Problems Solved:

46	47	48	49	50

Name:

Matrikel-Nr.:

Problem 46. Let $T(n)$ be the total number of times that the instruction `a[i,j] = a[i,j] + 1` is executed during the execution of `P(n,0,0)`.

```
procedure P(n, x, y)
    if n >= 1 then
        for (i = x; i < x+n; i++)
            for (j = y; j < y+n; j++)
                a[i,j] = a[i,j] + 1
        h = floor( n / 2)
        P(h, x,   y  )
        P(h, x+h, y  )
        P(h, x,   y+h)
        P(h, x+h, y+h)
    end if
end procedure
```

1. Compute $T(1)$, $T(2)$ and $T(4)$.
2. Give a recurrence relation for $T(n)$.
3. Solve your recurrence relation for $T(n)$ in the special case where $n = 2^m$ is a power of two, i.e. derive a guess for and explicit expression of $T(2^m)$ and then prove this formula by induction.
4. Use the Master Theorem to determine asymptotic bounds for $T(n)$.

Solution of Problem 46:

1. $T(1) = 1$, $T(2) = 8$, $T(4) = 48$
2. $$T(n) = 4T(\lfloor n/2 \rfloor) + n^2$$
3. For powers of two,
$$T(2^m) = 2^{2m} + 4T(2^{m-1})$$
(You may want to view this as a recursion for $t(m) := T(2^m)$.) Unfolding this recurrence twice gives
$$T(2^m) = 4^m + 4 \times (4^{m-1} + 4 \times T(2^{m-2})$$
which can be simplified. Continuing the pattern, unfolding the recurrence k times gives
$$T(2^m) = k4^m + 4^k T(2^{m-k}).$$
In particular, for $k = m$ we get $T(2^m) = m4^m + 4^m T(1)$. Since $T(1) = 1$, we get $T(2^m) = (m+1)4^m$.

We must prove this formula by induction on m. If m=0, we get $T(1) = T(2^0) = (0+1)4^0 = 1$, which is obviously correct. Assume that $t \in \mathbb{N}$ is an arbitrary but fixed number and $T(2^t) = (t+1)4^t$. Then $T(2^{t+1}) = 4^{t+1} + 4T(2^t)$ by the recurrence for T. Plugging in the induction hypothesis, we get $T(2^{t+1}) = 4^{t+1} + 4 \times (t+1)4^t = ((t+1)+1)4^{t+1}$. By induction principle, $T(2^m) = (m+1)4^m$ holds for every $m \in \mathbb{N}$.

4.
$$T(n) = 2^2 T(n/2) + \Theta(n^2)$$
Divisor for divide-and-conquer: $b = 2$. Number of new subproblems: $a = 4 = 2^2$. “Critical exponent”: $\log_b a = \log_2 4 = 2$. Indeed
$$n^2 = \Theta(n^{\log_b a}).$$
so we are in the case where all $\log_b n = \Theta(\log n)$ layers of the recursion tree contribute and we get
$$T(n) = \Theta(n^2 \log n).$$
This result agrees with what we have found in (3).

Problem 47. Let $T(n)$ be the number of multiplications carried out by the following Java program.

```
int a, b, i, product, max;
max = 1;
a = 0;
while ( a < n ) {
  b = a;
  while (b <= n) {
    product = 1;
    i = a;
    while (i < b) {
      product = product * factors[i];
      i = i + 1; }
    if (product > max) { max = product; }
    b = b + 1; }
  a = a + 1; }
```

1. Determine $T(n)$ exactly as a nested sum.
2. Determine $T(n)$ asymptotically using Θ-Notation by a derivation that justifies your result. In your derivation, you may use the asymptotic equation
$$\sum_{k=0}^{n} k^m = \Theta(n^{m+1}) \text{ for } n \to \infty$$
for fixed $m \geq 0$ which follows from approximating the sum by an integral:
$$\sum_{k=0}^{n} k^m \simeq \int_0^n x^m \, dx = \frac{1}{m+1} n^{m+1} = \Theta(n^{m+1})$$

Solution of Problem 47:

1.

line	frequency of execution
$2, 3$	1
4	$\sum_{0\le a<n+1} 1$
$5, 14$	$\sum_{0\le a<n} 1$
6	$\sum_{0\le a<n} \sum_{a\le b\le n+1} 1$
$7, 8, 12, 13$	$\sum_{0\le a<n} \sum_{a\le b\le n} 1$
9	$\sum_{0\le a<n} \sum_{a\le b\le n} \sum_{a\le i<b+1} 1$
$10, 11$	$\sum_{0\le a<n} \sum_{a\le b\le n} \sum_{a\le i<b} 1$

