Today's Outline

- Tools of the trade: Analysis, Pseudocode, & Proofs
- Review: Stacks and Queues
- Homework #1

Algorithm Analysis: Why?

- Correctness:
  - Does the algorithm do what is intended?
- Performance:
  - What is the running time of the algorithm?
  - How much storage does it consume?
- Multiple algorithms may correctly solve a given task
  - Analysis will help us determine which algorithm to use

Pseudocode

- In the lectures algorithms will often be presented in pseudocode.
  - This is very common in the computer science literature
  - Pseudocode is usually easily translated to real code.
  - This is programming language independent

Pseudocode Example

```plaintext
mystery(v[ ]: integer array, num: integer): integer {
    temp: integer;
    temp := 0;
    for i := 0 to num - 1 do
        temp := v[i] + temp;
    return temp;
}
```

What does this pseudocode do?

Another Pseudocode Example

```plaintext
func(v[ ]: integer array, num: integer): integer {
    if num = 0 then
        return 0
    else
        return v[num-1] + func(v,num-1);
}
```

What does this pseudocode do?
Iterative Algorithm for Sum

- Find the sum of the first \textbf{num} integers stored in an array \textit{v}.

\begin{verbatim}
sum(v[ ], num, integer): integer {
    temp_sum: integer;
    temp_sum := 0;
    for i = 0 to num – 1 do
        temp_sum := v[i] + temp_sum;
    return temp_sum;
}
\end{verbatim}

Note the use of pseudocode

Programming via Recursion

- Write a \textit{recursive} function to find the sum of the first \textbf{num} integers stored in array \textit{v}.

\begin{verbatim}
sum(v[ ], num, integer): integer {
    if num = 0 then
        return 0
    else
        return v[num-1] + sum(v,num-1);
}
\end{verbatim}

Analysis: How?

- We will use mathematical analysis to examine the efficiency of code (next few lectures)
- How do we prove that an algorithm is correct?

Proof by Induction

- \textbf{Basis Step}: The algorithm is correct for the base case \textit{(e.g. n=0)} by inspection.
- \textbf{Inductive Hypothesis (n=k)}: Assume that the algorithm works correctly for the first \textit{k} cases, for any \textit{k}.
- \textbf{Inductive Step (n=k+1)}: Given the hypothesis above, show that the \textit{k+1} case will be calculated correctly.

Program Correctness by Induction

- \textbf{Basis Step}: \textit{sum(v,0) = 0}. ✓
- \textbf{Inductive Hypothesis (n=k)}: Assume \textit{sum(v,k)} correctly returns sum of first \textit{k} elements of \textit{v}, i.e. \textit{v[0]+v[1]+...+v[k-1]}
- \textbf{Inductive Step (n=k+1)}: \textit{sum(v,n)} returns \textit{v[k]+sum(v,k)} which is the sum of first \textit{k+1} elements of \textit{v}. ✓

Today’s Outline

- Tools of the trade: Analysis, Pseudocode, & Proofs
- Review: Stacks and Queues
- Homework #1
The Queue ADT

Queue Operations:
- create
- destroy
- enqueue
- dequeue
- is_empty

Circular Array Queue Data Structure

// Basic idea only!
enqueue(x) {
  Q[back] = x;
  back = (back + 1) % size
}

// Basic idea only!
disqueue() {
  x = Q[front];
  front = (front + 1) % size;
  return x;
}

Circular Array vs. Linked List

Array:
- May waste unneeded space or run out of space
- Space per element excellent
- Operations very simple / fast
- Constant-time access to kth element
- For operation insertAtPosition, must shift all later elements
- Not in Queue ADT

List:
- Always just enough space
- But more space per element
- Operations very simple / fast
- No constant-time access to kth element
- For operation insertAtPosition must traverse all earlier elements
- Not in Queue ADT

The Stack ADT

- Stack Operations:
  - create
  - destroy
  - push
  - pop
  - top/peek
  - is_empty

- Can also be implemented with an array or a linked list
- This is Project 1!
Stacks in Practice

- Function call stack
- Removing recursion
- Checking if symbols (parentheses) are balanced
- Evaluating Postfix Notation

Homework #1 – Sound Blaster!

- Reverse sound clips using a stack!
- Implement a stack interface two ways:
  - With an array
  - With linked list nodes (make your own nodes)
- Do NOT use LinkedList or other things from Java Collections

When did you take cse 143 (what quarter)?

<table>
<thead>
<tr>
<th>Numeric value</th>
<th>Answer</th>
<th>Frequency</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 - autumn 11</td>
<td>13</td>
<td>17.57%</td>
</tr>
<tr>
<td>2</td>
<td>1 - summer 11</td>
<td>3</td>
<td>4.05%</td>
</tr>
<tr>
<td>3</td>
<td>2 - spring 11</td>
<td>7</td>
<td>9.46%</td>
</tr>
<tr>
<td>4</td>
<td>3 - winter 11</td>
<td>19</td>
<td>25.68%</td>
</tr>
<tr>
<td>5</td>
<td>4 - autumn 10</td>
<td>6</td>
<td>8.11%</td>
</tr>
<tr>
<td>6</td>
<td>5 - summer 10</td>
<td>2</td>
<td>2.70%</td>
</tr>
<tr>
<td>7</td>
<td>6 - before summer 10</td>
<td>10</td>
<td>13.51%</td>
</tr>
<tr>
<td>8</td>
<td>7 - I did not take cse 143 at UW (AP or transfer credit)</td>
<td>4</td>
<td>5.41%</td>
</tr>
<tr>
<td>9</td>
<td>Other:</td>
<td>1</td>
<td>1.35%</td>
</tr>
</tbody>
</table>

Total responses (N): 74 Did not respond: 8