1 Introduction

1.1 Big-O Notation

***def* 1.1.** *Let* $f, g : \mathbb{N} \to \mathbb{R}$

- *Write* $f = O(g)$ *if* $(\exists c > 0)(\exists N)(\forall n \geq N)(|f(n)| \leq cg(n))$.
- *Write* $f = \Omega(g)$ *if* $(\exists c > 0)(\exists N)(\forall n \geq N)(|f(n)| \geq cg(n))$.
- *Write* $f = \Theta(g)$ *if* $(\exists c_1, c_2 > 0)(\exists N)(\forall n \geq N)(c_1 g(n) \leq |f(n)| \leq c_2 g(n))$.
- *Write* $f = o(g)$ *if* $(\forall \epsilon > 0)(\exists N)(\forall n \geq N)(|f(n)| \leq \epsilon g(n))$.

Big-O Notation is the most commonly used one among the four.

***e.g.* 1.1.** $f(n) = 6n^4 - 3n^3 + 5 \Rightarrow f(n) = O(n^4)$

Proof. $|6n^4 - 3n^3 + 5| \leq 6n^4 + 3n^4 + 5n^4 = 13n^4$ $\square$

***e.g.* 1.2.** $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} = O(n^3)$

***exercises* 1.1.** *Write in big-O notation:*

1. $5 + 0.001n^3 + 0.25n$
2. $500n + 100n^{1.5} + 50n \log_{10} n$
3. $n^2 \log_2 n + n(\log_2 n)^2$
4. $3 \log_8 n + \log_2(\log_2 n)$

solution $O(n^3); O(n^1.5); O(n^2 \log n); O(\log n)$ $\square$

***prop* 1.1.**

- $f_1 = O(g_1), f_2 = O(g_2) \Rightarrow f_1 f_2 = O(g_1 g_2)$
- $f(O(g)) = O(fg)$
- $f_1 = O(g_1), f_2 = O(g_2) \Rightarrow f_1 + f_2 = O(max(g_1, g_2))$
- $f_1 = O(g), f_2 = O(g) \Rightarrow f_1 + f_2 = O(g)$
- $f = O(g) \Rightarrow kf = O(g)$

often encountered:

- *constant:* $O(1)$
  - $\sum\limits_{k=1}^{n} \frac{1}{k} = \ln n + O(1)$
  - $\sum\limits_{k=1}^{n} \frac{1}{k^2} = O(1)$
  - $\sum\limits_{k=1}^{n} \frac{1}{k \ln k} = \ln \ln n + O(1)$
- *double logarithmic:* $O(\log \log n)$
- *logarithmic:* $O(\log n)$
- *polylogarithmic:* $O((\log n)^c), c > 0$
- *linear:* $O(n)$
- *quasilinear:* $O(n \log^c n), O(n \log^{O(1)} n)$
- *quadratic:* $O(n^2)$

***def* 1.2.** $\omega(g), \theta(g)$

- $f = \omega(g)$ *if* $(\forall c > 0)(\exists N)(\forall n \geq N)(f(n) \geq cg(n))$
- $g = \theta(g)$ *(or equivalently* $f \sim g$*) if* $(\forall \epsilon > 0)(\exists N)(\forall n \geq N)(|f(n) - g(n)| < \epsilon g(n))$

1.2 Alphabets and Languages

***def* 1.3.** (*alphabet*) *An alphabet is a set of symbols*

- *Roman alphabet* : $a, b, c, d, \cdots, z$
- *binary alphabet* : $0, 1$

***def* 1.4.** *(string and its length)*

*A **string** (over an alphabet) is a finite sequence of symbols from the alphabet.* ***Empty string** is string of no symbols, denoted by* ε.

The set of all string is denoted by Σ^*. *Denote by* Σ^n *the set of all **string of length** n. So,* $\Sigma^* = \cup_{n \geq 0} \Sigma^n$.

Denote the length of a string w by $|w|$

***e.g.* 1.3.** $\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, \cdots\}, |\varepsilon| = 0, |0110| = 4$

***def* 1.5.** *(concatenation) Two strings over the same alphabet can be combined by the operation of **concatenation**. The concatenation of x and y is denoted by xy.*

***def* 1.6.** *(substring, suffix, prefix)*

A string v is a substring of w if $\exists$ strings x and y such that $w = xvy$.

*If $w = xv$ for some x, then v is a **suffix** of w.*

*If $w = vy$ for some y, then v is a **prefix** of w.*

***def* 1.7.** *("power") The string w^i is defined: $w^0 = \varepsilon, w^{i+1} = w^i w, i \in \mathbb{N}$*

***e.g.* 1.4.** $01^0 = \varepsilon, 01^1 = 01, 01^2 = 0101$

***def* 1.8.** *(reversal) The reversal of a string w,denoted by w^R,is the string "spelled backwards"*

A formal definition can be given by induction on length:

1. *If $w = \varepsilon, w^R = w = \varepsilon$*
2. *If $|w| = n + 1$,where $w = ua, a \in \Sigma$,then $w^R = au^R$*

***def* 1.9.** *(language) **Language** is a set of strings over an alphabet, That is, $L \subseteq \Sigma^*$.*

For example,$\emptyset, \Sigma^*, \Sigma$ are all languages

- $\sigma = \{0, 1\}$
- $Even = \{0, 10, 100, 110, \cdots, \}$
- $Odd = \{1, 11, 101, \cdots\}$
- $Prime = \{10, 11, 101, 111, \cdots\}$
- $Palindrome = \{w | w^R = w\} = \{\varepsilon, 0, 1, 00, 11, \cdots\}$

