Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Departement of Computer Science
Markus Püschel, David Steurer
Johannes Lengler, Gleb Novikov, Chris Wendler

30. September 2019

Algorithms & Data Structures — Exercise sheet 2 — HS 19

Exercise Class (Room & TA): _____

Submitted by: _____

Peer Feedback by: _____

Points: _____

The solutions for this sheet are submitted at the beginning of the exercise class on October 7th.

Exercises that are marked by * are challenge exercises. They do not count towards bonus points.

Exercise 2.1 *Iterative squaring* **(1 point)**.

In this exercise you are going to develop an algorithm to compute powers a^n, with $a \in \mathbb{Z}$ and $n \in \mathbb{N}$, efficiently. For this exercise, we will treat multiplication of two integers as a single elementary operation, i.e., for $a, b \in \mathbb{Z}$ you can compute $a \cdot b$ using one operation.

a) Assume that n is even, and that you already know an algorithm $A_{n/2}(a)$ that efficiently computes $a^{n/2}$, i.e., $A_{n/2}(a) = a^{n/2}$. Given the algorithm $A_{n/2}$, design an efficient algorithm $A_n(a)$ that computes a^n.

Solution:

Algorithm 1 $A_n(a)$

$x \leftarrow A_{n/2}(a)$

return $x \cdot x$

b) Let $n = 2^k$, for $k \in \mathbb{N}_0$. Find an algorithm that computes a^n efficiently. Describe your algorithm using pseudo-code.

Solution:

Algorithm 2 Power(a, n)

```
if n = 1 then
    return a
else
    x ← Power(a, n/2)
    return x · x
```

c) Determine the number of elementary operations (i.e., integer multiplications) required by your algorithm for part b) in $\mathcal{O}$-notation. You may assume that bookkeeping operations don't cost anything. This includes handling of counters, computing $n/2$ from n, etc.

Solution: Let $T(n)$ be the number of elementary operations that the algorithm from part b) performs on input a, n. Then

$$T(n) \leq T(n/2) + 1 \leq T(n/4) + 2 \leq T(n/8) + 3 \leq \ldots \leq T(1) + \log_2 n - 1 \in \mathcal{O}(\log n)\,.$$

d) Let Power(a, n) denote your algorithm for the computation of a^n from part b). Prove the correctness of your algorithm via mathematical induction for all $n \in \mathbb{N}$ that are powers of two.

In other words: show that Power$(a, n) = a^n$ for all $n \in \mathbb{N}$ of the form $n = 2^k$ for some $k \in \mathbb{N}_0$.

- **Base Case.**
  Let $k = 0$. Then $n = 1$ and Power$(a, n) = a = a^1$.
- **Induction Hypothesis.**
  Assume that the property holds for some positive integer k. That is, Power$(a, 2^k) = a^{2^k}$.
- **Inductive Step.**
  We must show that the property holds for $k + 1$.

$$\text{Power}(a, 2^{k+1}) = \text{Power}(a, 2^k) \cdot \text{Power}(a, 2^k) \overset{\text{I.H.}}{=} a^{2^k} \cdot a^{2^k} = a^{2^{k+1}}.$$

  By the principle of mathematical induction, this is true for any integer $k \geq 0$ and $n = 2^k$.

*e) Design an algorithm that can compute a^n for a general $n \in \mathbb{N}$, i.e., n does not need to be a power of two.

Hint: *Generalize the idea from part a) to the case where n is odd, i.e., there exists a $k \in \mathbb{N}$ such that $n = 2k + 1$.*

Solution:

Algorithm 3 Power(a, n)

```
if n = 1 then
    return a
else
    if n is odd then
        x ← Power(a, (n − 1)/2)
        return x · x · a
    else
        x ← Power(a, n/2)
        return x · x
```

*f) Prove correctness of your algorithm in e) and determine the number of elementary operations in $\mathcal{O}$-Notation. As before, you may assume that bookkeeping operations don't cost anything.

Solution: Let's prove correctness.

- **Base Case.**
  Let $n = 1$. Then Power$(a, n) = a = a^1$.

- **Induction Hypothesis.**
  Assume that the property holds for all positive integers $m < n$. That is, $\text{Power}(a, m) = a^m$.
- **Inductive Step.**
  We must show that the property holds for n. If n is even,

$$\text{Power}(a, n) = \text{Power}(a, n/2) \cdot \text{Power}(a, n/2) \overset{\text{I.H.}}{=} a^{n/2} \cdot a^{n/2} = a^n.$$

  If n is odd,

$$\text{Power}(a, n) = a \cdot \text{Power}(a, (n-1)/2) \cdot \text{Power}(a, (n-1)/2) \overset{\text{I.H.}}{=} a \cdot a^{(n-1)/2} \cdot a^{(n-1)/2} = a^n.$$

  By the principle of mathematical induction, this is true for any integer $n \geq 1$.

Let $T(n)$ be the number of elementary operations that the algorithm Power performs on input a, n. Let's prove by induction that $T(n) \leq 2 \log_2 n$.

