Complexity

- Even when problems are decidable (computable)
  - They may not be feasible
  - Solutions required too much time or memory
- Time Complexity

**Definition 7.1**

Let $M$ be a deterministic Turing machine that halts on all inputs. The *running time* or *time complexity* of $M$ is the function $f : \mathcal{N} \rightarrow \mathcal{N}$, where $f(n)$ is the maximum number of steps that $M$ uses on any input of length $n$. If $f(n)$ is the running time of $M$, we say that $M$ runs in time $f(n)$ and that $M$ is an $f(n)$ time Turing machine. Customarily we use $n$ to represent the length of the input.
Big-O Notation

- Asymptotic analysis
  - Estimate the run time of an algorithm for large inputs
  - Only consider the highest order terms
    - Disregard coefficient of the term and lower order terms

**Definition 7.2**

Let $f$ and $g$ be functions $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$. Say that $f(n) = O(g(n))$ if positive integers $c$ and $n_0$ exist such that for every integer $n \geq n_0$,

$$f(n) \leq c g(n).$$

When $f(n) = O(g(n))$, we say that $g(n)$ is an **upper bound** for $f(n)$, or more precisely, that $g(n)$ is an **asymptotic upper bound** for $f(n)$, to emphasize that we are suppressing constant factors.
Analyzing Algorithms

- Given $A = \{ 0^k1^k \mid k \geq 0 \}$
- Low level description of TM $M_1$ that decides $A$

\[
M_1 = \text{“On input string } w:\n1. \text{ Scan across the tape and reject if a 0 is found to the right of a 1.} \\
2. \text{ Repeat if both 0s and 1s remain on the tape:} \\
3. \text{ Scan across the tape, crossing off a single 0 and a single 1.} \\
4. \text{ If 0s still remain after all the 1s have been crossed off, or if 1s still remain after all the 0s have been crossed off, reject. Otherwise, if neither 0s nor 1s remain on the tape, accept.”}
\]

- Complexity of this TM will be the number of steps for a given input length
  - Consider some stages separately
    - 1 - Scans for $0^*1^*$ form by moving head across tape: $n$ steps = $O(n)$
    - 2 and 3 – a scan for every pair of 0’s and 1’s: At most $n/2$ scans
      - Each scan is $O(n)$ steps: $n/2 * O(n) = O(n^2)$
    - 4 – Single scan to find remaining inputs: $n$ steps = $O(n)$
  - $O(n) + O(n^2) + O(n) = O(n^2)$
Time Complexity Class

**Definition 7.7**

Let $t: \mathbb{N} \to \mathbb{R}^+$ be a function. Define the *time complexity class*, \( \text{TIME}(t(n)) \), to be the collection of all languages that are decidable by an \( O(t(n)) \) time Turing machine.

- From previous example
  - \( A \in \text{TIME}(n^2) \) because \( M_1 \) decides \( A \) in time \( O(n^2) \)
Given a an $O(t(n))$ multitape TM

- The equivalent single tape will run in $O(t^2(n))$
Section 7.2 Class P

- Problems solved in polynomial time, $O(n^c)$
  - Generally considered to be small and easily computable for large inputs

- Polynomialsly equivalent
  - Two computational models are Polynomially equivalent if simulating one with the other only increases by polynomial time.
    - Ex. Multitape to singletape TM
**DEFINITION 7.12**

\[ P \] is the class of languages that are decidable in polynomial time on a deterministic single-tape Turing machine. In other words,

\[ P = \bigcup_{k} \text{TIME}(n^k). \]

- Important because
  - \( P \) is invariant for all models of computation
  - \( P \) roughly corresponds to the class of problems that are realistically solvable on a computer
Section 7.3 Class NP

- Not every problem is solvable in polynomial time
- We may be unsure if a language is class P
  - If we can acquire a solution we can try to verify it in polynomial time
  - Easier to verify, harder to determine existence

**Definition 7.18**

A **verifier** for a language $A$ is an algorithm $V$, where

$$A = \{w | V \text{ accepts } \langle w, c \rangle \text{ for some string } c \}.$$  

We measure the time of a verifier only in terms of the length of $w$, so a **polynomial time verifier** runs in polynomial time in the length of $w$. A language $A$ is **polynomially verifiable** if it has a polynomial time verifier.

- $c =$ certificate or proof
  - Additional information used to show membership in language $A$
NP Class

- Nondeterministic polynomial time
  - class of languages that have polynomial time verifiers
  - Solvable by using a nondeterministic TM in polynomial time

- class \( P \) (all problems solvable, deterministically, in polynomial time) is contained in NP (problems where solutions can be verified in polynomial time)
  - \( P = \) class of languages for which membership can be decided quickly
  - \( NP = \) class of languages for which membership can be verified quickly

**Definition 7.21**

\[
\text{NTIME}(t(n)) = \{ L \mid L \text{ is a language decided by an } O(t(n)) \text{ time nondeterministic Turing machine} \}.
\]