Data structures, exercise 1, 19-22 January

Please note: the exercise sessions will start on the first week of lectures.

1. We want to prove that assumption A leads to assertion B, i.e. in terms of proposition logics, $A \rightarrow B$. There are many techniques for proving this. Whatever works best in each case depends on the circumstances.

   - *For deductive proof*, also called direct proof, we assume A and prove that B follows.

     Example: Let $x \in \mathbb{N}$ be uneven. We claim that x^2 is uneven.
     Proof: If $x \in \mathbb{N}$ is uneven, then $x = 2k+1$ for some $k \in \mathbb{N}$. Then $x^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1,$ i.e. x^2 is uneven. $\Box$

   - *In contra-positive proof*, we show that the hypothesis not B leads to not A, i.e. $\neg B \rightarrow \neg A$.

     Example: We claim that if $x > 0$ is an irrational number, then $\sqrt{x}$ is an irrational number.
     Proof: We show that if$\sqrt{x}$ is a rational number, then x is a rational number. This means that $\sqrt{x} = p/q$, for some $p, q \in \mathbb{Z}, q \neq 0$, so $x = \frac{p^2}{q^2}$ ja $p^2, q^2 \in \mathbb{Z}, q^2 \neq 0$. $\Box$

   - *In counter-argument proof*, we show that the assumption $A \wedge \neg B$ leads to a contradiction. This technique is also called indirect proof.

     Example: We claim that if $a + b = 2$, then $a \geq 1$ tai $b \geq 1$.
     Proof: Let us assume that $a + b = 2$ and make the counter-proposition that $a < 1$ ja $b < 1$. Then $2 = a + b < 1 + 1 = 2$, a contradiction. $\Box$

   Explain why, both in mathematical terms and on the level of common sense, why both contra-positive and counter-argument proof are mathematically acceptable ways to prove that $A \rightarrow B$.

   Further, explain why it is enough to show that $A \rightarrow B$ and $B \rightarrow A$ if we want to prove that propositions A and B are equally true, i.e. that $\leftrightarrow B$.

   Simple propositional logic with truth tables could help you with these.

   Of the above methods, the counter-argument will especially be used actively during the course, so though the problem set-up is long, please read it carefully and try to understand the proof methods as well as possible.

2. In addition to the counter-argument proof method, the **induction proof** plays a key part in Data structures. This course will make frequent use of various sum formulas. This formula, for example $\sum_{i=0}^{n} i = 1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}$ is used often. It is usually easiest to prove a sum formula with an induction.

   Prove that $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$

3. Computer scientists may consider a logarithm as follows: the a-based logarithm $\log_a n$ shows us how many times n must be divided by a to result in 1. Example: $\log_2 8$ is 3, because 8/2/2/2 is 1. If we cannot get 1 by even divides, there is a decimal part at the end of the logarithm to represent that last partial division. $\log_2 9$, for example, is around 3.17, because 9/2/2/2 is 1.123, so that after three divisions we still have a way to go to reach 1, but a fourth full division would take us well under it.

   Calculate the following logarithms without a calculator:

   (i) $\log_2 4$
   (ii) $\log_2 32$
   (iii) $\log_3 3$
   (iv) $\log_3 81$
   (v) $\log_7 1$
   (vi) $\log_7 49$
   (vii) $\log_{10} 1000$
   (viii) $\log_{10} 10000$

4. Continuing the previous exercise

   a. Give the rationale for the following calculus rule with the help of consecutive divisions:
      $\log_a(x * y) = \log_a x + \log_a y$
   b. Give the rationale for the following calculus rule with the help of the previous calculus rule: $\log_a x^y = y * \log_a x$

5. The n:th number in the Fibonacci number sequence can be defined as follows

$$F(n) = \begin{cases} 1 & \text{if } n = 1 \text{ or } n = 2 \\ F(n-1) + F(n-2) & \text{if } n > 2 \end{cases}$$

   Use pen and paper to calculate the first 10 numbers in the Fibonacci sequence.

   Create a recursive (static) method `long fib1(n)` that returns the n:th number in the Fibonacci sequence. To recap, a recursive method is one that calls itself.

   Use Java in this exercise. Instead of type `int`, use type `long`, since the sequence will grow very quickly.

It is also easy to find out the n:th number of the Fibonacci sequence without using a recursion.

Create the method `long fib2(n)` that returns the n:th number in the Fibonacci sequence. This method uses a table of length n to support the calculation.

We do not even necessarily need the table. Create another version, long `fib3(n)`, that works without table or recursion.

Which of these programs is the fastest? Which requires the most memory? Which is simplest?

Do not test the programs you create during the course on the servers melkki, melkinpaasi, or melkinkari.

6. Let x be a decimal number and $n \geq 0$ an integer. We can calculate x^n with a recursive algorithm that follows this reduction formula:

$$x^n = \begin{cases} 1 & \text{jos } n = 0 \\ xx^{n-1} & \text{if } n \text{ uneven} \\ x^{n/2}x^{n/2} & \text{if } n \text{ even} \end{cases}$$

   Demostrate how we can calculate 3^{11} using the above formula.

   Use Java to create the method double `power(double x, int n)` that implements the algorithm.

   As you can see in exercises 4 and 5, the recursive solution is not necessarily very fast. How about the solution to this exercise?