Algorithms - Homework

Formatiert: Englisch (USA)

Formatiert: Englisch (USA)

Student	Sascha Feldmann (547307)
Due Date	12-03-2014
Description	Basic Concepts: exponentiation

Task 1: Function func3()

I realized this better recursion as following:

```
/// <summary>
/// This own implementation makes usage of the exponential law x ^ (m * n) = (x ^ m) ^ n.
///
/// Therefore, we try to express x ^ n by x ^ (2 * n/2) = (x ^ 2) ^ n/2.
/// </summary>
/// <param name="x"></param>
/// <param name="n"></param>
/// <returns></returns>
3 Verweise
protected double func3(long x, long n)
{
    if (n == 1)
    {
        return x;
    }
    else
    {
        if (n % 2 == 0)
        {
            return func3( x * x, n / 2);
        }
        else
        {
            return x * func3(x, n - 1);
        }
    }
}
```

Figure 1 - Implementation of func3()

It makes use of the exponential law $x^{n*m} = (x^n)^m$ by working with the "2"-exponentials. In general, you can express the formula x^n by making use of "2": $x^{2*\frac{n}{2}}$ if the exponent n can be divided by 2. Then, the recursive algorithm should only calculate the first part of the formula: x^2 and hand in $\frac{n}{2}$ as new parameter for n.

I will give an example now to show that the number of recursive calls can be reduced in half by the implementation of func3. I want to calculate 2^8:

- **func3(2, 8)** *256*
  - n is even, so express the exponential by using 2
- **#1: func3(2 * 2, 8 / 2) = func3(4, 4)** *256*
  - n is even, so express by using 2
- **#2: func3(4 * 4, 4 / 2) = func3(16, 2)** *256*
  - n is even, so express the exponential again by using 2
- **#3: func3(16 * 16, 2/2) = func3(256, 1)**
  - n is 1, so return x = 256

This best-case example (due the basis of 2) shows us that only 3 iterations are required. Comparing to func2, we would have needed 4 more recursive calls:

- **func2(2, 8)**
- **#1 func3(2, 7)**
- **#2 func3(2, 6)**
- **#3 func3(2, 5)**
- **#4 func3(2, 4)**
- **#5 func3(2, 3)**
- **#6 func3(2, 2)**
- **#7 func3(2,1)**

Task 2: Complexity of func3()

Let's take a look at the best case: we want to calculate the 2^8 which is a best case cause 8 = 2^3, so 8 has an integer value for the logarithm of 2.

You can identify recurrence function calls for the first iteration of func3:

$$T(n) = 1 + T(\frac{n}{2})$$

In the next iteration we will have:

$$T(n) = 1 + T(\frac{n}{4})$$

So we can express T(n) as:

$$\begin{aligned} T(n) &= 1 + T\left(\frac{n}{2}\right) \\ &= 1 + 1 + T\left(\frac{n}{4}\right) \\ &= 1 + 1 + 1 + T\left(\frac{n}{8}\right) \\ &\qquad \dots \\ &= log2(n) + T\left(\frac{n}{n}\right) \\ &= log2(n) \end{aligned}$$

This means that func3(x, n) implementation is 0(log (n)).