Assignment 1

Surname: **Krasser**
First name: **Konstantin**
Matr.No.: **12028653**

Exercise 1: Mathematical induction

(a) $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

Base: $n = 1$, LHS = 1, RHS = 1. Holds.
Inductive step: Assume $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$. Then,

$$\sum_{i=1}^{n+1} i = \sum_{i=1}^{n} i + (n+1) = \frac{n(n+1)}{2} + (n+1) = \frac{(n+1)(n+2)}{2}.$$

Holds for $n + 1$. Proven.

(b) $\sum_{i=1}^{n} \sum_{j=1}^{i} j = \frac{1}{6}n(n^2 + 3n + 2)$

Base: $n = 1$, LHS = 1, RHS = 1. Holds.
Inductive step: Assume $\sum_{i=1}^{n} \sum_{j=1}^{i} j = \frac{1}{6}n(n^2 + 3n + 2)$. Then,

$$\sum_{i=1}^{n+1} \sum_{j=1}^{i} j = \sum_{i=1}^{n} \sum_{j=1}^{i} j + \sum_{j=1}^{n+1} j.$$

Using $\sum_{j=1}^{n+1} j = \frac{(n+1)(n+2)}{2}$,

$$\sum_{i=1}^{n+1} \sum_{j=1}^{i} j = \frac{1}{6}n(n^2 + 3n + 2) + \frac{(n+1)(n+2)}{2} = \frac{1}{6}(n+1)((n+1)^2 + 3(n+1) + 2).$$

Holds for $n + 1$. Proven.

(c) $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$

Base: $n = 0$, LHS = 1, RHS = 1. Holds.
Inductive step: Assume $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$. Then,

$$\sum_{i=0}^{n+1} 2^i = \sum_{i=0}^{n} 2^i + 2^{n+1} = (2^{n+1} - 1) + 2^{n+1} = 2^{n+2} - 1.$$

Holds for $n + 1$. Proven.

(d) $\sum_{i=1}^{n} \sum_{j=i}^{n} j = \frac{1}{6}n(2n^2 + 3n + 1)$

Base: $n = 1$, LHS = 1, RHS = 1. Holds.
Inductive step: Assume $\sum_{i=1}^{n} \sum_{j=i}^{n} j = \frac{1}{6}n(2n^2 + 3n + 1)$. Then,

$$\sum_{i=1}^{n+1} \sum_{j=i}^{n+1} j = \sum_{i=1}^{n} \sum_{j=i}^{n} j + \sum_{j=n+1}^{n+1} j.$$

Using $\sum_{j=n+1}^{n+1} j = n + 1$,

$$\sum_{i=1}^{n+1} \sum_{j=i}^{n+1} j = \frac{1}{6}n(2n^2 + 3n + 1) + (n+1).$$

After simplification,

$$\frac{1}{6}(n+1)(2(n+1)^2 + 3(n+1) + 1).$$

Holds for $n + 1$. Proven.

Exercise 2: Pseudocode and Runtime Analysis

```
Algorithm 1 algorithm(A)
for π ∈ Perm(n) do
    if procedure(π(A)) then Return π(A)
    end if
end for
```

Perm(n) denotes the set of permutations on n elements, in particular the for-loop considers every permutation just once. `procedure` is defined as follows.

```
procedure(A)
for i ← 0 to n − 2 do
    if A[i] > A[i + 1] then
        return False
    end if
    return True
end for
```

Abbildung 1: Runtime Analysis

- What is the output of algorithm1? Explain intermediate steps the algorithm does, in particular procedure(A)
  - The procedure checks if an array is sorted by iterating through each element and checking if it's $\leq$
- What is the runtime of algorithm1? Analyze the worst and the best case
  - Best runtime: is when the array is already sorted, in which case the runtime is $O(n)$.
  - Worst runtime: is when the array is sorted in descending order, in which case the runtime is $O(n^2)$.

Exercise 3: Runtime Analysis

```
algorithmA(n)
int count1 = 0;
for i = 1; i ≤ n; i++ do
    for j = 1; j ≤ i; j++ do
        Print(count1);
        count1++;
    end for
end for
return count1;
```

```
algorithmB(n)
int count2 = 0;
int x = 0;
for i = 1; i ≤ n; i = 2 · i do
    for j = 1; j ≤ i; j++ do
        count2 = x;
        count2++;
        x = count2;
    end for
end for
return count2;
```

```
algorithmC(n)
return algorithmA(n)+algorithmB(10n);
```

Abbildung 2: Runtime Analysis

- What is the output of each of the algorithms?
  - **a)** $\frac{n(n+1)}{2}$
  - **b)** 0
    Always 0, since it's initialized as 0
  - **c)** $\frac{n(n+1)}{2}$
    Since **Algorithm B(10n)**'s output is always 0
- What is the runtime of each of the algorithms?
  - **a)** $O(n^2)$
  - **b)** $O(n)$
  - **c)** $O(n^2) + O(n) = O(n^2)$

Exercise 4: Landau Notation

Definition of the O-notation to prove the following

(a) $0.01 \log_c n = \Theta(\ln n)$ for any $c > 1$

By change of base formula, $\log_c n = \frac{\ln n}{\ln c}$, so $0.01 \log_c n = \frac{0.01}{\ln c} \ln n$. Since $\frac{0.01}{\ln c}$ is a constant, the result follows.

(b) $\max\{f(n), g(n)\} = \Theta(f(n) + g(n))$

Since $f(n) \leq f(n)+g(n)$ and $g(n) \leq f(n)+g(n)$, we have $\max\{f(n), g(n)\} \leq f(n)+g(n)$. Ergo, $f(n)+g(n) \leq 2\max\{f(n), g(n)\}$, proving the claim.

Use the limit criteria to prove the following

(c) $(\log_2 n)^2 = \Omega(2^{\log_2(n^2)})$

Using $2^{\log_2(n^2)} = n^2$, we need to show $(\log_2 n)^2 = \Omega(n^2)$. The limit criterion gives:

$$\lim_{n\to\infty} \frac{(\log_2 n)^2}{n^2} = 0,$$

which contradicts $\Omega(n^2)$. Thus, the claim is false.

(d) $2n^3 + 4n^2 + 7\sqrt{n} = O(n^3)$

Taking the limit:

$$\lim_{n\to\infty} \frac{2n^3 + 4n^2 + 7\sqrt{n}}{n^3} = 2 + \frac{4}{n} + \frac{7\sqrt{n}}{n^3}.$$

Since all terms except 2 vanish as $n \to \infty$, the function is $O(n^3)$.

Exercise 5: Average Runtime of Linear Search