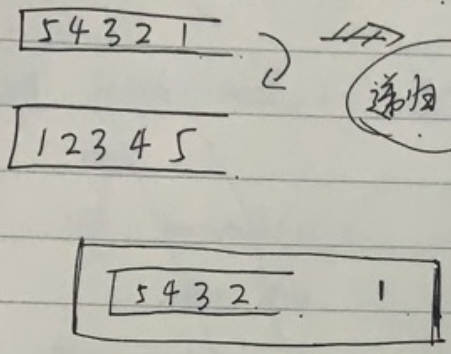
Date

如何使用递归函数和栈操作逆序一个栈



因为有递归栈，可以存储东西

递归函数的本质：输入为A

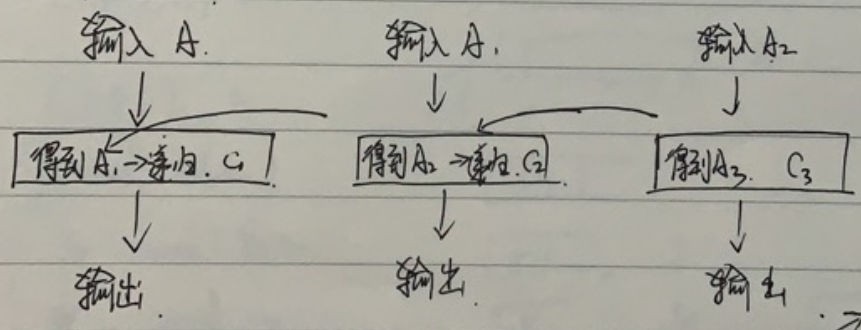
栈的某层（第i层栈）[存放东西]

进入来第i层栈 [存放东西]

返回一个东西 A_1

返回一个东西 A_2

不是返回，递归：



$f(5432)+1 \to 12345$

$f(543)+2 \to 2345$

$f(54)+3 \to 345$

$f(5)+4 \to 45$

$f(\square)+5 \to 5$

$\square$

$f(4321)+5$ → 似乎行，但分解不了

$f(321)+4$

$f(21)+3$

$f(1)+2$

$f(\square)+1$

$\square+1$

```
def f(stack):
    def rotate(stack):
        if len(stack) == 0:
            return None
        top = stack.pop()
        rotate(stack)
```

```
def f(stack, item):
    if stack == []:
        stack.append(item)
        return None
    top = stack.pop()
    f(stack, top)
```

```
def f(stack, item):
    if len(stack) >= 1:
        top = stack.pop()
        f(
```

```
def f(stack, item, top=None):
    if len(stak) == 0:
        stack.append(item)
    top = stack.pop()
    f(stack, item, top)
    stack.append(top)
```

item=None.

top=None.

Date

def f(stack): 去掉最后一个.

if len(stack)==0:

stack.append(item)

return None

top=stack.pop()

f(stack, top)

g(stack, item, None)

f([5,4,3,2,1])

↓

f([5432.] 1) → [2345.1] →

↓

[543] 2 → [2345] (5 4 3 2

↓

[54] 3 → [345]

↓

[5] 4 → [45]

↓

f([] 5) → [5]

[54] → [45] 3

↓

[5] 4 → [45]

f([] 5) → [5]

Date . .

5 4 3 2 1

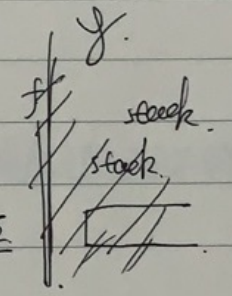
f(5432 , 1 , 2 , 543)

item top stack

stack.

item top stack.

① None. 1 5432



② 1 2 543 → 12345.

③ 2. 3 54 → 2345

④ 3 4. 5 → 345

⑤ 4 5 → 45

⑥ 5. 5.

item top

y([3,4,5], 2, None)

Date . .

	item.	top	stack.	stack.
y([3,4,5],2,None)2.		5	[3,4]	[2,3,4,5]
y([3,4],2,5)2.		4	[3]	[2,3,4]
y([3],2,4)2.		3	[]	[2,3]
y([],2,3)2.		3	[2]	↑

书中的写法

递归函数一：将栈中stack的栈底元素返回并移除

```
getAndRemoveLastElement(stack):
    result = stack.pop()
    if stack == []:
        return result
    else:
        last = getAnd——(stack)
        stack.push(result)
        return last
```

[123]

	stack.	result.	stack.	last.	
y([123])	[123]	3	[12]	1	[123]
y([12])	[12]	2	[1]	1	[12] ↑
y([1])	[1]	1	[]	↑	

Date

递归函数：逆序一个栈

reverse (stack):

  if stack==[]:

    return None

  i = f——(stack)

  reverse(stack)

  stack.push(i)

[5 4 3 2 1]

[4 3 2 1]

[1 2 3 4 5]

$O(n^2)$