- **Base Case.**
  Let $n = 1$. Then $T(n) = 0 \leq 2 \log_2 n$.
- **Induction Hypothesis.**
  Assume that the property holds for all positive integers $m < n$. That is, $T(m) \leq 2 \log_2 m$.
- **Inductive Step.**
  We must show that the property holds for n. If n is even,

$$T(n) \leq T(n/2) + 1 \overset{\text{I.H.}}{\leq} 2 \log_2 n/2 + 1 < 2 \log_2 n.$$

  If n is odd,

$$T(n) \leq T((n-1)/2) + 2 \overset{\text{I.H.}}{\leq} 2 \log_2 (n-1)/2 + 2 < 2 \log_2 n.$$

  By the principle of mathematical induction, this is true for any integer $n \geq 1$.

Exercise 2.2 *Induction.*

a) Prove via mathematical induction that for any positive integer n,

$$(1 + x)^n = \sum_{i=0}^{n} \binom{n}{i} x^i \,,$$

where n choose i is defined by

$$\binom{n}{i} := \frac{n!}{(n-i)!\, i!} = \frac{n \cdot (n-1) \cdots (n-i+1)}{i \cdot (i-1) \cdots 1}.$$

We set $\binom{n}{k} := 0$ for $k > n$ and for $k < 0$. Notice that $\binom{n}{0} = 1$ because $0! = 1$.

Hint: *Show first the identity*

$$\binom{n}{i} + \binom{n}{i-1} = \binom{n+1}{i}.$$

It can be obtained by fractional arithmetic and will be useful for the proof.

Solution: Proof of hint:

$$\begin{aligned}\binom{n}{i} + \binom{n}{i-1} &= \frac{n!}{(n-i)!\,i!} + \frac{n!}{(i-1!)(n-i+1)!} \\ &= \frac{(n-i+1)\cdot n! + i\cdot n!}{i!\,(n-i+1)!} \\ &= \frac{(n+1)!}{i!\,(n-i+1)!} \\ &= \binom{n+1}{i}.\end{aligned}$$

- **Base Case.**
  Let $n = 1$. Then $(1+x)^1 = \binom{1}{0}x^0 + \binom{1}{1}x^1 = \sum_{i=0}^{n}\binom{n}{i}x^i$.
- **Induction Hypothesis.**
  Assume that the property holds for some positive integer k. That is:

$$(1+x)^k = \sum_{i=0}^{k}\binom{k}{i}x^i.$$

- **Inductive Step.**
  We must show that the property holds for $k+1$.

$$\begin{aligned}(1+x)^{k+1} &= (1+x)(1+x)^k \\ &\overset{I.H.}{=} (1+x)\sum_{i=0}^{k}\binom{k}{i}x^i \\ &= \Big(\sum_{i=0}^{k}\binom{k}{i}x^i\Big) + \Big(\sum_{i=0}^{k}\binom{k}{i}x^{i+1}\Big) \\ &= \Big(\sum_{i=0}^{k}\binom{k}{i}x^i\Big) + \Big(\sum_{i=1}^{k+1}\binom{k}{i-1}x^i\Big) \\ &= \binom{k}{0}x^0 + \sum_{i=1}^{k}\Big(\binom{k}{i}x^i + \binom{k}{i-1}x^i\Big) + \binom{k}{k}x^{k+1} \\ &= \binom{k+1}{0}x^0 + \sum_{i=1}^{k}\binom{k+1}{i}x^i + \binom{k+1}{k+1}x^{k+1} = \sum_{i=0}^{k+1}\binom{k+1}{i}x^i.\end{aligned}$$

By the principle of mathematical induction, this is true for any positive integer n.

b) Given is a map that is divided into regions by n (pairwise different) straight lines. You want to color the regions on the map (i.e., the areas bordered by the lines), such that no two neighboring regions (i.e., regions that share a common segment of a line as a border) get the same color.

Prove by mathematical induction on n, that you can color every such map with 2 colors.

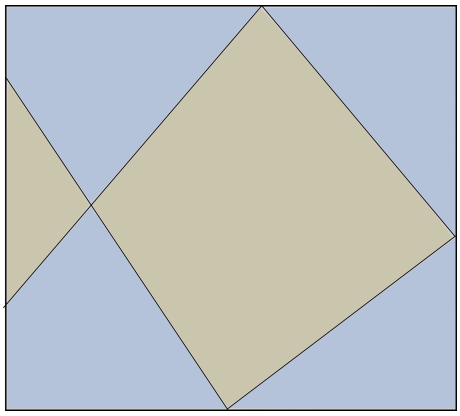

Figure 1: An example map. Defined by $n = 4$ straight lines. The resulting 6 regions are colored by two colors.

