Text Processing

- Document (text) processing algorithms also highlight some important algorithmic design patterns
- Internet is made up of zettabytes of natural language data.
  - 1,000,000,000,000,000,000,000 (10^{21} or 2^{70})
    1024 2^{10} kilo
    1024^2 2^{20} mega
    1024^3 2^{30} giga
    1024^4 2^{40} tera
    1024^5 2^{50} peta
    1024^6 2^{60} exa
    1024^7 2^{70} zetta
    1024^8 2^{80} yotta
- At the heart of all algorithms for processing text are methods for dealing with character strings.
- Character strings can come from a wide variety of sources, including scientific, linguistic, and Internet applications.
- The following are examples of such strings
  - P = “CGTAAACTGCTTTAATCAAACGC” (DNA sub-sequence)
  - S = “http://www.wiley.com” (Website URL)
- Typical string processing operations involve breaking large strings into smaller strings.
- We use the term substring of an m-character string P to refer to a string of the form:
  - $P[i]P[i+1]P[i+2]\cdots P[j]$, for some i, j where $0 \leq i \leq j \leq m-1$,
  - The string formed by the characters in P from index i to index j, inclusive
  - The entire string can be considered a substring of itself where $i = 0$ and $j = m - 1$
- The notion of "characters" in a character string can be generalized

- Often referred to as "symbols"
- Use the symbol, Σ to denote the set of symbols, or **alphabet**, from which the symbols come
    - ASCII or UNICODE are such symbols sets
    - Nucleotides character set Σ = {A, T, C, G}
- The size of the alphabet Σ, denoted with $|\Sigma|$, is a fixed constant.

C++ String Class Operations

- C++ has two types of strings
    1. C-style string (nul-terminated)
        - Array of characters terminated by (ending in) the value 0 ('\0')
    2. Standard Template Library (STL) string class.
        - Supports many string operations
- A few of the string operations supported by the STL
    - In the following, let S denote the STL string object on which the operation is being performed, and let Q denote another STL string object or a C-style string.
    - size(): Return the number of characters, n, of S.
    - empty(): Return true if the string is empty and false otherwise.
    - operator[i]: Return the character at index i of S, (no array bounds checking).
    - at(i): Return the character at index i of S. An out of range exception is thrown if i is out of bounds.
    - insert(i,Q): Insert string Q prior to index i in S and return a reference to the result.
    - append(Q): Append string Q to the end of S and return a reference to the result.
    - erase(i,m): Remove m characters starting at index i and return a reference to the result.
    - substr(i,m): Return the substring of S of length m starting at index i.

  - find(Q): If Q is a substring of S, return the index of the beginning of the first occurrence of Q in S, else return n, the length of S.
  - c_str(): Return a C-style string containing the contents of S
- By default, a string is initialized to the empty string
- A string may be initialized from another STL string or from a C-style string but cannot be initialized by a char
- STL strings also support functions that return both forward and backward iterators
- The STL string class also supports assignment of one string to another.
- Provides overloaded relational operators, such as ==, <, >=, which are performed lexicographically.
- Overloads '+' operator so an be concatenated
- Can append one string to another using +=.
- Has a function getline(*in*,S) that reads an entire line of input from the input stream in and assigns it to the string S.
- The STL string class is actually a special case of a more general templated class, called *basic_string<T>*, which supports all the string operations but allows its elements to be of an arbitrary type, T, not just char
- The STL string is just a short way of saying basic_string<char>.
- A “string of integers” could be defined as *basic_string<int>*.
- Consider the following series of operations, which are performed on the string S = “abcdefghijklmnop”:

Operation	*Output*
S.size()	16
S.at(5)	'f'
S[5]	'f'
S + "qrs"	"abcdefghijklmnopqrs"
S == "abcdefghijklmnop"	**true**
S.find("ghi")	6
S.substr(4,6)	"efghij"
S.erase(4,6)	"abcdklmnop"
S.insert(1,"xxx")	"axxxbcdklmnop"
S += "xy"	"axxxbcdklmnopxy"
S.append("z")	"axxxbcdklmnopxyz"