Definition 7.1. **Running Time or Time complexity**

- M is a deterministic TM that halts on all inputs
- $f_M: \mathbb{N} \rightarrow \mathbb{N}$ maps for machine M length $n$ into max # of steps M takes to solve any string of length $n$ for which it halts
  - Usually drop subscript
- We say that M is an $f(n)$ time TM

Definition 7.2 We say $f(n) = O(g(n))$ if

- $f, g$ functions $\mathbb{N} \rightarrow \mathbb{R}^+$
- there are positive ints $c$ and $n_0$ such that
- $f(n) \leq cg(n)$ for all $n > n_0$
- $g(n)$ said to be an **asymptotic upper bound**

Notes on "big O"

- "O" of polynomials = largest exponent
- $\log_a(n)$ differs by a constant value from $\log_b(n)$ for any $a, b$
  - Thus $O(\log_a(n)) = O(\log_b(n)) = O(\log(n))$
- **polynomial bounds** if $O(n^c)$
- **exponential bounds** if $O(2^{\delta n})$

"Little o" Def 7.5: $f(n) = o(g(n))$ if $\lim_{n \rightarrow \infty} (f(n)/g(n)) = 0$

- for any $c$, there is some $n_0$ such that $f(n) < cg(n_0)$ for $n > n_0$
- $f(n)$ is asymptotically **less** than $g(n)$
• (p. 279) **Time Complexity class:**
  • \( \text{TIME}(t(n)) = \text{set of all languages decidable by } O(t(n)) \text{ TM} \)
  • Time\( (n) \) called “Linear Time”

• Example: \( A = \{0^k1^k | k \geq 0\} \)
  • M1 with input \( w, |w| = n \)
    1. \( O(n) \) Scan tape & reject if 0 to right of a 1
    2. \( O(n) \) repetitions: Repeat if both 0s and 1s on tape
      3. \( O(n) \) each: Scan tape, crossing off one 0 and one 1
      4. \( O(n) \) If 0s remain after all 1s, or vv, reject. Else accept
  • Time = \( O(n) + O(n^2) + O(n) = O(n^2) \)
  • Thus \( A \) in \( \text{TIME}(n^2) \). Anything better?

• M2 with input \( w, |w| = n \)
  1. \( O(n) \) Scan tape & reject if 0 to right of a 1
  2. \( O(\log(n)) \) repetitions Repeat if both 0s and 1s on tape
     3. \( O(n) \) each: Scan tape to see if even or odd #s of 0s & 1s. If odd, reject
     4. \( O(n) \) each: scan tape, crossing off every other 0, then every other 1
     5. \( O(n) \) if no 0s or 1s, accept, else reject
  • Time = \( O(n) + O(\log(n)) \times O(n) + O(n) = O(n \log(n)) \)
  • Thus \( A \) now in \( \text{TIME}(n \log(n)) \). Anything better?
• 2-tape TM M3 can solve in $O(n)$ time!
  • M3 with input $w$ on tape 1, $|w|=n$
    1. $O(n)$ Scan tape & reject if 0 to right of a 1
    2. $O(n)$ scan tape 1 to 1$^{st}$ 1, copying all 0s to tape 2
    3. $O(n)$ each: Scan tape 1. For each 1, cross off a 0 from tape 2. If all 0s crossed off before all 1s, reject
    4. $O(n)$: If all 1s off tape1, & no 0s on tape2, accept, else reject
  • Thus A in TIME$_{2tape}(n)$
• Generalization: (Problem 7.49): any language decidable in $o(n\log(n))$ on single tape TM is regular
Theorem 2.8. Every $O(t(n))$ multi-tape TM has an equivalent $O(t^2(n))$ 1 tape TM

- Assume $M = k$-tape TM with $O(t(n))$ time
- Let $S = $ equivalent 1-tape machines
- $S$'s 1st step: initialize its 1 tape to store $k$ tapes
  - Use "#" to separate and "'" to show tape head
- For each of $M$'s $O(t(n))$ steps, $S$ performs
  - $O(t(n))$: scan tape to find current values under heads
  - $O(t(n))$: scan tape again to update each of $k$ tapes
    - If any of $M$'s tapes writes into blank area, shift rest of simulated tapes 1 cell right
- Total is $O(t(n)) \cdot O(t(n)) = O(t^2(n))$
(p. 283) Definition 7.9: **Running time of a NTM** (1-tape) decider is \( f:N \rightarrow N \) where \( f(n) \) is max # steps for an input of length \( n \) on any branch of computation tree.

- See Fig. 7.10 on p. 283

(p. 284) **Theorem 7.11.** Let \( t(n) \) be a function where \( t(n) > n \). Every \( t(n) \) NTM (1-tape) has equivalent \( 2^{O(t(n))} \) time deterministic 1-tape TM.

  - Proof: given input of length \( n \)
    - Each branch of NTM computation of length \( t(n) \)
    - If \( b = \text{max # of choices in each tree of computation} \)
      - Then # of leaves at most \( b^{t(n)} \)
    - TM simulator \( D \) (Theorem 3.16) uses 3-tapes and visits all choices at depth \( d \) before going to depth \( d+1 \)
    - Total # nodes in tree < \( 2X \) # leaves, so bound as \( O(b^{t(n)}) \)
    - Time from root to node is \( O(t(n)) \)
    - Running time of \( D \) is \( O(t(n)b^{t(n)}) = 2^{O(t(n))} \)
    - Simulating on 1-tape squares time: \( (2^{O(t(n))})^2 = 2^{O(t(n))} \)

- Sample problems:
  - \( O \) notation: 7.1
  - \( o \) notation: 7.2