Overview

- Debriefing Exercise 11
- Briefing Exercise 12
U11 Time Complexity

- Landau-Symbols

- Estimation by analysis
  - grows... By

- O-Notation
  - Upper bound

- Omega-Notation
  - Lower bound

- Theta-Notation
  - Tight bound

<table>
<thead>
<tr>
<th>Notation</th>
<th>Intuitive Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f \in O(g) )</td>
<td>( f ) does not grow faster than ( g )</td>
</tr>
<tr>
<td>( f \in \Omega(g) )</td>
<td>( f ) does not grow much slower than ( g )</td>
</tr>
<tr>
<td>( f \in \theta(g) )</td>
<td>( f ) grows exactly as quickly as ( g )</td>
</tr>
</tbody>
</table>

http://de.wikipedia.org/wiki/Landau-Symbole

<table>
<thead>
<tr>
<th>Notation</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O(1) )</td>
<td>Constant</td>
</tr>
<tr>
<td>( O(\log(n)) )</td>
<td>Logarithmic</td>
</tr>
<tr>
<td>( O((\log(n))^c) )</td>
<td>Polylogarithmic</td>
</tr>
<tr>
<td>( O(n) )</td>
<td>Linear</td>
</tr>
<tr>
<td>( O(n^2) )</td>
<td>Quadratic</td>
</tr>
<tr>
<td>( O(n^c) )</td>
<td>Polynomial</td>
</tr>
<tr>
<td>( O(c^n) )</td>
<td>Exponential</td>
</tr>
</tbody>
</table>
U11.A1 Sorting by search trees

- Insert all numbers and then read in-order
- In the best case, the values in the list are well-mixed
  - balanced tree
- In the worst case, the values in the list are sorted in ascending or descending order
  - degenerate tree

Complexity
- In best case: \( O(n \cdot \log n) \)
- In average case: \( O(n \cdot \log n) \)
- In worst case: \( O(n^2) \)
U11.A2

// Fragment 1  \(O(n)\)
for (int i=0; i<n; i++)
a++;

// Fragment 2  \(O(n)\)
for (int i=0; i<2n; i++) a++;
for (int j=0; j<n; j++) a++;

// Fragment 3  \(O(n^2)\)
for (int i=0; i<n; i++)
  for (int j=0; j<n; j++) a++;

// Fragment 4  \(O(n^2)\)
for (int i=0; i<n; i++)
  for (int j=0; j<i; j++) a++;

// Fragment 5  \(O(\log n)\)
while(n >=1 )
  n = n/2;

// Fragment 6  \(O(n^5)\)
for (int i=0; i<n; i++)
  for (int j=0; j<n*n; j++)
    for (int k=0; k<j; k++)
      a++;
U11.A3 Complexity (1)

\[ t_{op} = \frac{1}{3} t_{op} \]

- Time per Operation
- Input Size
- Total run time

\[ M' \]
\[ T'_\text{tot} \]
# U11.A3 Complexity (2)

