Stacks

Subhabrata Samajder

IIIT, Delhi

Summer Semester,

19th May, 2022

Stacks and Queues

- Stacks and Queues are dynamic sets where the DELETE operation is prespecified.
  - Stack: Last-in, first-out (LIFO) or first-in, last-out (FILO).
  - Queue: First-in, first-out (FIFO) or last-in, last-out (LILO).

Stacks and Queues

- Stacks and Queues are dynamic sets where the DELETE operation is prespecified.
  - Stack: Last-in, first-out (LIFO) or first-in, last-out (FILO).
  - Queue: First-in, first-out (FIFO) or last-in, last-out (LILO).
- There are several efficient ways to implement stacks and queues.
- Here we will use arrays.

Stack

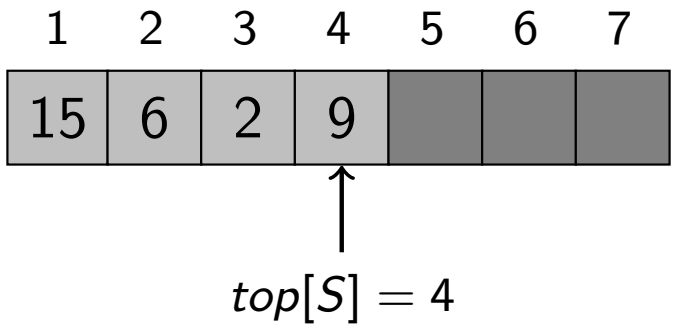
Stack

- The Insert is called Push.
- The Delete operation is called Pop.
  - **Note:** Pop does not take an element as argument,

Stack

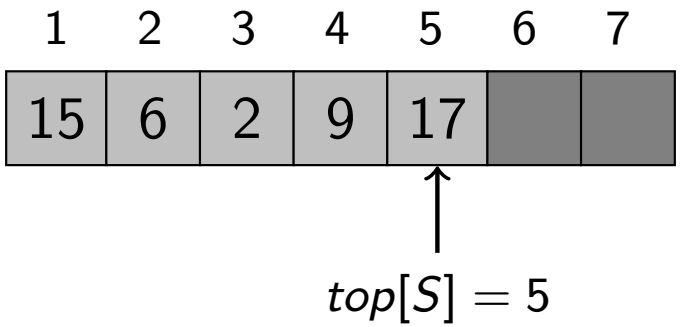
- The INSERT is called PUSH.
- The DELETE operation is called POP.
  - **Note:** POP does not take an element as argument,
- An array $S[1 \ldots n]$ denotes a stack of at most n elements.
- $top[S]$: Points to the most recently inserted element.
- The stack consists of elements $S[1 \ldots top[S]]$, where
  - $S[1]$ is the element at the bottom of the stack and
  - $S[top[S]]$ is the element at the top.
- **Empty Stack:** $top[S] = 0$.
- **Stack Underflow:** Empty stack is popped.
- **Stack Overflow:** $top[S] > n$.

Stack: An Example



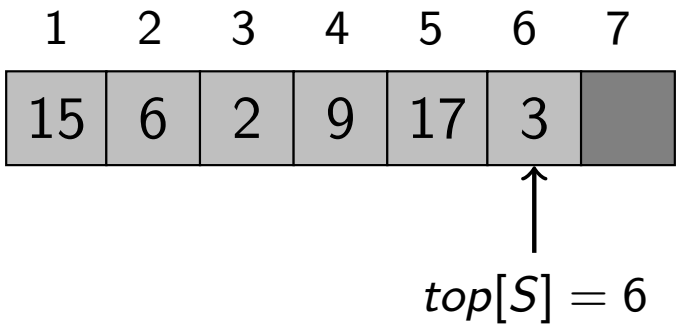
Stack: An Example

PUSH$(S, 17)$:



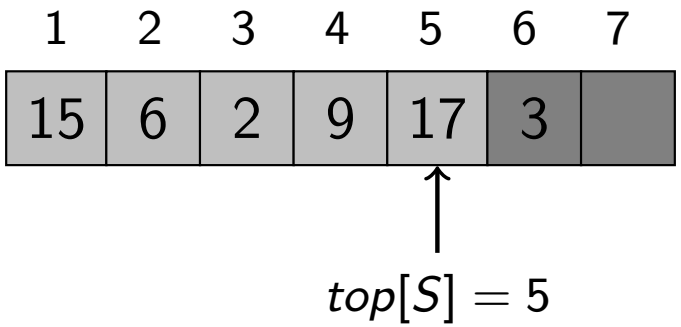
Stack: An Example

PUSH$(S, 3)$:



Stack: An Example

POP(S):



PUSH and POP

```
STACK-EMPTY(S)
Begin
   If (top[S] = 0)
      return TRUE;
   Else
      return FALSE;
End
```

Complexity: $\mathcal{O}(1)$

PUSH and POP

```
STACK-EMPTY(S)
Begin
  If (top[S] = 0)
    return TRUE;
  Else
    return FALSE;
End
```

Complexity: $\mathcal{O}(1)$

```
PUSH(S, x)
Begin
  If (top[S] = n)
    error "overflow";
  Else
    top[S] ← top[S] + 1;
    S[top[S]] ← x;
End
```

Complexity: $\mathcal{O}(1)$

PUSH and POP

```
STACK-EMPTY(S)
Begin
  If (top[S] = 0)
    return TRUE;
  Else
    return FALSE;
End
```

Complexity: $\mathcal{O}(1)$

```
PUSH(S, x)
Begin
  If (top[S] = n)
    error "overflow";
  Else
    top[S] ← top[S] + 1;
    S[top[S]] ← x;
End
```

Complexity: $\mathcal{O}(1)$

```
POP(S)
Begin
  If (STACKEMPTY(S))
    error "underflow";
  Else
    top[S] ← top[S] − 1;
    return S[top[S]];
End
```

Complexity: $\mathcal{O}(1)$

Books Consulted

1. Chapter 10.1 & 10.2 of *Introduction to Algorithms* by Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein.

Thank You for your kind attention!

Questions!!