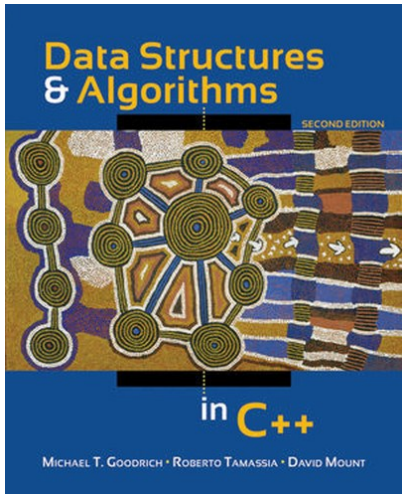
Data Structure

Array List & Linked List

Shin Hong

3 Mar 2023



DS&A
- Sec. 3.1. Using Arrays

Foundation of Computer Science http://infolab.stanford.edu/~ullman/focs.html
- Ch. 6. The List Data Model

Motivation

- You are writing a program that receives an arbitrary number of data items.
- How to store these data items in the memory?

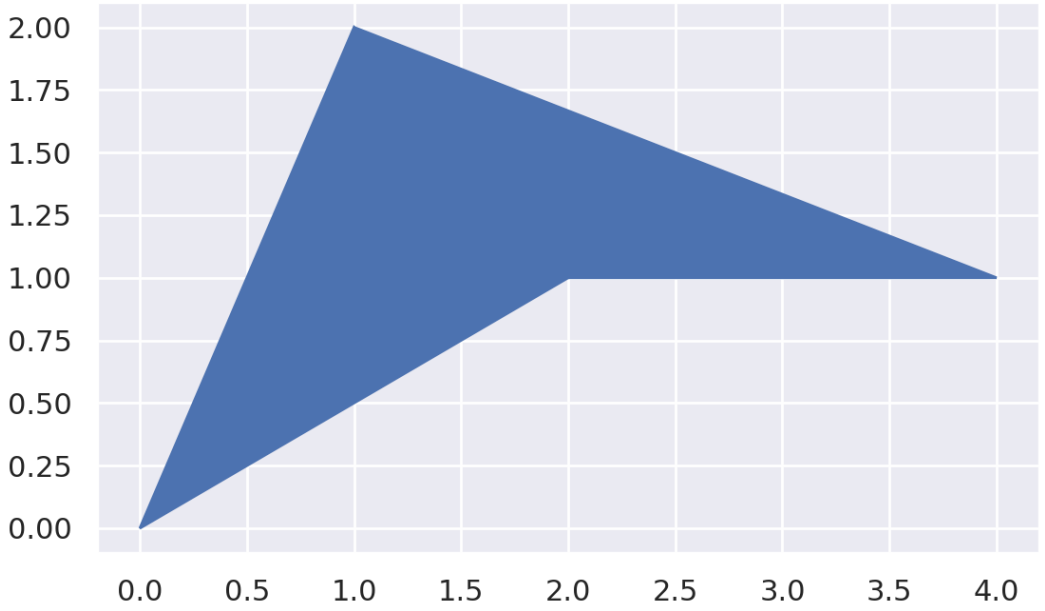
List

- A list is a finite sequence of zero or more elements
  - a list is a list of a type *T* if all its elements belong to *T*
  - a list is written with its elements separated by commas and enclosed in parentheses: $(a_1, a_2, \ldots, a_n)$
    - we say that element a_i occurs at position i

- Examples
  - (2, 3, 5, 7, 11, 13, 17, 19)
  - (helium, neon, argon, krypton, xenon, radon)
  - (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
  - A text document is a list of strings, and a string is a list of characters

Example: 2-D Polygon

- A list of points such that the first and the last are the same
- A point is a list of two real number
- Ex. ((0,0), (2, 1), (4, 1), (1, 2), (0,0))



Attributes of List

- The length of a list of the number of occurrences of elements on the list
  - the empty list a list of length 0
  - the length counts positions, not distinct symbols

- A non-empty list has a head and a tail
  - head: first element
  - tail: the remainder list excluding the first element
  - ex. (helium, neon, argon, krypton, xenon, radon)
    - head: helium
    - tail: (neon, argon, krypton, xenon, radon)

List Operations

- insertion
- deletion
- lookup
- concatenation
- sorting
- merging

Insertion, Deletion and Concatenation

- Inset an element x onto a list L
  - add x after the last element
  - add one more occurrence of x

- Delete an occurrence of x from L
  - need to specify which occurrence to delete
    - e.g., delete first occurrence, delete all occurrences, etc.

