Data Structures and Algorithm Analysis

Lab 4, Stack and Queue.

Contents

Stack with algs4

The algs4 library contains 2 Stack implementation.

Stack using linked list:

https://algs4.cs.princeton.edu/code/javadoc/edu/princeton/cs/algs4/Stack.html

Stack using array:

https://algs4.cs.princeton.edu/code/javadoc/edu/princeton/cs/algs4/ResizingArrayStack.html

Using stack with algs4

Using the Stack class in algs4 library:

```
import edu.princeton.cs.algs4.Stack;
import edu.princeton.cs.algs4.StdOut;
public class TestStack {
  public static void main( String[] args ) {
    Stack<Integer> stack = new Stack<>();
    stack.push(0);
    stack.push(1);
    stack.push(2);

    StdOut.println("Iterate all elements in Stack:");
    for( Integer i : stack ) StdOut.print(" "+i);

    StdOut.printf("\n\nPop an element: %d\n", stack.pop());

    StdOut.println("\nAfter pop, the stack is:");
    for( Integer i : stack ) StdOut.print(" "+i);
    StdOut.println();
  }
}
```

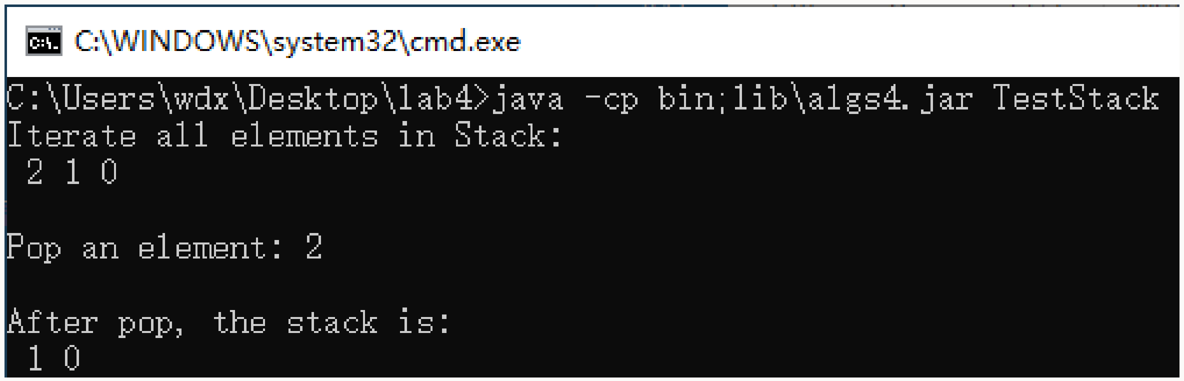
Using stack with algs4

```java
Stack<Integer> stack = new Stack<>();
stack.push(0);
stack.push(1);
stack.push(2);

StdOut.println("Iterate all elements in Stack:");
for( Integer i : stack ) StdOut.print(" "+i);

StdOut.printf("\n\nPop an element: %d\n", stack.pop());

StdOut.println("\nAfter pop, the stack is:");
for( Integer i : stack ) StdOut.print(" "+i);
StdOut.println();
```



```
C:\Users\wdx\Desktop\lab4>java -cp bin;lib\algs4.jar TestStack
Iterate all elements in Stack:
 2 1 0

Pop an element: 2

After pop, the stack is:
 1 0
```

Using stack with algs4

- Push: add one element in the stack.
- Pop: removes and returns the element on the top.
- Peek: returns the element on the top, do not remove.

```
Stack<Integer> stack = new Stack<>();

stack.push(0);
stack.push(1);
stack.push(2);

StdOut.println(stack.peek());
StdOut.println(stack.pop());
```

Using stack with algs4

Using the ResizingArrayStack:

```
import edu.princeton.cs.algs4.ResizingArrayStack;
import edu.princeton.cs.algs4.StdOut;
public class TestResizingArrayStack {
  public static void main( String[] args ) {
    ResizingArrayStack<Integer> stack = new
      ResizingArrayStack<>();
    stack.push(0);
    stack.push(1);
    stack.push(2);

    StdOut.println("Iterate all elements in Stack:");
    for( Integer i : stack ) StdOut.print(" "+i);

    StdOut.printf("\n\nPop an element: %d\n", stack.pop());

    StdOut.println("\nAfter pop, the stack is:");
    for( Integer i : stack ) StdOut.print(" "+i);
    StdOut.println();
  }
}
```

