Approaches to reverse 2-13-23

A: In Place A: In place
List: 1 2 . . . . . . n

1.) Move 1 item at a time
2.) Swapping

$i \rightleftarrows n-1-i$

$0 \rightleftarrows n-1$ $1 \rightleftarrows n-1$ $0 \to \frac{n-1}{2}$

	from	to
A1 Beginning	$0 \to n-1$, $0 \to n-2$, $0 \to n-3$	$0 \to i$, $i = n-1 \searrow 1$, $i--$
A1 Beginning	$1 \to 0$, $2 \to 0$ - - - $n-1 \to 0$	$i \to 0$, $i = 1 \nearrow n-1$
A1 End	$n-1 \to 0$, $n-1 \to 1$, $n-1 \to$	$n-1 \to i$, $i = 0 \nearrow n-2$
A1 End	$n-2 \to n-1$, $n-3 \to n-1$, $0 \to n-1$	$i \to n-1$, $n-2 \searrow 0$

$\approx (n-1)(1 \text{ remove} + 1 \text{ add}) = 2 \times (n-1)$
Analysis for A1

$\approx \frac{n}{2}(2 \text{ remove} + 2 \text{ add}) = 2n$ or $2(n-1)$
Analysis for A2

B: Using Temp collection

List 1 2 ..... n

Temp n n-1 1

Cannot use temp collection with reference as parem.

```
public ListInterface reverse(ListInterface list)
{
    ListInterface Temp = new... ();
    move from list to temp
    return temp;
}
```

B1 { i n-1 ↘ 1
i ↓ temp size() }

B2 { i i = 0 ↗ n-1
i ↓ 0 }

n (1 get + 1 add) = $2n$ B1, B2 Analysis

Stack ADT (LiFo Structure)

Linear Collection w/ all approaches

boolean isEmpty()

Object peek() throw...

void push() throw...

Object pop() throw...

void popAll()

with list implementation...

peek ≡ coll.get(coll.size() - 1);
OR peek ≡ coll.get(0);

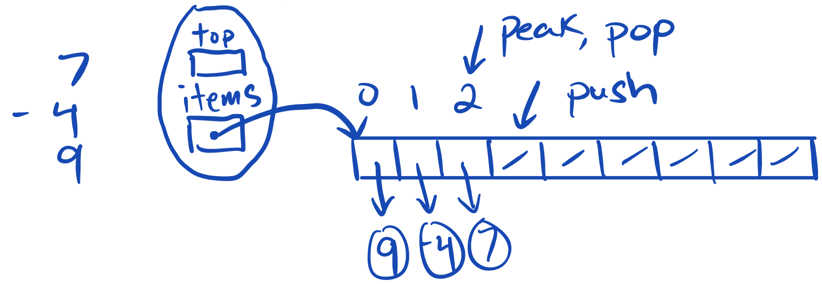
push ≡ coll.add(coll.size());
OR push ≡ coll.add(0);

pop ≡ coll.remove(coll.size() - 1);
OR pop ≡ coll.remove(0);

Stack ADT Implementations:

Array Based

0. StackRA



public stackRA()

{

items = new Object[3];

top = -1;

}

isEmpty → return top == -1;

peek → if non empty return items[top];

if empty, throw exception

push (object item)

{

if (top == items.length - 1)

{

resize(); //resize from old lab

}

items[++top] = item;

}

Object Pop ()

{ if non empty

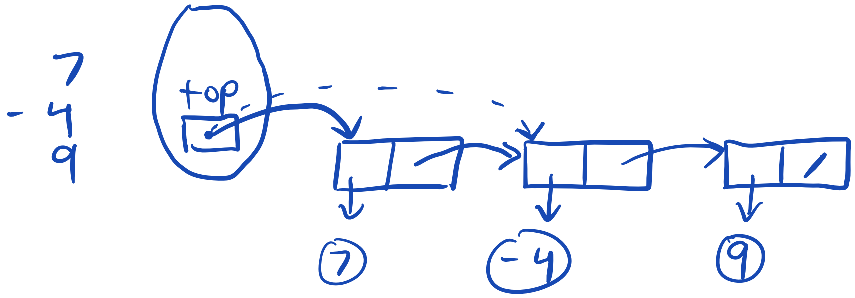
result = items[top]

items[top--] = null;

