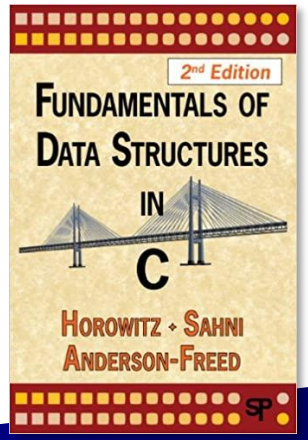
Data Structure

Stack & Backtracking

Shin Hong

Mar 31, 2020

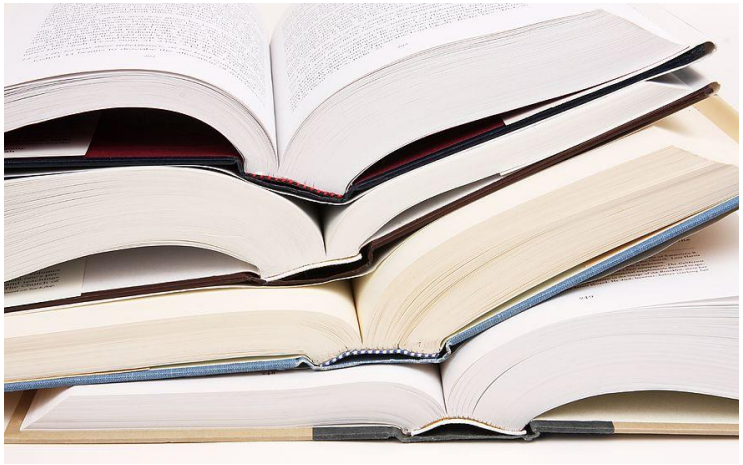
Chapter 3.
Stack and Queue

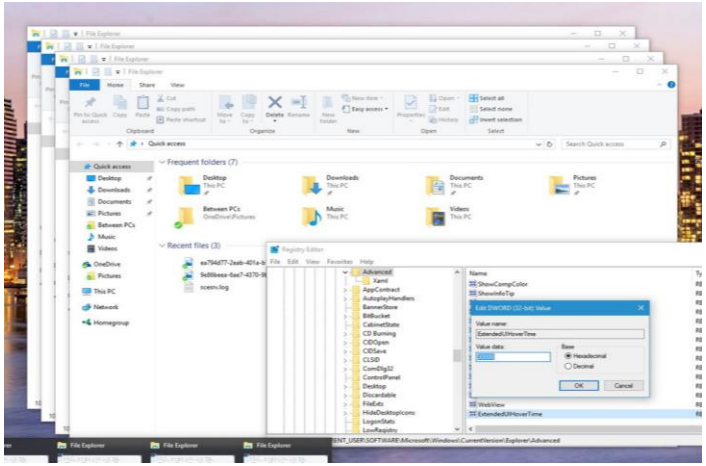


Stack

- A stack is a list where insertion and deletion is made only at one side of the end (top)
  - a stack is also called as LIFO (Last-In-First-Out)

- A stack is useful for storing a temporal state of a search on a hierarchical structure