Linked list stack and resizing array stack

Stack and ResizingArrayStack provides the same functions. Let's comparing their implementations:

https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/Stack.java.html

https://algs4.cs.princeton.edu/code/edu/princeton/cs/algs4/ResizingArrayStack.java.html

Example: Arithmetic expression evaluation

Consider the following arithmetic expression:

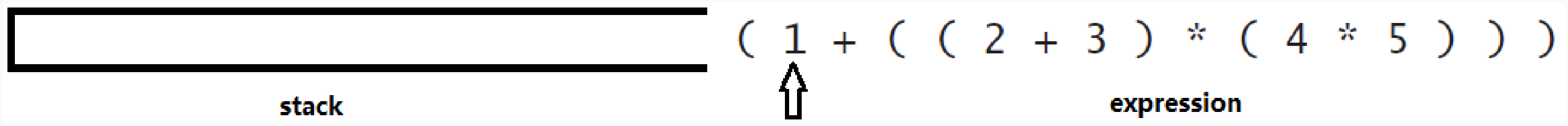
(1 + ((2 + 3) * (4 * 5)))

We want to parse this expression and calculate the final result (in this example, 101).

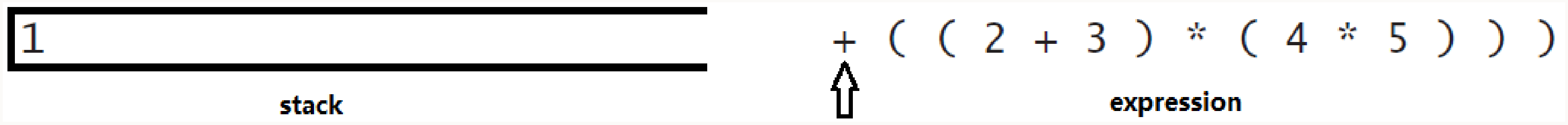
To simplify our calculation, we assume we have all the necessary spaces and parentheses.

Example: Arithmetic expression evaluation

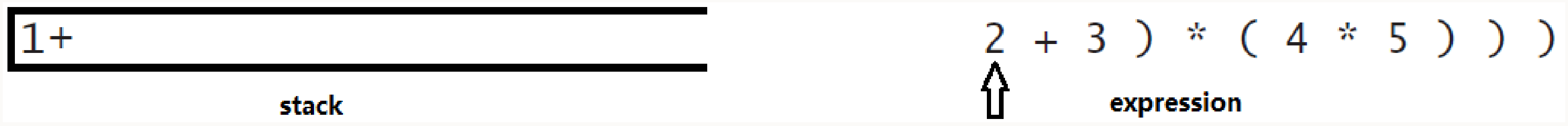
Using stack to parse the expression:



Push elements one by one in the array:

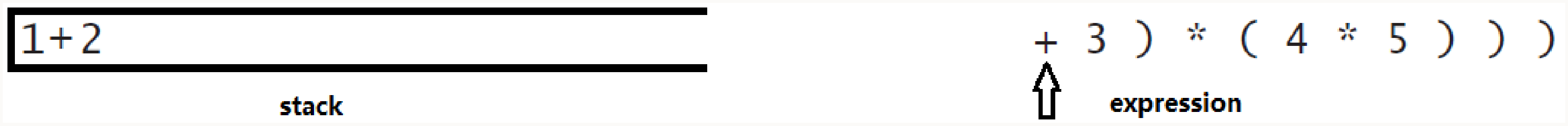


Ignoring all the left parentheses.

