Informatik II
Tutorial 3

Subho Shankar Basu
subho.basu@inf.ethz.ch
16.03.2016
Overview

- Debriefing Exercise 2
- Briefing Exercise 3
U2.A1

- Represent tree with brackets and indented form

- Reconstruct a tree from bracket representation?

  $S(R(H(K)), P(A(N,O), Q,T), V(J,F(G)))$

- Yes, if the position of the nodes is irrelevant (left/right)
Leaves: 65, 41, 7, 1, 13, 54, 17, 5, 3
Height of tree: 4 (considering it starts from 0)
Levels: 5
Longest path: 2->76->11->25->13, etc.
Some tips

- Pay attention to the number of brackets
- How are “K, N, O, etc.” called?
  - Leaves
- Longest path depends if the tree is defined as directed
  - Computer Science: tree = Connected, acyclic and directed
recursiveSort(4)

recursiveSort(3)

recursiveSort(2)

recursiveSort(1)

Sorted!

9 <- findLargest(0,3)
Swap

5 <- findLargest(1,3)
Swap

2 <- findLargest(2,3)
Swap

No need for further swap!

→ List is now in decreasing order!
Generate random number to fill the random array

```java
import java.util.Random;

Random r = new Random();
r.nextInt(1000); // produces from 0 to 999
```
toString() function for the array

// numbers is the array of integers

public String toString()
{
    String s="["
    for (int i=0; i<numbers.length; i++) {
        if (i!=0) {
            s = s + ", ";
        }
        s = s + numbers[i];
    }
    s = s + "]";
    return s;
}
/**
 * swaps two fields of {@link RandomArray#numbers}
 * @param i a valid index into {@link RandomArray#numbers}
 * @param j a valid index into {@link RandomArray#numbers}
 */
private void swap(int i, int j)
{
    int tmp = numbers[j];
    numbers[j] = numbers[i];
    numbers[i] = tmp;
}
## Swap

- Is there a way to swap without a temporary variable?

\[
\begin{align*}
X &:= X \text{ XOR } Y \\
Y &:= X \text{ XOR } Y \\
X &:= X \text{ XOR } Y \\
\end{align*}
\]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1010</td>
<td>0011</td>
<td>(\oplus)</td>
</tr>
<tr>
<td>1001</td>
<td>0011</td>
<td>(\oplus)</td>
</tr>
<tr>
<td>1001</td>
<td>1010</td>
<td>(\oplus)</td>
</tr>
<tr>
<td>0011</td>
<td>1010</td>
<td></td>
</tr>
</tbody>
</table>

Now try with “+-” and “*/” operators
How to do the swap?

- Swap inside the loop

```java
void recursiveSort( int until ) {
    // 0 elements are considered to be sorted
    if( until == 0 )
        return;

    // sort first until-1 elements in the array
    recursiveSort( until - 1 );

    // bring the greatest element from the rest to position until-1
    for( int i = until; i < a.length; i++ )
    {
        if( a[i] > a[until-1] ){
            swap(until-1, i);
        }
    }
}
```
How to do the swap?

- Any better idea?
  - First find the item to swap, then do only 1 swap!

```java
void recursiveSort( int until ) {
    // 0 elements are considered to be sorted
    if( until == 0 )
        return;

    // sort first until-1 elements in the array
    recursiveSort( until - 1 );

    // find index of greatest element after until-1
    int maxIndex = until - 1;
    for( int i = until; i < a.length; i++ ) {  // loop through the array
        if( a[i] > a[maxIndex] ) {
            maxIndex = i;
        }
    }

    // swap elements at maxIndex and until-1
    swap( until-1, maxIndex );
}
```
Coding Style

- Formatting code
- Eclipse: Ctrl+Shift+F and the code is nicely formatted (indented)

```java
while ((e+i)<=14) {
    if (a[e] > a[e+i]) {
        e++;
        i=1;
    }
    else
        i++;
}
```
Coding style

- Try to avoid hardcoding!

```java
if (myString.compareTo("hello world") == 0);
private static final String REF = "hello world";
... if (myString.compareTo(REF) == 0);
```
Coding style

- Loops

for:

when iterating

```c
for( initialization; condition; increment/decrement)  
{
    //statements to execute...You know in advance how
    //many times to loop through...
}
```

while:

for specific cases

```c
int guess;  
while( guess != 100 )  
{
    // Try again...You don’t know until the outcome happens...
}
```
## Coding style

### Differences

<table>
<thead>
<tr>
<th>Codice</th>
<th>Codice</th>
</tr>
</thead>
</table>
| ```c
if (index >= boundary)
    return;
else if (array[index] == 'x')
    return;
```                                                                 | ```c
if ( index >= boundary ||
    array[index] == 'x' )
    return;
```                                                                 |
| **Y in expression (X || Y) is only evaluated if X == false (border effect)** |                                                                       |
| | **Y in expression (X && Y) is only evaluated if X == true**                                                      |
| ```c
if (index < boundary)
    if (array[index] == 'x')
        array[index] = '\0';
```                                                                 | ```c
if ( index < boundary &&
    array[index] == 'x' )
    array[index] = '\0';
```                                                                 |
| |                                                                       |
| ```c
int counter = 0;
while (counter < n) {
    ...            
    counter++;  
}
```                                                                 | ```c
for ( int counter = 0;
    counter < n;
    counter++) {
    ...            
}
```                                                                 |
| **Warning: counter is still defined outside the loop!**                        | **Clean counting: counter can be reused out of the for loop.**            |

