The ADT Stack and its Implementations

Friday, October 18, 2019 5:08 PM

- A "stack" is a sequence of elements of the same type

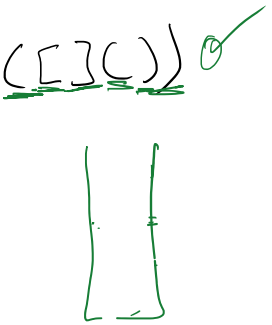
$S_a = \langle a, r, p, l, q \rangle$ (top)

- Operations
  - top(S_a) = 'q'
  - push(S_a, x) = $\langle a, r, p, l, q, x \rangle$
  - pop(S_a) = $\langle a, r, p, l \rangle$

- Example application:

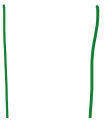
"(([]))" good "(((" bad "([)]" bad

read string character by character 'c'
  if 'c' is an opening symbol
    push 'c' into stack
  if 'c' is a closing symbol
    Compare c to top of stack.
    if c does not matches top of stack
      error!!
    else // c matches top of stack
      pop the stack
  if stack is empty SUCCESS!

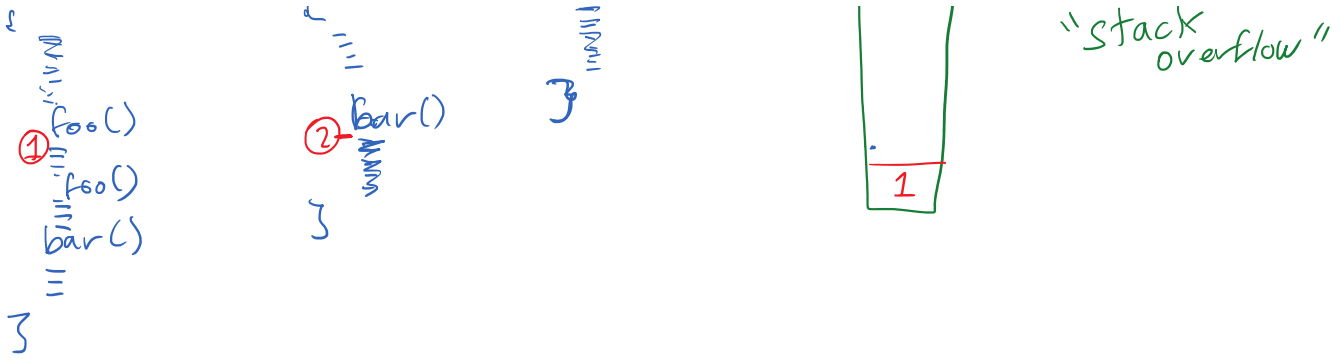


Application:

run_stack
call_stack

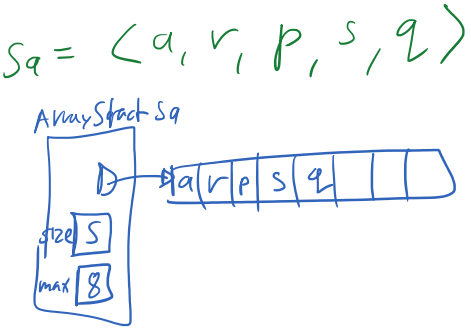


"stack overflow"



The ArrayStack Data Structure.

top = last
return m_data[m_size-1];



push = insert_back:

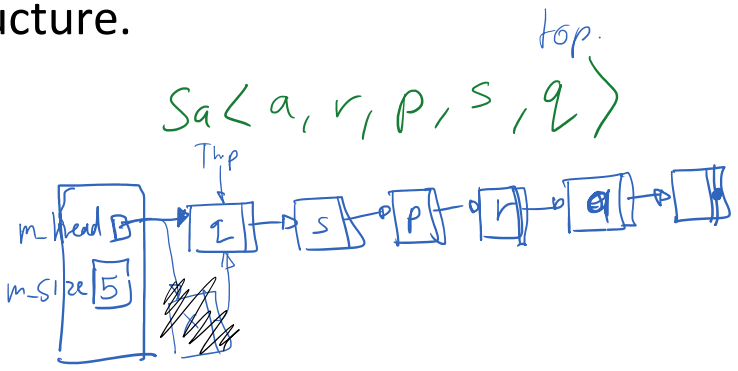
m_data[m_size] = x
m_size++

pop = remove_back

m_size--;

The LinkedStack Data Structure.

top = front
return m_head->m_data;



push = insert_front:

tmp = new LLNode(x, m_head)
m_head = tmp;
m_size++

pop = remove_front:

tmp = m_head->m_next;

```
delete m_head;
m_head = tmp;
m_size--;
```