Adapting Data Structures

- Using one data structure to implement another
- Ideally, still needs to be efficient
- Sometimes doesn't make sense
- Good educational exercise

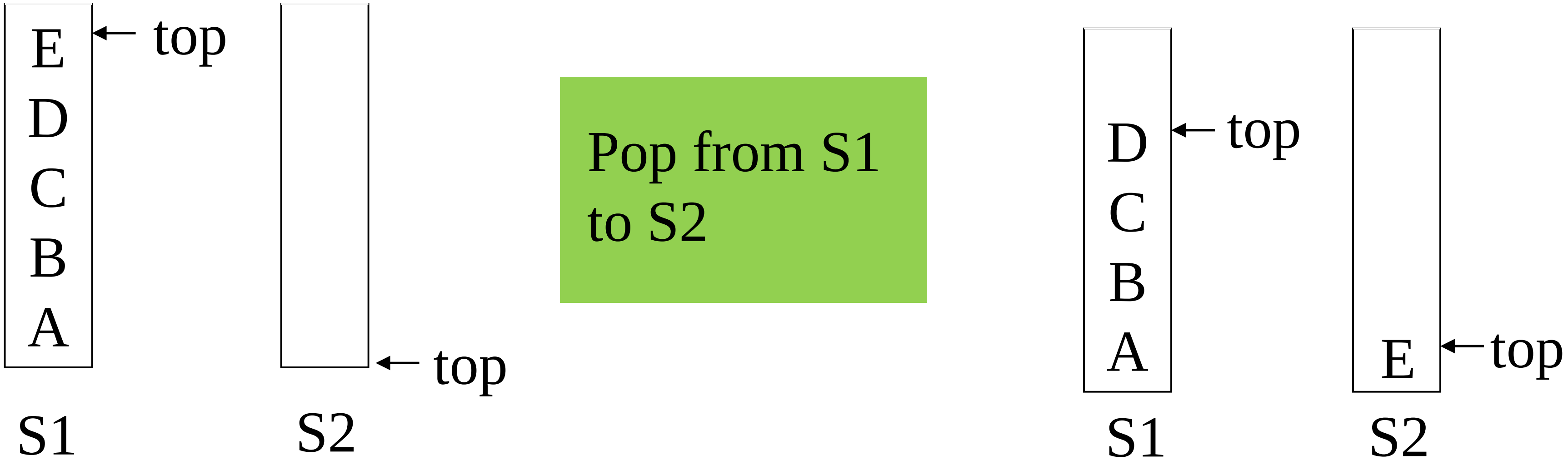
Can you make a Queue using a Stack?

No, not with just one stack

How about a Queue using two Stacks?

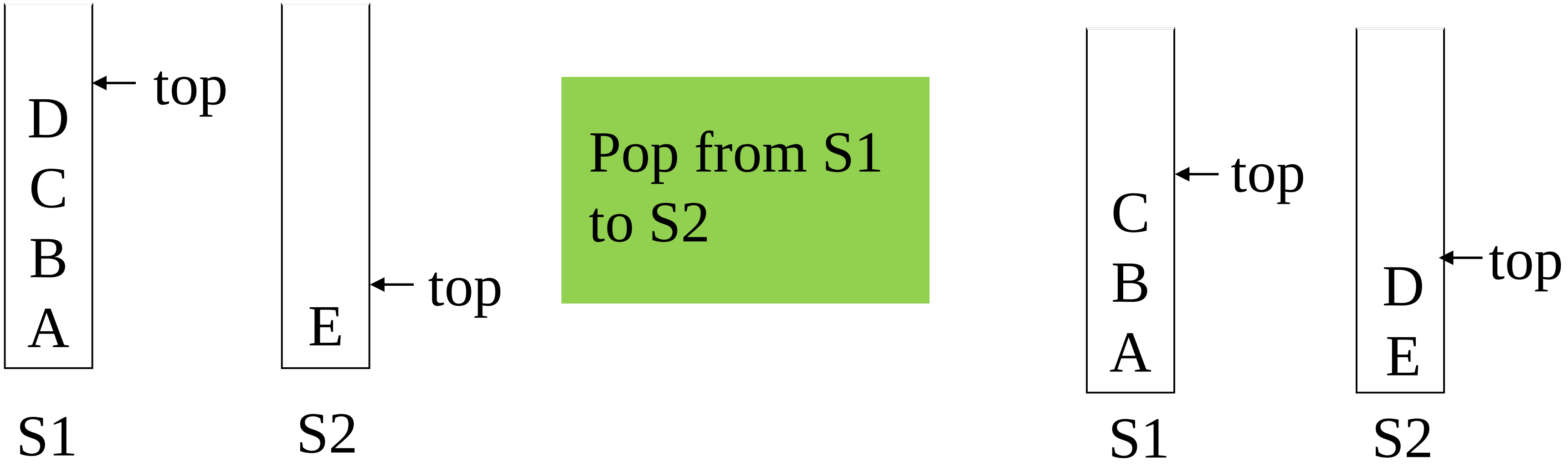
Yes, but how?