<table>
<thead>
<tr>
<th>O(...)</th>
<th>$T_{tot}$</th>
<th>$T'_{tot}$</th>
<th>$T'<em>{tot} = T</em>{tot}$</th>
</tr>
</thead>
</table>
| $O(n)$ | $T_{tot} = t_{op} \cdot M_1$ | $T'_{tot} = t'_{op} \cdot M'_1$ | $t'_{op} \cdot M'_1 = t_{op} \cdot M_1 \Rightarrow \frac{1}{3}t'_{op} \cdot M'_1 = t_{op} \cdot M_1$  
$\Rightarrow M'_1 = 3M_1$ |
| $O(n^2)$ | $T_{tot} = t_{op} \cdot M_2^2$ | $T'_{tot} = t'_{op} \cdot M'_2^2$ | $t'_{op} \cdot M'_2^2 = t_{op} \cdot M_2^2 \Rightarrow \frac{1}{3}t'_{op} \cdot M'_2^2 = t_{op} \cdot M_2^2$  
$\Rightarrow M'_2^2 = 3M_2^2 \Rightarrow M'_2 = \sqrt{3}M_2$  
$\sim 1,7$ |
| $O(2^n)$ | $T_{tot} = t_{op} \cdot 2^{M_3}$ | $T'_{tot} = t'_{op} \cdot 2^{M'_3}$ | $t'_{op} \cdot 2^{M'_3} = t_{op} \cdot 2^{M_3} \Rightarrow \frac{1}{3}t'_{op} \cdot 2^{M'_3} = t_{op} \cdot 2^{M_3}$  
$\Rightarrow 2^{M'_3} = 2^{M_3} \cdot 3 \Rightarrow M'_3 = M_3 + \log_2 3 \sim 1,5$ |
| $O(\log_2 n)$ | $T_{tot} = t_{op} \cdot \log_2 M_4$ | $T'_{tot} = t'_{op} \cdot \log_2 M'_4$ | $t'_{op} \cdot \log_2 M'_4 = t_{op} \cdot \log_2 M_4$  
$\Rightarrow \frac{1}{3}t'_{op} \cdot \log_2 M'_4 = t_{op} \cdot \log_2 M_4$  
$\Rightarrow \log_2 M'_4 = 3 \cdot \log_2 M_4 \Rightarrow M'_4 = 2^{3\log_2 M_4}$  
$\Rightarrow M'_4 = (2^{\log_2 M_4})^3 \Rightarrow M'_4 = (M_4)^3$ |
U11.A4 A Knight on a chessboard
U11.A4a Reachable Fields

- Find the set of fields:
  - Reachable by \((n)\) moves,
  - Given: start position
U11.A4a Reachable fields

```java
public ArrayList<Position> getReachableSet(Position pos, int numberOfMoves) {
    ArrayList<Position> visited = new ArrayList<Position>();
    visit(pos, numberOfMoves, 0, visited);
    return visited;
}
```

```java
private void visit(Position pos, int maxDepth, int depth, ArrayList<Position> visited) {
    if (!visited.contains(pos)) {
        visited.add(pos);
    }
    if (depth == maxDepth) return;
    for (Position possibleMove : possibleMoves) {
        Position newPos = pos.add(possibleMove);
        if (check(newPos)) {
            visit(newPos, maxDepth, depth+1, visited);
        }
    }
}
```
U11.A4a How to get all the possible moves?

```java
private ArrayList<Position> possibleMoves;

Knight()
{
    possibleMoves = new ArrayList<Position>(8);
    possibleMoves.add(new Position( 1, 2));
    possibleMoves.add(new Position( 2, 1));
    possibleMoves.add(new Position( 2,-1));
    possibleMoves.add(new Position( 1,-2));
    possibleMoves.add(new Position(-1,-2));
    possibleMoves.add(new Position(-2,-1));
    possibleMoves.add(new Position(-2, 1));
    possibleMoves.add(new Position(-1, 2));
}
```
U11.A4b Knight’s tour

public ArrayList<Position> findCompletePath(Position pos) {
    ArrayList<Position> path = new ArrayList<Position>();
    if (explore(pos, path)) {
        return path;
    } else {
        return null;
    }
}

private boolean explore(Position pos, ArrayList<Position> path) {
    if (path.contains(pos)) {
        return false;
    }
    path.add(pos);
    if (path.size() == IKnight.boardSize * IKnight.boardSize) {
        return true;
    }
    for (Position possibleMove : possibleMoves) {
        Position newPos = pos.add(possibleMove);
        if (check(newPos)) {
            if (explore(newPos, path)) {
                return true;
            }
        }
    }
    path.remove(path.size() - 1);
    return false;
}
Overview

- Debriefing Exercise 11
- Briefing Exercise 12
U12.A1 Heap

- Heap = binary tree, but:
  - all levels (except possibly the last) are completely filled
  - The last level is filled from the left
  - For all k nodes (except the root):
    - value (previous (k)) \leq value (k) in a MIN-Heap
    - Or \geq in a MAX-Heap

- Properties (MIN-Heap):
  - Root has the smallest value
  - All paths from the root to a leaf are monotonically increasing
U12.A1 Heap

Heap as tree

Insert

Heap as Array
U12.A1

- **a,b: Theory**
  - How many elements are in a heap of height \( h \) containing minimum and maximum?
  - Is a sorted array a heap? (if interpreted as a binary tree? And the other way around?)

