Finishing Stacks

1/26/16
Running times from lab
(courtesy of Lola Renteria and Vlad Papancea; Winter ‘15)

<table>
<thead>
<tr>
<th>size</th>
<th>ArrayList</th>
<th></th>
<th>LinkedList</th>
<th></th>
<th>TreeSet</th>
<th></th>
<th>HashSet</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>add</td>
<td>get</td>
<td>remove</td>
<td>add</td>
<td>get</td>
<td>remove</td>
<td>add</td>
<td>contains</td>
</tr>
<tr>
<td>25K</td>
<td>211</td>
<td>0</td>
<td>172</td>
<td>235</td>
<td>1,804</td>
<td>5</td>
<td>5,624</td>
<td>4,050</td>
</tr>
<tr>
<td>50K</td>
<td>399</td>
<td>8</td>
<td>391</td>
<td>389</td>
<td>3,039</td>
<td>12</td>
<td>12,134</td>
<td>3,837</td>
</tr>
<tr>
<td>100K</td>
<td>649</td>
<td>3</td>
<td>1,242</td>
<td>779</td>
<td>7,157</td>
<td>15</td>
<td>25,135</td>
<td>4,174</td>
</tr>
<tr>
<td>200K</td>
<td>1,391</td>
<td>3</td>
<td>2,727</td>
<td>1,565</td>
<td>13,667</td>
<td>5</td>
<td>58,807</td>
<td>4,974</td>
</tr>
<tr>
<td>400K</td>
<td>2,829</td>
<td>12</td>
<td>4,649</td>
<td>2,979</td>
<td>24,492</td>
<td>2</td>
<td>112,334</td>
<td>5,063</td>
</tr>
<tr>
<td>800K</td>
<td>3,922</td>
<td>13</td>
<td>8,696</td>
<td>4,764</td>
<td>48,448</td>
<td>5</td>
<td>233,346</td>
<td>5,236</td>
</tr>
</tbody>
</table>
Announcements

• HW 6: implement RPN calculator

• Friday’s quiz: Running times and maybe RPN

• Reading:
  – Friday: Chapter 7 thru “Recursively processing a linked chain” (pp. 198-211)
  – Monday: rest of Chapter 7
Recall: Post-fix notation
(aka Reverse Polish notation)

• Way to write arithmetic expressions where operators appear after the operands

  1 2 +  //evaluates to 1+2=3
  1 2 + 3 +  //evaluates to (1+2)+3=6

• Evaluation using a stack
  – push read values onto stack
  – when read an operator, pop off the top 2 values, apply operator, and push result onto the stack
  – after the expression, result will be the only value on the stack
Which of the following evaluates to 20 in RPN?

A. 2 3 4 2 * 7 + + +
B. 5 6 + 2 * 3 - 1 +
C. 2 3 5 2 * 1 + + + 3 1 + +
D. 1 2 2 * + 1 3 - 1 1 2 + + * *
E. Not exactly one of the above
Which of the following evaluates to 20 in RPN?

A. 2 3 4 2 * 7 + + +
B. 5 6 + 2 * 3 - 1 +
C. 2 3 5 2 * 1 + + + 3 1 + +
D. 1 2 2 * + 1 3 - 1 1 2 + + * *
E. Not exactly one of the above (not D=-40)
Implementing a RPN calculator

• Create stack of integers
  
  Stack<Integer> s = new Stack<Integer>();

• As you read tokens from the input
  
  – if it is an integer, push it onto stack
  
  – if it is an operator, pop off the top 2 values, apply
    operator, and push result onto the stack

• At the end of the input, the result should be
  the only value on the stack
Another application: program stack

• Each function call is represented by an activation record or activation frame stored in a stack in memory
  – This contains values of local variables and return address

• New function calls are pushed onto stack and returning calls pop off it
Implementing a stack

• Array-based
  – store values with oldest at cell 0
  – keep track of top index (just like num)

• Linked memory
  – keep reference (top) to Node at top of the stack
What is the running time of the linked-memory implementation of the Stack’s push method (add to front)?

A. Constant time  
B. Linear time
What is the running time of the linked-memory implementation of the Stack’s push method (add to front)?

A. Constant time
B. Linear time
What is the running time of the linked-memory implementation of the Stack’s pop method (remove first)?

A. Constant time
B. Linear time
What is the running time of the linked-memory implementation of the Stack’s pop method (remove first)?

A. Constant time
B. Linear time
What is the running time of the array-based implementation of the Stack’s push method (add to end)?

A. Constant time
B. Linear time
What is the running time of the array-based implementation of the Stack’s push method (add to end)?

A. Constant time
B. Linear time (but constant amortized time)
What is the running time of the array-based implementation of the Stack’s pop method (remove from end)?

A. Constant time
B. Linear time
What is the running time of the array-based implementation of the Stack’s pop method (remove from end)?

A. Constant time
B. Linear time
## Stack running times

<table>
<thead>
<tr>
<th></th>
<th>Push</th>
<th>Pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>array-based</td>
<td>Linear (constant amortized)</td>
<td>Constant</td>
</tr>
<tr>
<td>linked list</td>
<td>Constant</td>
<td>Constant</td>
</tr>
</tbody>
</table>