

# 04-A

栈与队列

栈接口与实现

邓俊辉

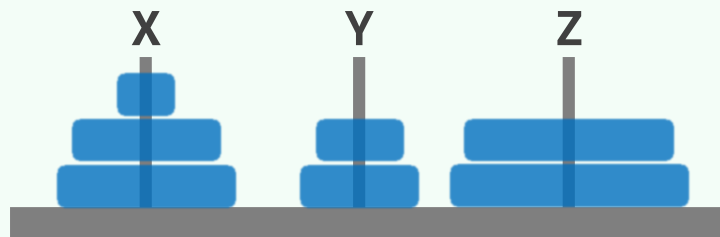
deng@tsinghua.edu.cn

陛下用群臣，如积薪耳，后来者居上

# 操作与接口

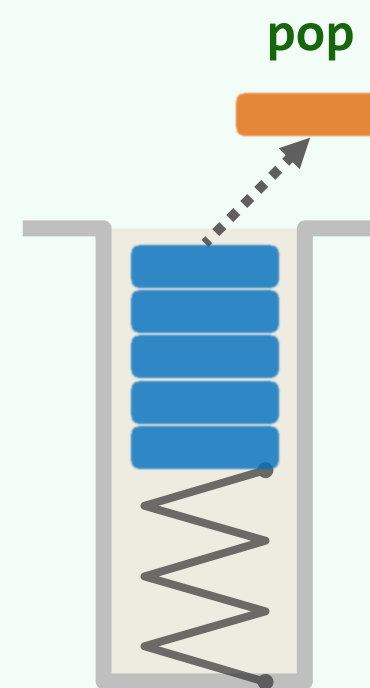
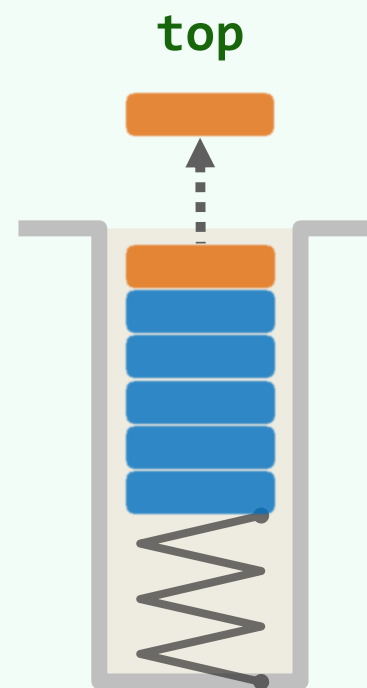
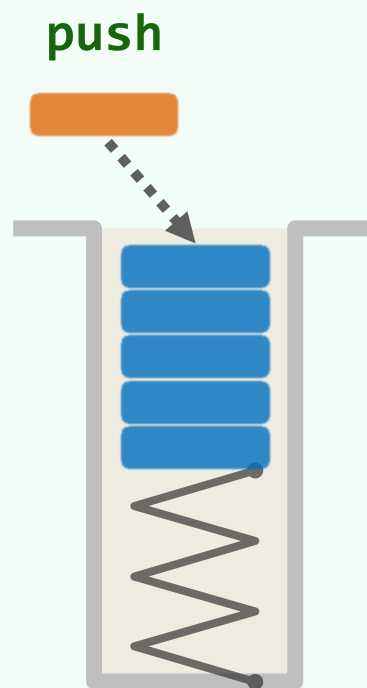
## ❖ 栈 (stack) 是受限的序列

- 只能在栈顶 (top) 插入和删除
- 栈底 (bottom) 为盲端



## ❖ 基本接口

- size() / empty()
- push() 入栈
- pop() 出栈
- top() 查顶



## ❖ 后进先出 (LIFO)

先进后出 (FILO)

## ❖ 扩展接口: getMax()...

实例

操作	输出	栈 (左侧栈顶)
Stack()		
empty()	true	
push(5)		5
push(3)		3 5
pop()	3	5
push(7)		7 5
push(3)		3 7 5
top()	3	3 7 5
empty()	false	3 7 5

操作	输出	栈 (左侧栈顶)
push(11)		11 3 7 5
size()	4	11 3 7 5
push(6)		6 11 3 7 5
empty()	false	6 11 3 7 5
push(7)		7 6 11 3 7 5
pop()	7	6 11 3 7 5
pop()	6	11 3 7 5
top()	11	11 3 7 5
size()	4	11 3 7 5

实现：既然属于序列的特例，故可直接基于向量或列表**派生**

```
template <typename T> class Stack: public Vector<T> {
```

```
public: //原有接口一概沿用
```

```
void push( T const & e ) { insert( e ); } //入栈
```

```
T pop() { return remove( size() - 1 ); } //出栈
```

```
T & top() { return (*this)[ size() - 1 ]; } //取顶
```

```
}; //以向量首/末端为栈底/顶——颠倒过来呢？
```

❖ 确认：如此实现的栈各接口，均只需 $O(1)$ 时间

❖ 课后：基于列表，派生定义栈模板类；你所实现的栈接口，效率如何？