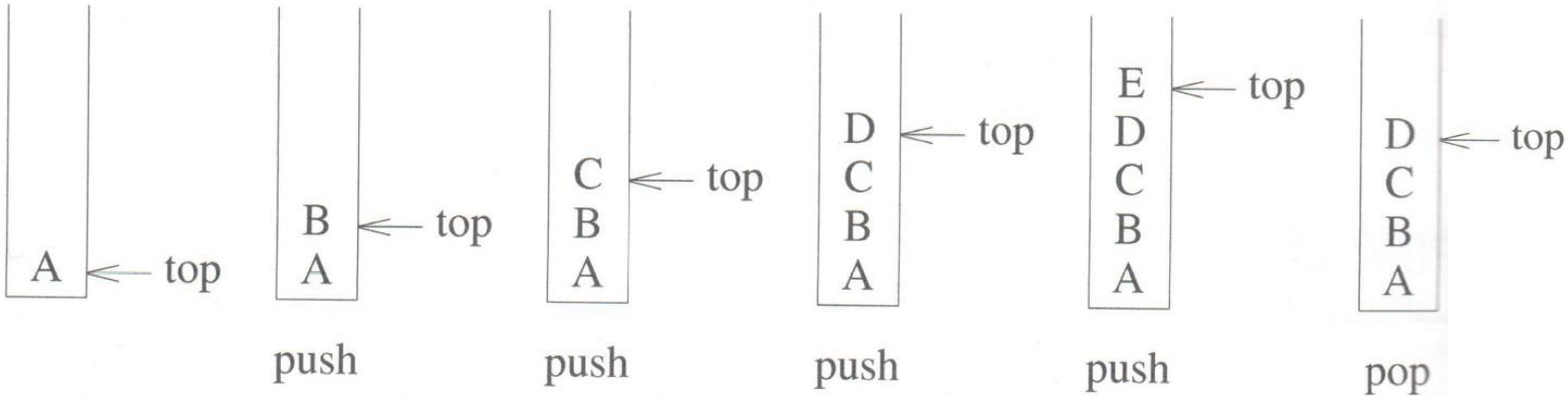


Stack Abstract Data Type

- Structure
  - **buffer**: an array to hold elements
  - **capacity**: the capacity of the buffer array
  - **top**: an index of the array to place a next element if the buffer is not full, or the capacity of the buffer

- Operations
  - **push(e)**: insert a new element **e** to the stack if the stack is not full
  - **pop()**: return the most recently inserted element if the stack is not empty
  - **isEmpty()** : return whether the stack has at least one element or not
  - **isFull()** : return whether the stack is full or not

Example


A
top
B
A
top
push
C
B
A
top
push
D
C
B
A
top
push
E
D
C
B
A
top
push
D
C
B
A
top
pop

Implementation

- Stack for integers
  - see https://github.com/hongshin/DataStructure/tree/stack/ver1

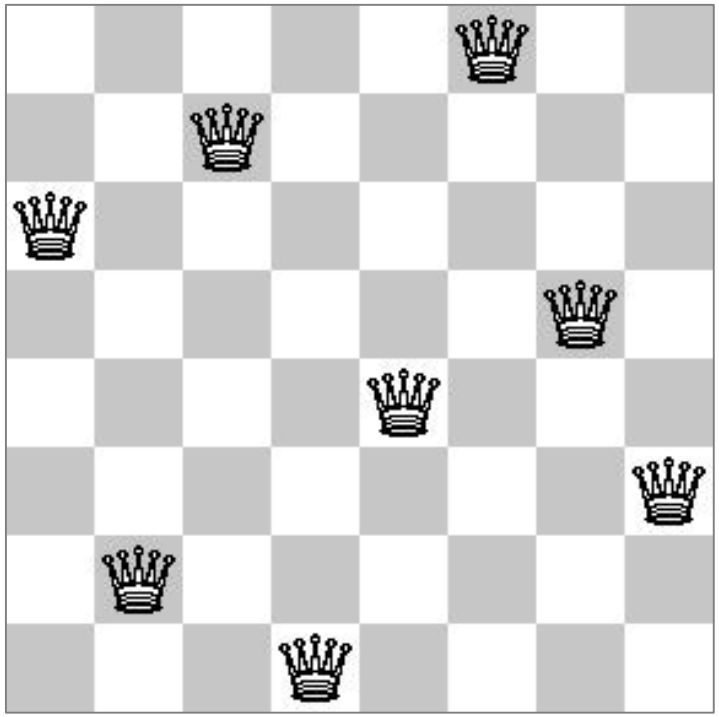
- How to construct a stack for all element types?
  - see https://github.com/hongshin/DataStructure/tree/stack/ver2

Backtracking

- There is a problem whose solution is a combination of (small) decisions
- A backtracking is a strategy to enumerate all possible solutions by recursively exploring all decision sequences
  - E.g., breaking a dial lock
- A stack is useful to represent the current status of the solution (a sequence of decisions) in backtracking

Ex. N Queen Problem

- Find a placement of N queens on a checkboard such that they do not conflict with each other
  - Two queens cannot stand together if they are on the same vertical / horizontal / diagonal line



Backtracking Queen Placement