Observation #1



Pop from S1 to S2

Observation #1



top

D

C

B

A

S1

E

top

S2

Pop from S1 to S2

top

C

B

A

S1

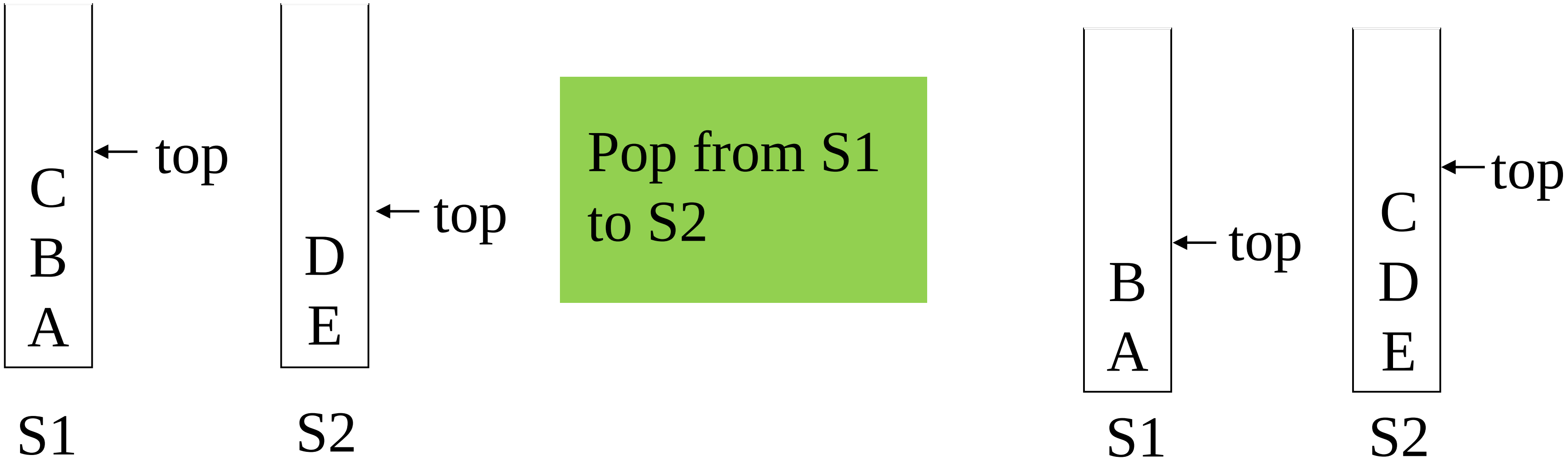
top

D

E

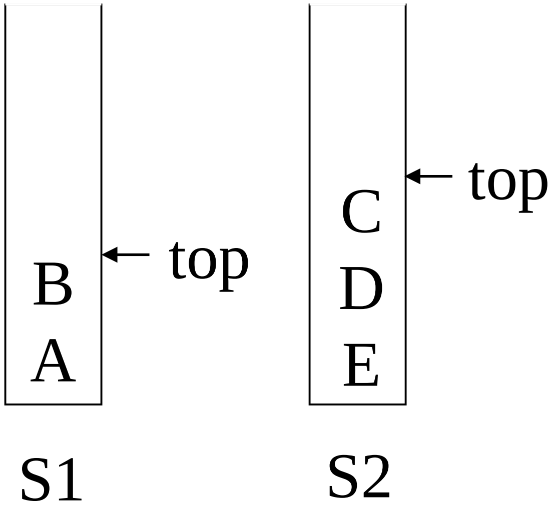
S2

Observation #1

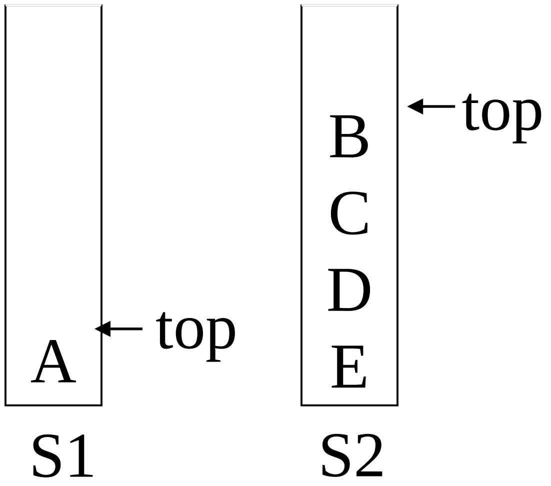


Pop from S1 to S2

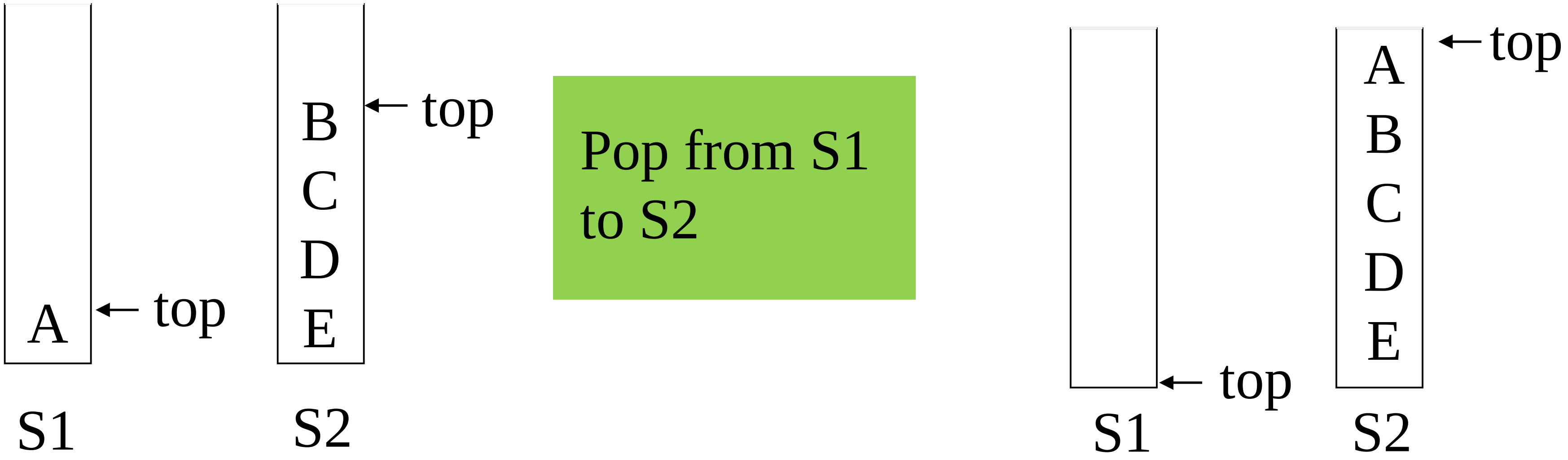
Observation #1



Pop from S1 to S2

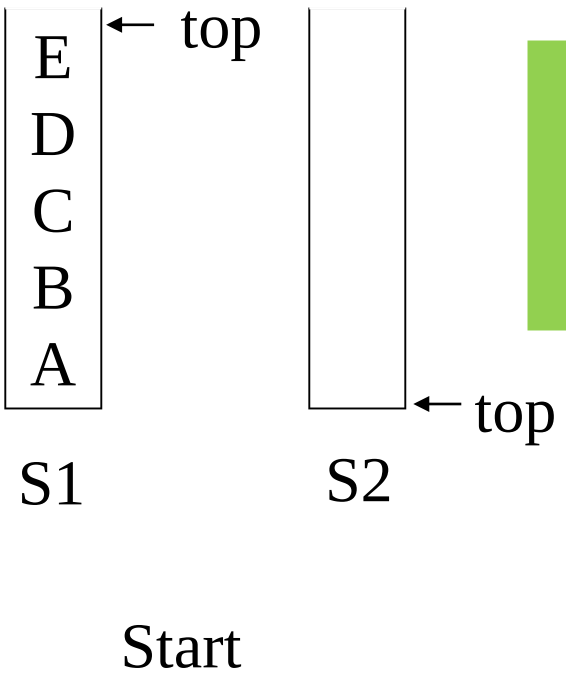


Observation #1



Pop from S1 to S2

Observation #1



E
D
C
B
A

← top

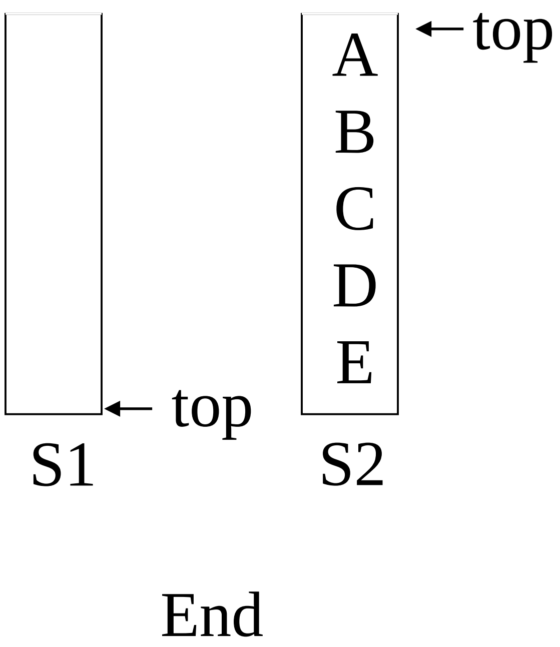
← top

