Informatik II (D-ITET)

Tutorial 2

TA: Anwar Hithnawi, E-mail: hithnawi@inf.ethz.ch
Distributed Systems Group, ETH Zürich
Outlook

- Exercise 1 solution discussion
- Exercise 2 tackles (trees, recursion, sorting)
Exercise 2

1. Rooted Trees (theory)
   a. Representation of tree using (i) brackets and (ii) indented
   b. Given brackets representation, (i) draw tree and (ii) give indented format
   c. Can the tree in 1b be reconstructed no ambiguously? Why/Why not?
   d. For the trees in 1a and 1b: Give (i) height of tree [1 node has height 1], (ii) longest paths [trees are directed!], and (iii) set of leaves

2. Recursive Sorting
   a. Constructor: Create array of given size and fill with random numbers
   b. Build method toString
   c. Create recursiveSort(int until) to sort numbers in descending order

3. Binary Trees
   § Check Trees
Exercise 2 – Q 1 & 3

Overview on some different types of trees

- **General Tree**: Every node has \( X \) child nodes
Excersise 2 – Q 1 & 3

Overview on some different types of trees

- **General Tree**: Every node has X child nodes
- **Binary Tree**: Each node has at most two child nodes
Overview on some different types of trees

- **General Tree**: Every node has \( X \) child nodes

- **Binary Tree**: Each node has at most two child nodes

- **Binary/Ternary Search Tree (BST)**: Nodes are saved in an ordered form
Excercise 2 – Q 1 & 3

Overview on some different types of trees

- **General Tree**: Every node has X child nodes
- **Binary Tree**: Each node has at most two child nodes
- **Binary/Ternary Search Tree (BST)**: Nodes are saved in an ordered form
- **Trie (from «Retrieval»)**: Not the content but the position of the node that matters, i.e. edges carry information!
  
  (e.g. Suffix tree → text autocomplete)

**Task**: Deal with different representations of trees!
Q2: Recursive Sorting

- **Constructor**
  - Produce array of randomly generated numbers
  - Import Random Class (package: `import java.util.Random`)

```java
//RandomGenerator
Random r = new Random();

//Array...

//1 random number generieren:
r.nextInt(1000);
```

- **Method toString()**

```java
String s = "";
for (int i=0; i < array.length, i++ )
    ...  
return s;
```
Q2: Recursive Sorting

- `recursiveSort(int until)`
  - Core idea of recursion is to reduce the problem to smaller instances of the same problem ...
  - NOW! Given a list of (N) element

To order list of i elements in descending order, I need the following...
  ... Sorting the first (i - 1) elements with descending order
  ... Search for the largest element in the list reminder
  ... place it in the first place of the list reminder

- The empty list is a sorted list
List sorted in descending order!
Q3: Binary Tree as an Array

- Binary trees can be saved as arrays given proper interpretation are set

- The idea is as follow:
  - Set the root of the tree to have an index 0
  - The next two array positions from (i) are saved, namely the position (2i + 1) and (2i +2)
  - What is the size of the array that stores the binary tree?

\[ 2^{ \text{height}-1} \leq \text{array.length} < 2^{\text{height}} \]
Q3: Binary Tree as an Array

The code snippet is as follows:

```java
char[] tree = new char[7];

tree[0] = 'A';
tree[1] = 'B';
tree[2] = 'C';
tree[3] = 'D';
tree[4] = ' ';
tree[5] = 'F';
tree[6] = 'E';
```

Is this also possible with general (= non-binary) trees?
Q3: Binary Tree as an Array

- **toString()**
  - **Idea:**
    - `toString()` call ➔ `toString(int node, String indentation)`
    - E.g. `toString(0," ");`

- **checkTree()**
  - **Idea:**
    - Root at index 0
    - Direct successor $i$ to $2i + 1$ and $2i + 2$
      - $2^{\text{height}-1} \leq \text{array.length} < 2^{\text{height}}$
    - Check if this applies for the passed array
      - Test: Every element has a parent node
        - "The root is its own father."
      - What about the empty nodes?
Trees in Computer Science...

Images:  http://kitabundsunnah.wordpress.com/,  http://www.cs.lmu.edu/courses
Tree traversal...

```
preOrder(node) {
    print(node)
    if left != null then preOrder(left)
    if right != null then preOrder(right)
}
```

- **Pre-Order**  «root, left, right»
- **In-Order**    «left, root, right»
- **Post-Order** «left, right, root»
Tree traversal...

- **Pre-Order** «root, left, right»

- **In-Order** «left, root, right»

- **Post-Order** «left, right, root»

```java
inOrder(node) {
    if left != null then preOrder(left)
    print(node)
    if right != null then preOrder(right)
}
```

8, 3, 1, 6, 4, 7, 10, 14, 13

1, 3, 4, 6, 7, 8, 10, 13, 14
Tree traversal...

- **Pre-Order** «root, left, right»
- **In-Order** «left, root, right»
- **Post-Order** «left, right, root»

```java
postOrder(node) {
    if left != null then preOrder(left)
    if right != null then preOrder(right)
    print(node)
}
```

**Example:**
- **Pre-Order** sequence: 8, 3, 1, 6, 4, 7, 10, 14, 13
- **In-Order** sequence: 1, 3, 4, 6, 7, 8, 10, 13, 14
- **Post-Order** sequence: 1, 4, 7, 6, 3, 13, 14, 10, 8
Eclipse more tricks...

- Display keyboard shortcuts: Control + Shift + L
- Auto-formatting: Control + Shift + F
- Auto-completion: Control + Space
Have Fun!