2. See line 10 above:

$$T(n) = \sum_{a=0}^{n-1} \sum_{b=a}^{n} \sum_{i=a}^{b-1} 1$$

To get the correct answer $T(n) = \Theta(n^3)$ we evaluate

$$T(n) = \sum_{0\le a<n} \sum_{a\le b\le n} \sum_{a\le i<b} 1$$

starting with the innermost sum:

$$\sum_{a\le i<b} 1 = b - a$$

We proceed with the sum in the middle, starting with a shift of the summation index b, i.e., we replace b by $a + k$.

$$\sum_{a\le b\le n} (b-a) = \sum_{0\le b-a\le n-a} (b-a) = \sum_{0\le k\le n-a} k = \frac{1}{2}(n-a)(n-a+1)$$

$$\frac{1}{2}(n-a)(n-a+1) = \frac{1}{2}(n^2 - 2na + a^2 + n - a)$$

After splitting the sum $\sum_{0\le a<n} \frac{1}{2}(n^2 - 2na + a^2 + n - a)$ and pulling out constant factors (i.e., factors free of the summation index a), all that remains are the sums $\sum_{a=0}^{n-1} a^m$ for $m = 0, 1, 2$. The asymptotics of these sums is given by the hint, and the final result is $\Theta(n^3)$. *Remark:* To shorten the calculation, it is tempting to use Θ-notation already in the summands. But those Θ's would refer to limits involving the summation indices instead of the limit $n \to \infty$. As the limit taken is suppressed in Θ-notation, that would be most confusing for the uninitiated.

Problem 48. Consider the following pseudo code of an implementation of a FIFO (first in first out) queue with two functions enqueue and dequeue.

```
input  := EMPTYLIST
output := EMPTYLIST
function enqueue(e, input, output) { push(e, input) }
function dequeue(input, output) {
```

```
    if isempty(output) {
        while not isempty(input) { push(pop(input), output) }
    }
    pop(output)
}
```

Analyze its amortized cost of these functions by (a) the aggregate method and (b) the potential method.
Here,

- push(e, L) is the operation of adding an element e to the front of a list L,
- isempty(L) returns TRUE if the list L is empty,
- pop(L) is the operation that removes the first element of a list L and returns it.

All these operations are assumed to cost constant time.
In the code above, a queue is represented by a pair (input, output). Putting a new element into the queue via enqueue, first puts it to the front of input. Only when an element is requested via a call to dequeue, elements are moved from input to output list, thus effectively reversing input so that in total the queue returns its elements in a FIFO principle.
Hint: For the potential method you might want to consider the function Φ such that for a queue q that is represented by the pair (input, output) of two lists, $\Phi(q)$ is the size of the input list.

Solution of Problem 48:

(a) Aggregate method:
Let us assume that in the sequence of n operations occur k dequeue operations. Let the number of enqueue operations after the $(i-1)$-th dequeue operation and before the i-th dequeue operation be denoted by r_i $(i = 1, \ldots, k)$. And let $r_{k+1} \geq 0$ be such that $n = k + \sum_{i=1}^{k+1} r_i$.
The size of the input list before the i-th dequeue operation is r_i. It is clear that the i-th dequeue operation has cost $E \cdot r_{i-1}$ for some constant E. Since the enqueue operation has constant cost C, we get for the total complexity over the n operations.

$$
\begin{aligned}
T(n) &= \sum_{i=1}^{k}(C \cdot r_i + E \cdot r_i) + C \cdot r_{k+1} \\
&\leq \sum_{i=1}^{k+1}(C + E) \cdot r_i \\
&\leq (C + E) \cdot k + \sum_{i=1}^{k+1}(C + E) \cdot r_i \\
&= (C + E) \cdot n = O(n).
\end{aligned}
$$

Thus, the amortized cost of a single operation is $O(1)$.

(b) Potential method:
Take $C \geq 0$ such that it bounds the constant time of enqueue, and the time of dequeue in the constant case and $C \cdot n$ bounds the time of dequeue in the linear case. Furthermore, let $\Phi(q)$ be the size of the `input` list of the queue q.

Let c_i be the actual cost and $\hat{c}_i$ be the amortized cost of the i-th operation, and let q_i be the queue after the i-th operation ($i = 1, \ldots, n$).

Then we have

- for `enqueue`:

$$\begin{aligned}\hat{c}_i &= c_i + C(\Phi(q_i) - \Phi(q_{i-1})) \\ &\leq C + C((n_i + 1) - n_i) = 2C\end{aligned}$$

  Where n_i denotes the size of the input list of q_i.