---

- In expression `(X || Y)` is only evaluated if `X == false` (border effect).
- In expression `(X && Y)` is only evaluated if `X == true`.
- Warning: counter is still defined outside the loop!
- Clean counting: counter can be reused out of the `for` loop.
Coding style

- Efficiency

Object initialization is expensive!

```java
void initialize() {
    for (int i=0; i<a.length; i++) {
        Random r = new Random();
        a[i] = r.nextInt(1000);
    }
}
```

```java
void initialize() {
    Random r = new Random();
    for (int i=0; i<a.length; i++) {
        a[i] = r.nextInt(1000);
    }
}
```
U2.A3

a) leftChild, rightChild and father

- Root at index 0
- Direct successors for i are at position $2i + 1$ and $2i + 2$

```c
int leftChild( node ){
    return 2 * node + 1;
}

int rightChild( node ){
    return 2 * node + 2;
}

int father( node ){
    return (node - 1) / 2;
}

(father(0) = -1 / 2 = 0)
```
checkTree()

Test if an input array represent a binary tree
  - Each node must have a father
  - The root is its own father

What about empty nodes?
  - We ignore them (no need for a father)
/**
 * Check if the given array represents a valid binary tree.
 * 
 * @param array a binary tree encoded as char array
 * @throws IllegalArgumentException if check fails
 */
private static void checkTree(char[] array)
{
    if (array.length == 0) throw new IllegalArgumentException("At least one, probably empty node is required.");
    for (int i=0; i<array.length; i++) {
        if (array[i] != ' ') {
            int f = father(i);
            if (array[f] == ' ') {
                throw new IllegalArgumentException("node number " + i + " has no father");
            }
        }
    }
}
checkTree() fail case:

U2.A3 toString()  (Recursive approach)

/**
 * recursive toString
 *
 * @param node index of the root of the subtree which has to be converted.
 * @param indentation the current indentation
 * @return indented form of the selected subtree
 */
private String toString(int node, String indentation)
{
    assert (tree[node] != ' ');

    String s = indentation;
    s = s + tree[node] + '\n';

    int c1 = leftChild(node);
    if (isNode(c1)) {
        s = s + toString(c1, indentation + " ");
    }

    int c2 = rightChild(node);
    if (isNode(c2)) {
        s = s + toString(c2, indentation + " ");
    }

    return s;
}
Overview

- Debriefing Exercise 2
- Briefing Exercise 3
Homework

1. Objects and references (e.g. Strings)
   - Strings (Immutable, fast search, use for infrequent updates) vs. StringBuffer (Mutable, slow search, use for heavy updates)
   - Caesar cipher
   - Encrypt(using Strings) and decrypt(using StringBuffer), understand how the program works.

2. Syntax diagrams
   - Given some diagrams, which expressions can be produced?

3. Syntax checker for trees
   - Complete the syntax diagram from class to include empty trees and subtrees. Take care that no invalid trees are generated.
   - Implement it
Ein Syntaxdiagramm-Beispiel: Klammerdarstellung eines Baums

Beispiel: $A( B( D ), C( E, F ))$

**Baum:**

**Knoten:**

**Unterbäume:**

Wie könnte man Binärbäume durch ein Syntaxdiagramm darstellen?
U3.A1 Hints

- **String**
  - Immutable
  - Optimization possible because static
  - Modification only through copy

- **StringBuffer**
  - Mutable
  - Easily modifyable (without copy)
  - Some operations are more expensive (e.g. search)
String vs. StringBuffer

**String**

```java
String myString = "hello";
myString = myString + " world";
```

**JAVA String concatenation**

**StringBuffer**

```java
StringBuffer myStringBuffer = new StringBuffer("hello");
myStringBuffer.append(" world");
```

**StringBuffer Method**

**Memory**

- "hello"
- "world"
- "hello world"
- "world"
More about Strings

```java
String myString = "hello";
myString = myString+" world";
myString = myString+" how";
myString = myString+" are";
myString = myString+" you";
myString = myString+" today";
```

Speicher

"hello"       " world"
"hello world" " how"
"hello world how" " are"
"hello world how are" " you"
"hello world how are you" " today"
"hello world how are you today"
U3.A2 Hints

- Syntax diagrams were covered in class

Var:

Clause:

Expr:

\[ \text{e.g. } (~ X_1 \text{OR} X_2) \text{AND} (X_n) \]
U3.A3 Hints (Recursive descent parser)

- Implementing a syntax checker for trees
  - First you have to modify the syntax to accept empty trees and subtrees
  - Implement
    - Own methods for Tree, Successor and Node

- Offset = current position in the bracket representation of the tree. At the end, the offset should be equal to `str.length()`.

- Possible problems
  - `StringIndexOutOfBoundsException` – you are trying to access character at position `n` in the string, but the array is shorter than `n`. 
Ein Syntaxdiagramm-Beispiel: Klammerdarstellung eines Baums

Beispiel: $A(B(D), C(E,F))$

Baum:

Knoten:

Unterbäume:

Wie könnte man Binärbäume durch ein Syntaxdiagramm darstellen?
Have Fun!