S1

S2

Start

The order of the elements has been reversed



A
B
C
D
E

←top

← top

S1

S2

End

Idea #1

Implementation #1

```
For Enqueue(x):

    push x onto S1
```

```
For Dequeue():

    pop all elements from S1 to S2

    pop result from S2

    pop all elements from S2 to S1
```

Analysis #1

For Enqueue(x):

push x onto S1

Time: O(1)

For Dequeue():

pop all elements from S1 to S2

pop result from S2

pop all elements from S2 to S1

Time: O(n) if S1 has n elements

Idea #2

It’s not necessary to move the elements back from S2 to S1!

Implementation #2

```
For Enqueue(x):

    push x onto S1
```

```
For Dequeue():

    if S2 is empty:
        pop all elements from
         S1 to S2

    pop result from S2
```

Analysis #2

For Enqueue(x):

```
    push x onto S1
```

Time: O(1)

For Dequeue():

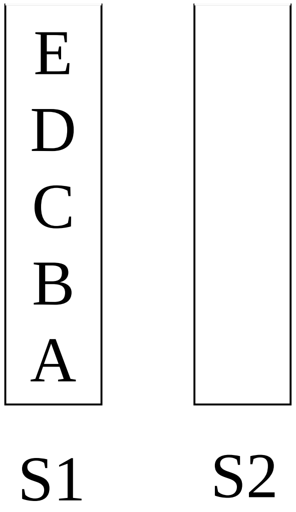
```
    if S2 is empty:
        pop all elements from S1 to S2

    pop result from S2
```

Time: O(n) if S2 is empty…

Example

Enqueue A,B,C,D,E

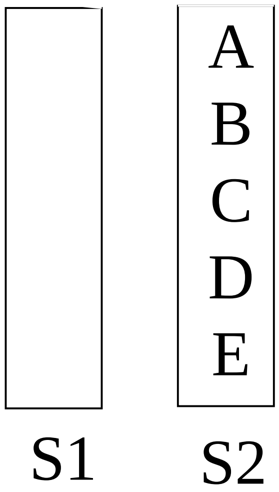


E
D
C
B
A

S1

S2

Dequeue()