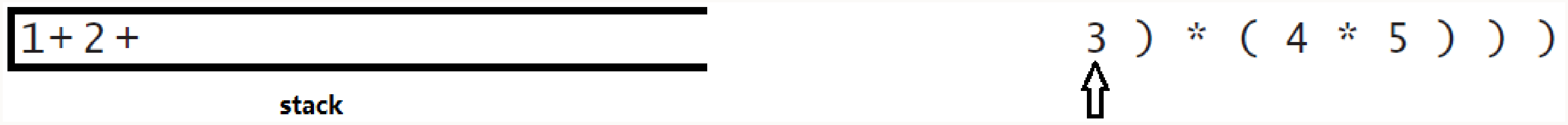


Example: Arithmetic expression evaluation

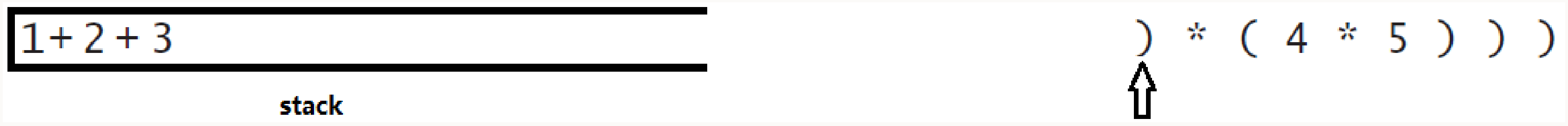
Keep parsing elements one by one:



Push elements one by one in the array:

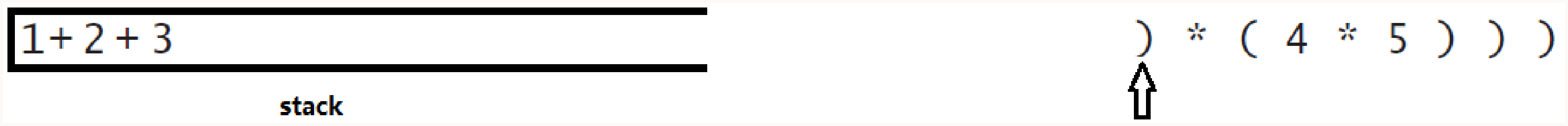


Ignoring all the left parentheses.

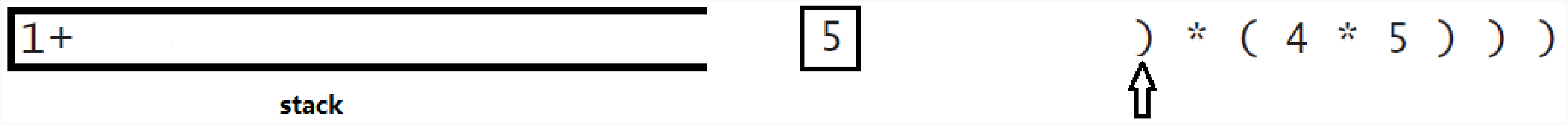


Example: Arithmetic expression evaluation

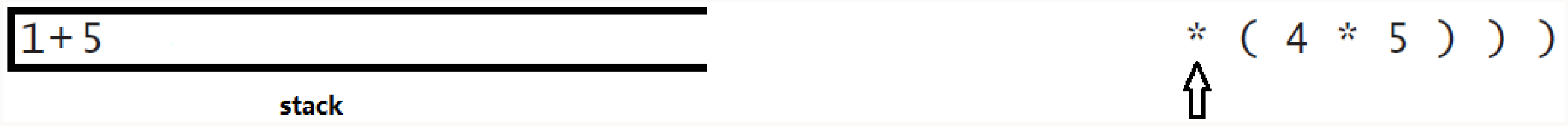
Here comes the problem: how to handle right parentheses ")":



We pop the top 3 elements from the stack, and calculate them:

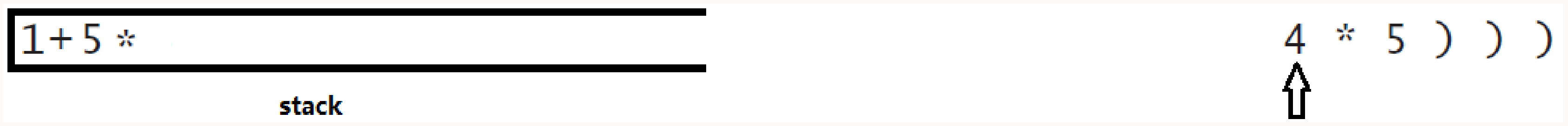


Then we push it back and move forward:

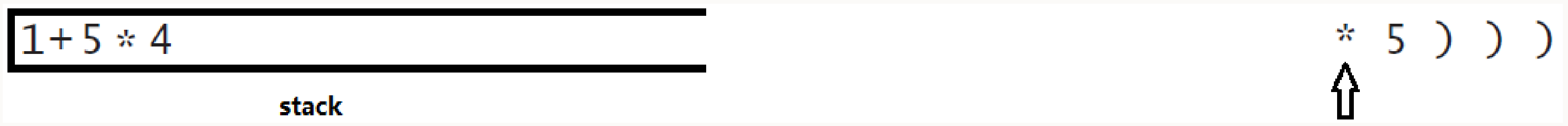


Example: Arithmetic expression evaluation

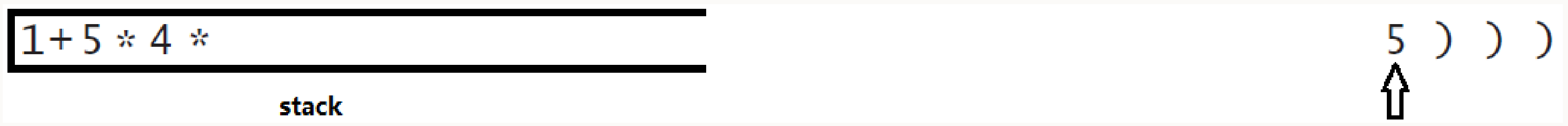
Continue:



Continue:

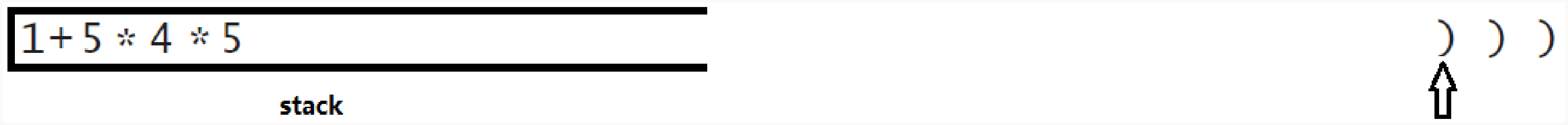


Continue:

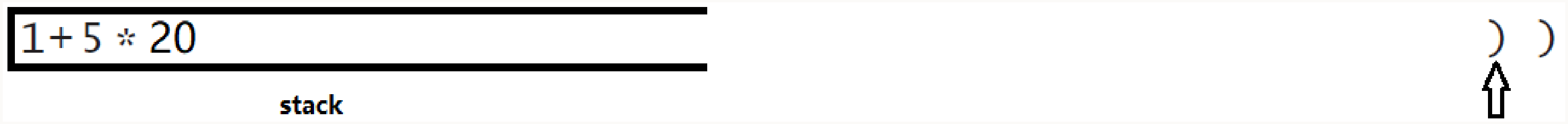


Example: Arithmetic expression evaluation

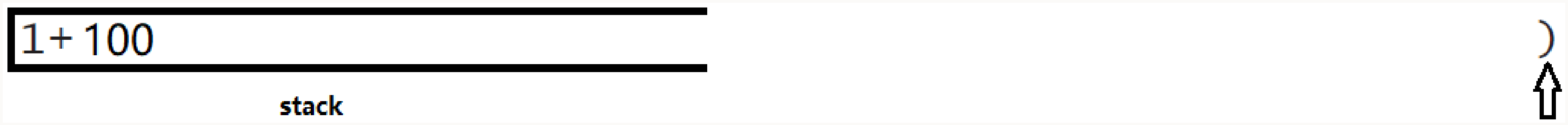
Continue:



Continue:



Continue:



Using queue with algs4

The algs4 library also contains 2 Queue implementations:

Queue using linked list:

https://algs4.cs.princeton.edu/code/javadoc/edu/princeton/cs/algs4/Queue.html

Queue using array:

https://algs4.cs.princeton.edu/code/javadoc/edu/princeton/cs/algs4/ResizingArrayQueue.html

