Complexity I

- From past discussion, we know

$$\text{decidable} \rightarrow \text{computationally solvable}$$

- However, this does not mean it is solvable in practice
- The running time may be just too long

Example I

- $A = \{0^k 1^k \mid k \geq 0\}$
  What's the # steps by a 1-tape TM to process a string?
- Remember the procedure
  1. check if input is 0^*1^*
  2. repeat until no 0 or 1
     scan, cross off single 0 and 1
  3. if 0 or 1 remains, reject
     otherwise, accept
- How much time?
  Need to count # steps

Analysis I

- Worst-case analysis

  Longest time (i.e., largest # of steps) for all inputs
- Average-case analysis
- Usually it is easier to do worst-case analysis
- We use a function

$$f : N \rightarrow N$$

  to represent the number of steps

  N: natural number

  n: length of input, $f(n)$: number of steps

Big-O I

- A way to understand the running time of the algorithm when it is run on large inputs
- Consider

  $$f(n) = 6n^3 + 5$$

  We have

  $$n \to \infty, 6n^3 + 5 \approx 6n^3$$

- $O(f(n)) = O(n^3)$
  How about 6?

  $$6n^3 \text{ vs. } n^3$$
  $$6n^3 \text{ vs. } n^4$$

Big-O II

Only things involved with n are important

- Definition:

$$f(n) = O(g(n))$$

if

$$\exists c, n_0, \forall n \geq n_0, f(n) \leq cg(n).$$

Example I

- Consider

$$f(n) = 6n^3 + 5$$

We have

$$6n^3 + 5 \leq 7n^3 \text{ after } n \geq 2$$

That is, we choose

$$c = 7 \text{ and } n_0 = 2$$

Thus

$$f(n) = O(n^3)$$

Example II

- $f(n) = O(n^4)$ as

$$6n^3 + 5 \leq 7n^4, \text{ after } n \geq 2$$

- But $f(n) \neq O(n^2)$

$$6n^3 + 5 \leq cn^2$$

cannot always hold because we can choose large n such that

$$n^3 > cn^2$$

Example III

- Formally we have the following opposite statement of the definition:

$$\forall c, n_0, \exists n \geq n_0, f(n) > cg(n)$$

Example 7.4 I

- Consider

$$f(n) = 3n\log_2 n + 5n\log_2\log_2 n$$

- We prove

$$f(n) = O(n\log n)$$

- Note that we write

$$\log n \text{ instead of } \log_2 n$$

as we will show that the result holds for any base b for the log function

Example 7.4 II

- Proof: From

$$n \le 2^n, \forall n \ge 1,$$

we have

$$\log_2 n \le n$$

From this,

$$\log_2 \log_2 n \le \log_2 n$$

Therefore

$$f(n) \le 8n \log_2 n = 8n \log_2 b \log_b n, \forall n \ge 1$$

Example 7.4 III

by using

$$\frac{\log_2 n}{\log_2 b} = \log_b n$$

Other properties I

- We have

$$O(n^2) + O(n) = O(n^2)$$

- Formally,

$$f(n) = O(n^2), g(n) = O(n)$$
$$\Rightarrow f(n) + g(n) = O(n^2)$$

Other properties II

- Proof

$$\exists c_1, n_1, \forall n \geq n_1, f(n) \leq c_1 n^2$$
$$\exists c_2, n_2, \forall n \geq n_2, g(n) \leq c_2 n$$

Then

$$f(n) + g(n) \leq c_1 n^2 + c_2 n \leq (c_1 + c_2) n^2$$
$$\text{after } n \geq \max(n_1, n_2)$$

Thus we choose

$$c = c_1 + c_2 \text{ and } n_0 = \max(n_1, n_2)$$

Other properties III

- Definition:

$$f(n) = 2^{O(n)}$$

  if $\exists c, n_0$ such that

$$f(n) \leq 2^{cn}, \forall n \geq n_0$$

- $O(1)$: $\exists c, n_0$ such that

$$f(n) \leq c1, \forall n \geq n_0$$

  Thus

$$f(n) \leq \max\{f(1), \ldots, f(n_0 - 1), c\}, \forall n$$

Other properties IV

- That is,

$$f(n) \text{ always } \leq \text{ a constant}$$