapped to an instance of the harder prob. by ignoring some of the parameters of the latter.)
- We say that a problem Π_1 is reducible to a problem Π_2 if every instance of Π_1 can be transformed into an instance of Π_2 such that the former has a solution iff the latter has a solution.
- We have polynomial time reducibility if this transformation can be carried out in polynomial time.
- In class NP, every problem can be thus reduced to the problem of Satisfiability (SAT). That means, SAT must be one of the most difficult problems in NP.
- If SAT can be reduced to another problem Π , then Π must be at least as hard as SAT.
- The set of the most difficult problems within class NP is called **NP-Complete**.