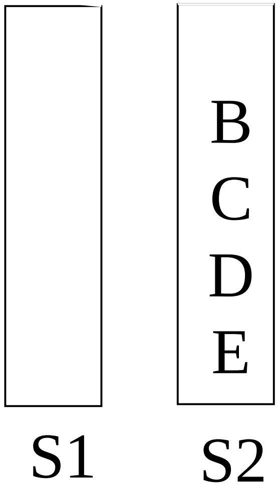


A
B
C
D
E

S1

S2

Return A



B
C
D
E

S1

S2

Example

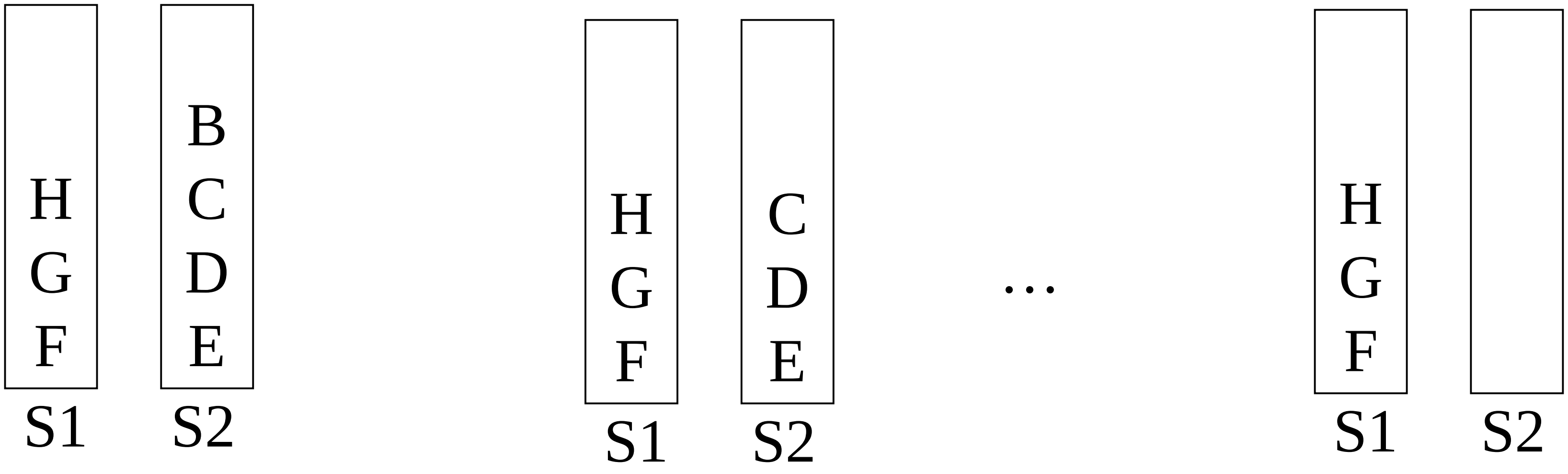
Enqueue F,G,H

Dequeue(),Dequeue(),Dequeue(),Dequeue()

Return B

Return C,D

Return E



H
G
F

B
C
D
E

S1 S2

H
G
F

C
D
E

S1 S2

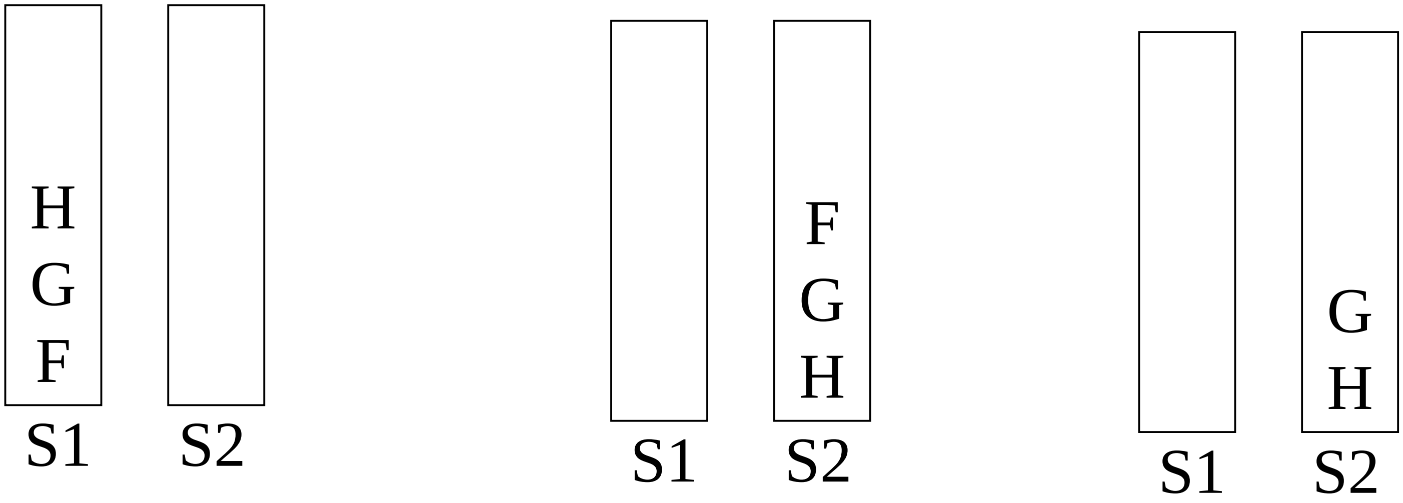
…

H
G
F

S1 S2

Example

Dequeue()



Return F

How many times can each element move from S1 to S2?

Just once

Amortized Analysis

For Enqueue(x):

```
    push x onto S1
```

Two tokens:
One pays for the push
Store one with x

Time: O(1) amortized

For Dequeue():

```
    if S2 is empty:
        pop all elements from
        S1 to S2

    pop result from S2
```

Pay for moving the items from S1 to S2 with the tokens stored in S1

Pay one token for the pop

Time: O(1) amortized