- for `dequeue`:

$$\begin{aligned}\hat{c}_i &= c_i + \Phi(q_i) - \Phi(q_{i-1}) \\ &\leq C \cdot n_i + C(0 - n_i) = 0\end{aligned}$$

  Where n_i denotes the size of the input list of q_i.

Thus, the amortized cost of one operation is $O(1)$.

Problem 49. Consider a RAM program that evaluates the value of $\sum_{i=1}^{n} i^2$ in the naive way (by iteration). Analyze the worst-case asymptotic time and space complexity of this program assuming the existence of operations `ADD r` and `MUL r` for the addition and multiplication of the accumulator with the content of register r.

1. Determine a Θ-expression for the number $S(n)$ of registers used in the program with input n (space complexity).
2. Determine a Θ-expression for the number $T(n)$ of instructions executed for input n (time complexity in constant cost model),
3. Assume a simplified version of the logarithmic cost model of a RAM where the cost of every operaton is proportional to the length of the arguments involved. In particular, if a is the (bit) length of the accumulator and l is the (bit) length of the content of register r then `MUL r` costs $a + l$ and `ADD r` costs $\max(a, l)$.

   Determine the asymptotic costs $C(n)$ (using O-notation) of the program for input n.

As usual, you must give appropriate justification for each of your results.

Solution of Problem 49:

1. There is only need for accumulator, a register for the resulting sum and a register for the summation index. So we get $S(n) = \Theta(1)$.
2. Since multiplication can be done by just one operation, we only have to iterate over the summation index, i.e., $T(n) = \Theta(n)$.
3. The cost of the i-th iteration is:

   multiplication of accumulator with Register 2 (holding i)
   - `LOAD 2` $\ldots \log(i)$
   - `MUL 2` $\ldots \log(i) + \log(i)$

   addition of partial sum with i^2
   - accu already has i^2
   - partial sum $s = \sum_{k=1}^{i-1} i^2 = \Theta(i^3)$
   - cost of addition: $\max(\log(i^2), \log(s)) = \Theta(3\log(i))$

   We have to compute the sum

$$\begin{aligned} C(n) &= O\left(\sum_{i=1}^{n} (\log(i^2) + 3\log(i))\right) \\ &\leq O\left(\sum_{i=1}^{n} (\log(n^2) + 3\log(n))\right) \\ &= O(n \log n) \end{aligned}$$

Problem 50. Take the following recursive program.

```
1 f(n,b) ==
2     if n < 1 then return 0
3     d := floor(n/3)
4     return b + f(d,1) + 2*f(d,2)
```

Let $C(n)$ be the number of comparisons executed in line 2 while running $f(n, 0)$ for some positive integer n.

1. Write down a recurrence for C and determine enough initial values.
2. Solve that recurrence for the given initial values and arguments n of the form $n = 3^m$, i.e., derive a guess for a closed form expression for $C(3^m)$.
3. Prove by induction that your guess is correct.

Solution of Problem 50:

1. $C(n) = 1 + 2 \cdot C(\lfloor \frac{n}{3} \rfloor)$, $C(0) = 1$.
2. Let $n = 3^m$ for some $m \in \mathbb{N}$. Then

$$\begin{aligned} C(3^m) &= 1 + 2C(3^{m-1}) = \ldots = 1 + 2 + \ldots 2^{m-1} + 2^m C(3^{(m-m)}) \\ &= \sum_{k=0}^{m} 2^k + 2 \cdot 2^m \\ &= 2^{m+1} - 1 + 2^{m+1} = 2^{m+2} - 1 \end{aligned}$$

3. We show $C(3^m) = 2^{m+2} - 1$ by induction on m. For $m = 0$ we have $C(3^0) = 1 + 2C(\lfloor \frac{1}{3} \rfloor) = 3 = 2^{0+2} - 1$, i.e., we have shown the induction base. Now assume that for an arbitrary but fixed k it holds $C(3^m) = 2^{m+2} - 1$ (induction hypothesis). Then

$$\begin{aligned} C(3^{k+1}) &= 1 + 2C\left(\left\lfloor \frac{3^{k+1}}{3} \right\rfloor\right) = 1 + 2C(3^k) \\ &= 1 + 2(2^{k+2} - 1) = 1 + 2^{(k+1)+2} - 2 = 2^{(k+1)+2} - 1. \end{aligned}$$

Therefore, by induction principle, $C(3^m) = 2^{m+2} - 1$ holds for all natural numbers m.