***def* 1.10.** *(complement, binary language operations)*

*Let L be a language. The **complement** of L, denoted by $\overline{L}$, is $\Sigma^* - \overline{L}$. So $\overline{\overline{L}} = L$.*

*Note that since L is a set, we can define **union**($\cup$), **interchapter***($\cap$) and difference.*

*The **concatenation** of L_1 and L_2 is defined by $L_1 L_2 = w \in \Sigma^*, w = xy, \exists x \in L_1, y \in L_2$*

1.3 Encoding of Problems

1.3.1 Examples

1. *(**Integer multiplication**) Given two non-negative integers x, y, compute xy.*
2. *(**Primality testing**) Given $n \in \mathbb{N}$, decide if n is a prime.*
3. *(**Hamiltonian cycle**) Given an undirected graph G, test if G has a Hamiltonian cycle.*

1.3.2 Analysis

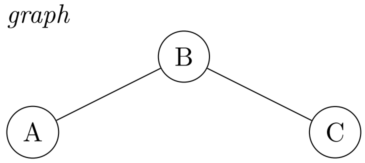
- ***Decision*** *problem: 2,3*
- ***Computation*** *problem: 1*

1.3.3 Conclusion

- *By* ***encoding****, decision problem is language. Any computation problem is a function from* Σ^* *to* Σ^**. Our course only concerns decision problem, namely language.*
- *By* ***preprocessing****, one can switch between encodings.*

e.g. **1.5.**

- *graph*

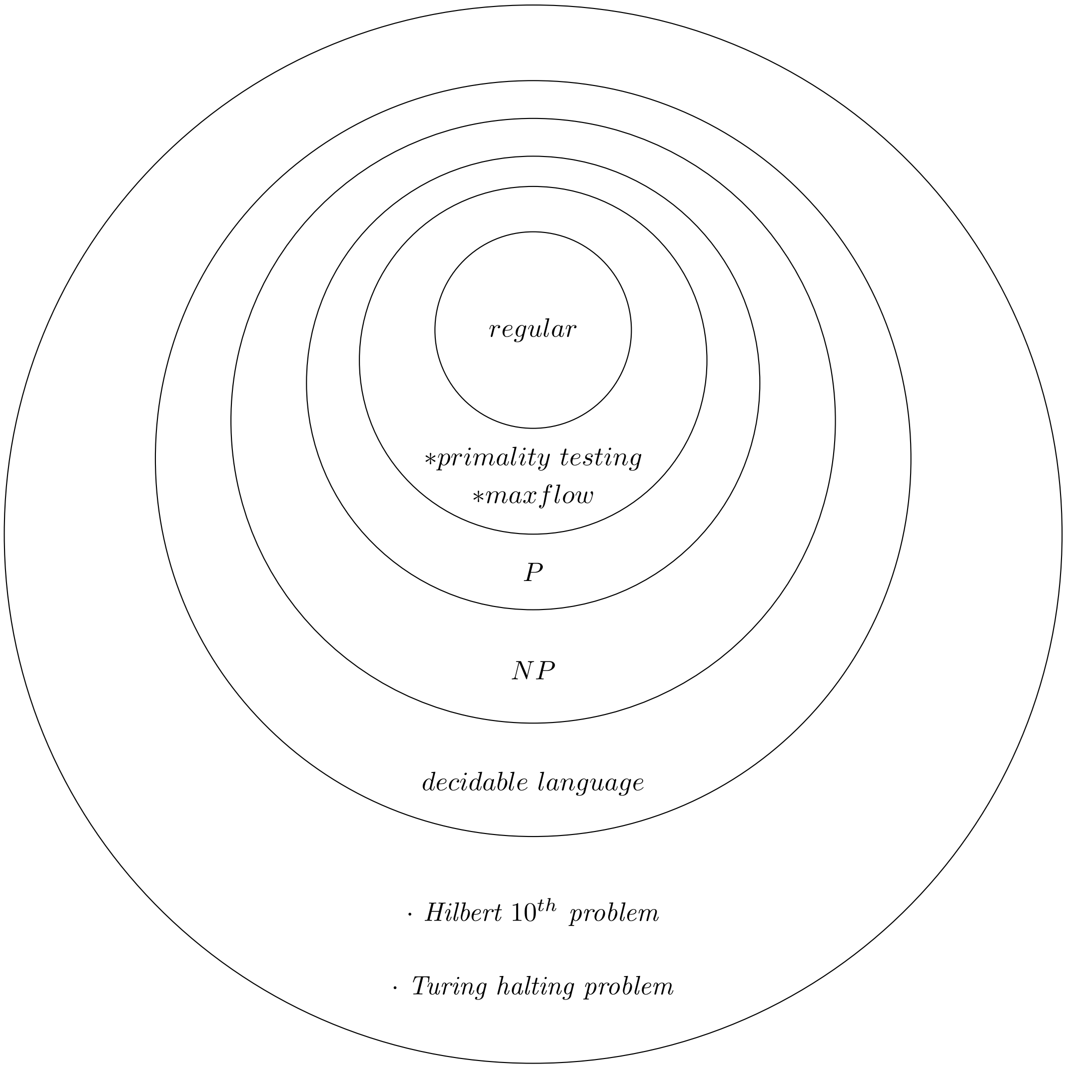


- *adjacency matrix*

$$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

- *adjacency list*

  (1,2),(1,3),$\cdots$



all languages $L \subseteq \Sigma^* = \{\varepsilon, 0, 1, 00, 01, \cdots\}$

remark.

NP*: problems that are efficiently verifiable*

P*: problems that are efficiently solvable, i.e. solvable in* $n^{O(1)}$ *time*

regular language*: problems that are solvable without memory,i.e. solvable by finite automation*