CSC-257
Theory Of Computation
(BSc CSIT, TU)

Ganesh Khatri
kh6ganesh@gmail.com

Reducibility

- Reducibility is a way of converging one problem into another in such a way that, a solution to the second problem can be used to solve the first one.
- Many complexity classes are defined using the concept of a reduction.
- A reduction is a transformation of one problem into another problem.
- It captures the informal notion of a problem being at least as difficult as another problem.
- For instance, if a problem X can be solved using an algorithm for Y, X is no more difficult than Y, and we say that X reduces to Y.
- There are many different type of reductions, based on the method of reduction, such as Cook reductions, Karp reductions and Levin reductions, and the bound on the complexity of reductions, such as polynomial-time reductions or log-space reductions

Reducibility

- The most commonly used reduction is a polynomial-time reduction.
- This means that the reduction process takes polynomial time.
- For example, the problem of squaring an integer can be reduced to the problem of multiplying two integers.
- This means an algorithm for multiplying two integers can be used to square an integer.
- Indeed, this can be done by giving the same input to both inputs of the multiplication algorithm.
- Thus we see that squaring is not more difficult than multiplication, since squaring can be reduced to multiplication.

Circuit Satisfiability

- **Cook's Theorem : SAT is a NP – Complete Problem**
  - To show that SAT is NP-complete, we have to show two properties as given by the definition of NP-complete problems. The first property i.e. SAT is in NP
  - Circuit satisfiability problem (SAT) is the question "Given a Boolean combinational circuit, is it satisfiable? i.e. does the circuit has assignment sequence of truth values that produces the output of the circuit as 1?"
  - Given the circuit satisfiability problem take a circuit x and a certificate y with the set of values that produce output 1, we can verify that whether the given certificate satisfies the circuit in polynomial time.
  - So we can say that circuit satisfiability problem is NP.
  - This claims that SAT is NP.
  - Now it is sufficient to show the second property holds for SAT.
  - The proof for the second property i.e. SAT is NP-hard is from a lemma that says "SAT is NP Hard". This completes the proof

Undecidability

- In computability theory, an undecidable problem is a decision problem for which it is impossible to construct a single algorithm that always leads to a correct "yes" or "no" answer, so, the problem is not decidable.
- An undecidable problem consists of a family of instances for which a particular yes/no answer is required, such that there is no computer program that, given any problem instance as input, terminates and outputs the required answer after a finite number of steps.
- More formally, an undecidable problem is a problem whose language is not a computable or decidable.
- In computability theory, the halting problem is a decision problem which can be stated as follows :
- "Given a description of a program and a finite input, decide whether the program finishes running or will run forever"

Undecidability

- Alan Turing proved in 1936 that a general algorithm running on a Turing machine that solves the halting problem for all possible program-input pairs necessarily cannot exist.
- Hence, the halting problem is undecidable.
- Another example is “Post’s Correspondence Problem(PCP)”

Halting Problem

- "Given a Turing Machine M and an input w, does M halts on w?"
- Algorithms may contain loops which may be infinite or finite in length. The amount of work done in an algorithm usually depends on data input.
- Algorithms may consists of various numbers of loops nested or in sequence.
- Thus, the halting problem asks the question; "Given a program and an input to the program, determine if the program will eventually stop when it is given that input."
- The question is simply whether the given program will ever halt on a particular input.

Halting Problem

- **Trial Solution :**
  - Just run the program with the given input.
  - If the program stops we know the program halts.
  - But if the program does not stop in reasonable amount of time, we can not conclude that it won't stop.
  - May be we did not wait long enough!
  - The halting problem is famous because it was one of the first problems proven algorithmically undecidable.
  - This means there is no algorithm which can be applied to any arbitrary program and input to decide whether the program stops when run with that input.
  - The Halting Problem is one of the simplest problems known to be unsolvable