} return result;

Array Based

0. StackSLS



isEmpty ⟶ return top == null;

peek ⟶ if non-empty, return top.getItem();

push ⟶ top = new Node(item, top);

Pop ⟶ if non-empty, result = top.getItem();
top = top.getNext();

StackRA

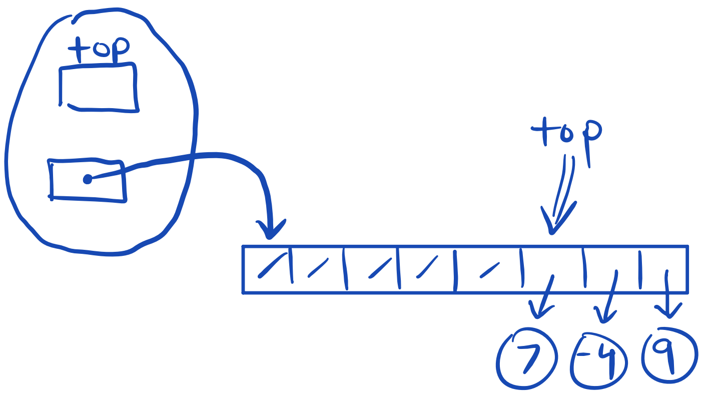
pros: simple, no additional supporting

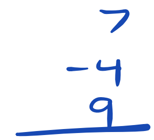
cons: copy upon resizing
pre-allocation of mem

StackSLS

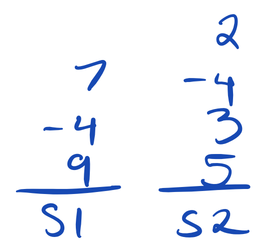
pros: on demand allocation,
still simple

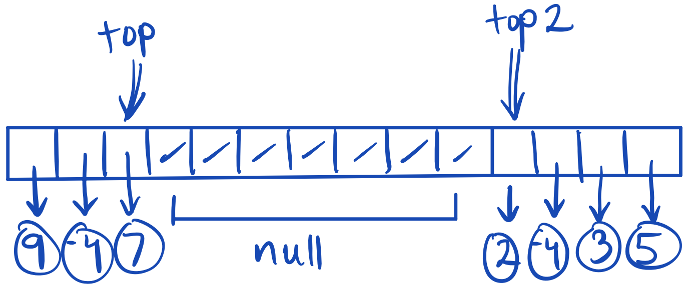
Cons: double amount of mem/item

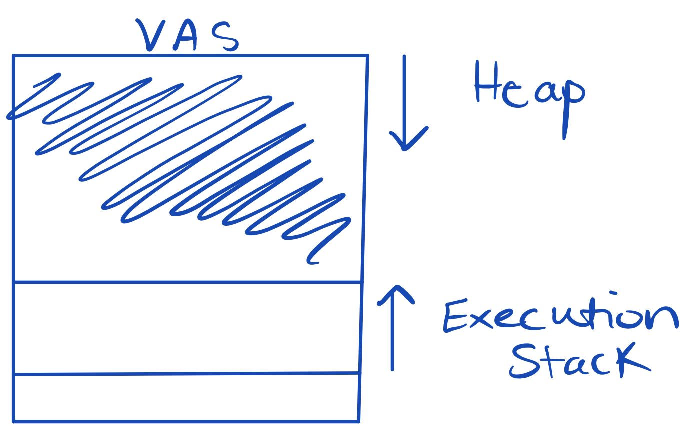




S1, S2 type Stack







Generic Classes

ArrayList<String> list;

Hashmap<Integer, String> hmp;

StackRA, StackSLS, Node

All occurrences of Object type replace with T.

Except for array allocation

```
T[] items;
items = (T[]) new Object[3]; // don't replace here
```

Use Node<T>