Using queue with algs4

```java
import edu.princeton.cs.algs4.Queue;
import edu.princeton.cs.algs4.StdOut;
public class TestQueue {
  public static void main( String[] args ) {
    Queue<Integer> queue = new Queue<>();

    queue.enqueue(0);
    queue.enqueue(1);
    queue.enqueue(2);

    StdOut.println("Iterate all elements in Queue:");
    for( Integer i : queue ) StdOut.print(" "+i);

    StdOut.printf("\n\nDequeue an element: %d\n", queue.
      dequeue());

    StdOut.println("\nAfter dequeue, the queue is:");
    for( Integer i : queue ) StdOut.print(" "+i);
    StdOut.println();
  }
}
```

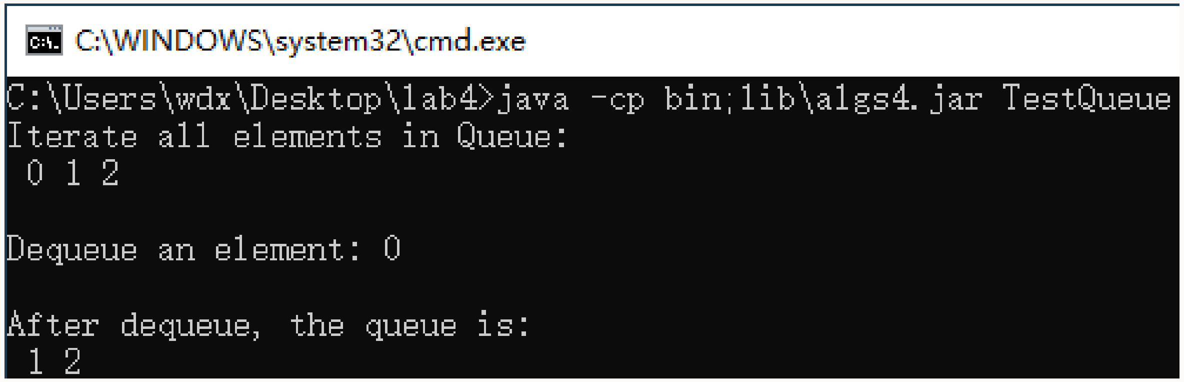
Using queue with algs4

```java
Queue<Integer> queue = new Queue<>();
queue.enqueue(0);
queue.enqueue(1);
queue.enqueue(2);

StdOut.println("Iterate all elements in Queue:");
for( Integer i : queue ) StdOut.print(" "+i);

StdOut.printf("\n\nDequeue: %d\n", queue.dequeue());

StdOut.println("\nAfter dequeue, the queue is:");
for( Integer i : queue ) StdOut.print(" "+i);
StdOut.println();
```



```
C:\Users\wdx\Desktop\lab4>java -cp bin;lib\algs4.jar TestQueue
Iterate all elements in Queue:
 0 1 2

Dequeue an element: 0

After dequeue, the queue is:
 1 2
```

Exercise 1: brackets check

Given a string with "(", ")", "[", "]", "{", "}", determine whether the string is valid.

A string is valid if:

- Open brackets must be closed by the same type of brackets.
- Open brackets must be closed in the correct order.
- An empty string could be seen as valid.

For example:

"(){}[]()", "(((())))", "([]){()[]}" are valid,

"(}(}", "(", "[}}]", "([)] are invalid.

Exercise 1: brackets check

Input: One line of string containing "(", ")", "[", "]", "{", "}".

Output: Print "1" for valid string and "0" for invalid string. Do not print extra new line.

Exercise 1: brackets check

Sample Input 1:

```
{}[]()
```

Sample Output 1:

```
1
```

Sample Input 2:

```
{}[](){
```

Sample Output 2:

```
0
```

Exercise 2: Queue iterator remove

Implement the "remove" method in Queue iterator. What's the time complexity of your implementation?

```
private class LinkedIterator implements Iterator<Item> {
  private Node<Item> current;

  public LinkedIterator(Node<Item> first) {
      current = first;
  }

  public boolean hasNext()  { return current != null;
                          }
  public void remove()      { // implement this }

  public Item next() {
      if (!hasNext()) throw new NoSuchElementException();
      Item item = current.item;
      current = current.next;
      return item;
  }
```