- **c: Heap Sort**
  - Phase 1: Array converted to heap
  - Phase 2: Read sorted heap, remove from the root

- **d: Implementation**
  - 2 phases
  - All operations are in-place!

U12.A2 – Parallelized Merge Sort

a) Much is up to you
   - u10a1.ISort you still (hopefully) have
   - ISort.sort: returns a sorted copy of the vector
   - Your MergeSort class should provide a way to select the number of parallel threads

b) 1'000'000 Integers
   - A main class to perform the measurements
   - Here also U10 A1 offers a reference
   - An important indication of your measurements is the number of available CPU cores on your system (Google helps)
   - Don't forget the explanation!
Thread vs. Process / Multithreading vs. Multiprocessing

- A thread is a subset of a process. A process can have many threads
  - E.g.: Process = when opening Microsoft Word
  - E.g.: Thread = a specific path of execution within Word, when you insert an image on your page

- Threads are lightweight compared to processes

- Threads share the same address space and can share both data and code

- Context switching between threads is (usually) less expensive as with processes

- Threads can communicate directly with other threads; Processes (usually) need interprocess communication
Threads in Java

- Two options
  - Extend Thread class
  - Implement the Runnable interface

- Why the two options?
  - In Java, one can implement many interfaces, but can extend only one class!

```java
public class HelloRunnable implements Runnable {
    public void run() {
        System.out.println("Hello from a thread!");
    }
    public static void main(String args[]) {
        new HelloRunnable().start();
    }
}

public class HelloThread extends Thread {
    public void run() {
        System.out.println("Hello from a thread!");
    }
    public static void main(String args[]) {
        (new HelloThread()).start();
    }
}
```
Some more about threads

- How to pause a thread?
  - `Thread.sleep(4000);` // time is measured in milliseconds

- How can a thread wait for another thread?
  - If `t` is a `Thread` object whose thread is currently executing
  - `t.join();`

- How to run a piece of code in the current thread?
  - Just use the `.run()` method

- How to run a piece of code in a new thread?
  - Use the `.start()` method
  - Creates a new thread and then calls the run method
The company “Springli” intends to bring a new chocolate on the market.

Acceptance of all rectangular formats with a maximum of $n$ bits must be tested.

How many formats in terms of $n$ must the company Springli test in the market?

Hint:
- For $n = 1, 2, 3, 4, 5, 6$ exists $1, 3, 5, 8, 10, 14$ formats.
U12.A3 Springli, n = 1

n = 1 \rightarrow 1 \text{ Format}
U12.A3 Springli, n = 2

\[ n = 2 \rightarrow 3 \text{ Formats} \]
U12.A3 Springli, n = 3

n = 3 → 5 Formats
U12.A3 Springli, \( n = 4 \)

\[ n = 4 \rightarrow 8 \text{ Formats} \]
U12.A3 Formula?

- Recursive solution

- $F(n) = F(n-1) + \ldots$ ?
Last but not least

- Last tutorial, no tutorial next week!
  - Only way to get feedback is to submit in Codeboard

- Reversi tournament

- Make sure to fill in the feedback questionnaire

- What about office hours for exam preparation?
  - Nothing planned yet
  - When several of you are interested, I can book a room and we can discuss questions
  - Individual questions, either through e-mail or we can meet in my office
    - E-mail before to arrange an appointment
Have Fun!