- Concatenate two lists L and M by forming the list that begins with the elements of L and continues with the elements of M

List Implementation

- Array list
  - use an array to store elements
- Linked list
  - use a chain of element-pointer pair (i.e., node) to store elements
- Two types of lists are compatible as these offer the same operations

Array List

- An array refers to a finite list of consecutively arranged elements of a certain type
  - an element can be immediately retrieved by its index

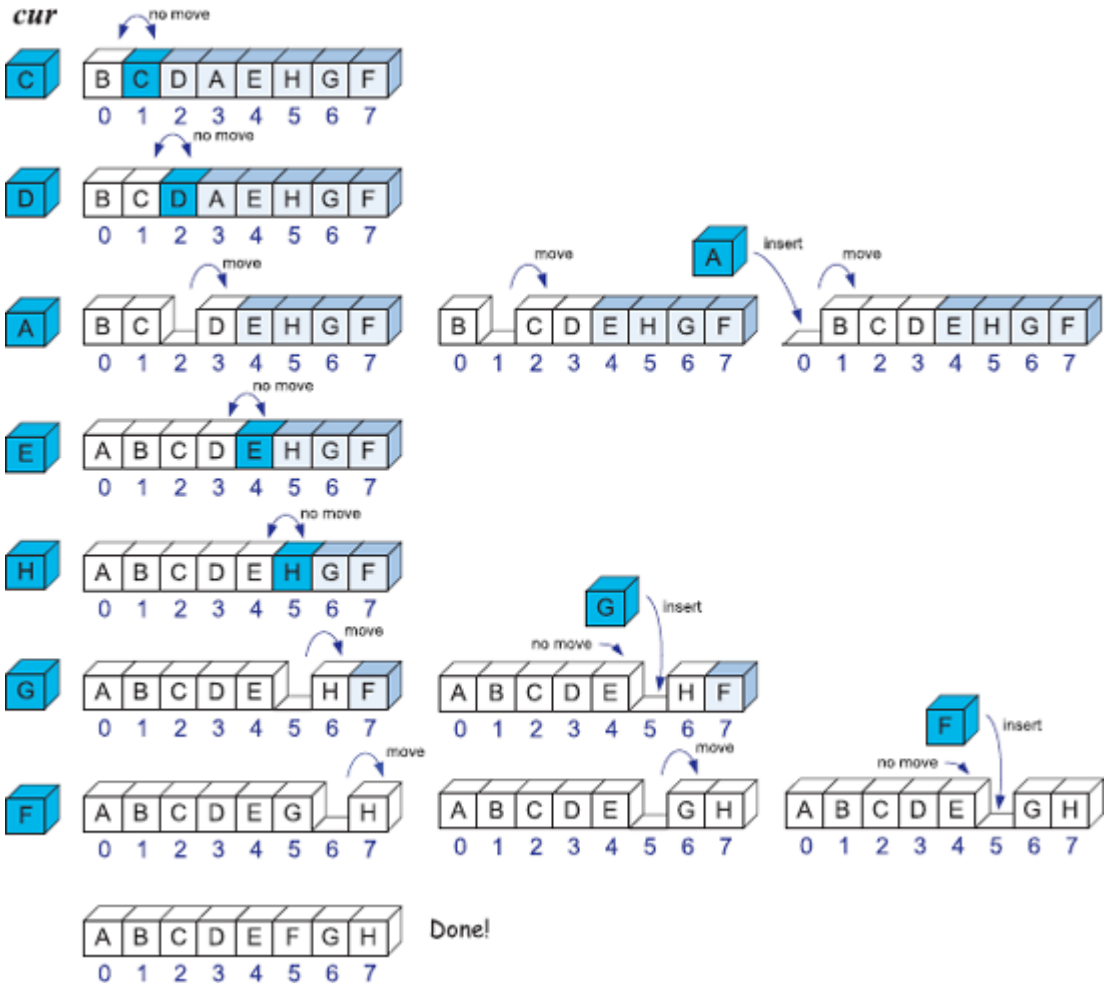
- An array list uses an array as a container to store elements
  - fixed-length list
  - variable-length list

Ex. Game Entry List

- An entry is a pair of a name and a score
- A game entry list holds high scorer up to *N* elements in descending order of their scores
  - when a new entry is added, an entry with the lowest score must be dropped

Insertion Sort (1/2)

- Make first i elements sorted, and then add (i+1)-th element to the sorted list



Insertion Sort (1/2)

Algorithm InsertionSort(A):
  Input: An array A of n comparable elements
  Output: The array A with elements rearranged in nondecreasing order

```
for i ← 1 to n − 1 do
    cur ← A[i]
    j ← i − 1
    while j ≥ 0 and A[j] > cur do
        A[j + 1] ← A[j]
        j ← j − 1
    A[j + 1] ← cur
```

Sublist and Subsequence

- A sublist of a list $L = (a_1, a_2, \ldots\ a_n)$ is a list formed by starting at a position i and taking all the elements up to a later position j $(a_i, a_{i+1}, \ldots\ a_j)$ for $1 \leq i \leq j \leq n$, or ϵ
  - a sublist is sometime called as substring
  - prefixes and suffixes are sublists

- A subsequence is a list $L = (a_1, a_2, \ldots\ a_n)$ formed by selecting some elements while keeping the same order, $(a_{k_1}, a_{k_2}, \ldots\ a_{k_m})$ where $1 \leq m \leq n$ and $k_j < k_{j+1}$ for $1 \leq j < m$ or ϵ

- E.g., Given list (a, b, c)
  (a,b) is a sublist, but (a,c) is not a sublist;
  (a,c) is a subsequence where m=2 and $a_{k_1} = 1$ and $a_{k_2} = 1$