- **Base Case.**
  Let $n = 1$. In this case, there are exactly two regions. We color one black and the other one white.
- **Induction Hypothesis.**
  Suppose that a map defined by k lines can be colored with two colors, for some positive integer k.
- **Inductive Step.**
  We must show that the property holds for $k + 1$. Consider any line l (out of the $(k + 1)$ lines). If we remove this line from the map, by induction hypothesis, we can color the map using two colors. l divides the map into a left part and a right part. Now if we take the coloring of the map without l and flip the signs in the left part, we get a coloring of the map with l. Indeed, take any two neighbouring regions. If they don't share a common segment that is a part of l, these regions are neighbouring on the map without l and they are both in the same part (left or right), so they should have different colors. If these regions share a common segment that is a part of l, they have different colors since before flipping signs they had the same color.

Exercise 2.3 *$\mathcal{O}$-Notation of lecture.*

Recall the definition of $\mathcal{O}$-notation introduced in the lecture:

$$\mathcal{O}(g) = \{f : N \to \mathbb{R}^+ | \text{ There exists a constant } C > 0 \text{ such that for all } n \in N,\ f(n) \leq C \cdot g(n)\}\,.$$

Prove your statements directly with this definition. In particular, you should *not* use theorems from Exercise sheet 0.

For example, to show that $4n + 1 \in \mathcal{O}(n)$ you would look for a constant C such that $4n + 1 \leq Cn$ for all $n \in \mathbb{N}$. In this example $C = 5$ is such a constant.

a) Write the following in the asymptotic $\mathcal{O}$-notation. Your answer should be simplified as much as possible. Unless otherwise stated, we assume $N = \mathbb{N} = \{1, 2, 3, \dots\}$. You do not need to check that the involved functions take values in $\mathbb{R}^+$.

   1) $5n^3 + 40n^2 + 100$.
   Solution: $40n^2 \leq 40n^3$, $100 \leq 100n^3$, hence $5n^3 + 40n^2 + 100 \leq 145n^3$ for all $n \geq 1$, so $5n^3 + 40n^2 + 100 = \mathcal{O}(n^3)$

2) $1^2 + 2^2 + 3^2 + \ldots + n^2$.
Solution: $\mathcal{O}(n^3)$

Recall from Homework 0 that $1^2 + 2^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6} = \frac{2n^3+3n^2+n}{6}$. We claim that this is in $\mathcal{O}(n^3)$. Indeed, for all $n \geq 1$ we have $n^2 \leq n^3$ and $n \leq n^3$, and therefore,

$$\frac{2n^3 + 3n^2 + n}{6} \leq \frac{2n^3 + 3n^3 + n^3}{6} = n^3.$$

The condition holds, e.g., for $C = 1$. Hence, $1^2 + 2^2 + 3^2 + ... + n^2 = \mathcal{O}(n^3)$.

3) $2n \log_3 n^4$ with $N = \{2, 3, 4, \ldots\}$.
Solution: $\mathcal{O}(n \log n)$

We must show that for some positive C, and for all $n > 1$,

$$2n \log_3 n^4 \leq Cn \log n.$$

This follows from a direct calculation. We use the formula for base change, $\log_b n = \frac{\log_a n}{\log_a b}$, and obtain

$$2n \log_3 n^4 = 8n \log_3 n = 8n \frac{\log n}{\log 3} n = \frac{8}{\log 3} n \log n,$$

so the condition holds with $C := 8/\log 3$.

b) Prove that if $f_1(x), f_2(x) \leq \mathcal{O}(g(x))$, then $f_1(x) + f_2(x) \leq \mathcal{O}(g(x))$.

Solution: Since both f_1 and f_2 are $\mathcal{O}(g(x))$, we know that for all $x \geq 1$, there exist positive real numbers C_1, C_2 that:

$$f_1(x) \leq C_1 g(x)$$

and

$$f_2(x) \leq C_2 g(x).$$

Then $f_1(x) + f_2(x) \leq C_1 g(x) + C_2 g(x) = (C_1 + C_2) g(x)$ Thus, if we set $C_3 := C_1 + C_2$ then for all $x \geq 1$,

$$f_1(x) + f_2(x) \leq C_3 g(x).$$

Thus we have shown that $f_1(x) + f_2(x) \leq \mathcal{O}(g(x))$.

c) Let $f_1(x), f_2(x), g(x) > 0$. Prove or disprove the following.

1) If $f_1(x), f_2(x) \leq \mathcal{O}(g(x))$ then $\frac{f_1(x)}{f_2(x)} \leq \mathcal{O}(1)$.

Solution: We will disprove this. Let $f_1(x) = x$, $f_2(x) = \sqrt{x}$, and $g(x) = x$. $x \leq \mathcal{O}(x)$, $\sqrt{x} \leq \mathcal{O}(x)$, but $\frac{f_1(x)}{f_2(x)} = \sqrt{x} \not\leq O(1)$.

2) If $f_1(x) \leq \mathcal{O}(g(x))$ and $f_2(x) \leq \mathcal{O}(\frac{1}{g(x)})$, then $f_1(x) f_2(x) \leq \mathcal{O}(1)$.

Solution: Because $f_1(x) \leq \mathcal{O}(g(x))$, there exists C_1 such that $f_1(x) \leq C_1 g(x)$ for all $x \geq 1$.

Because $f_2(x) \leq \mathcal{O}(\frac{1}{g(x)})$, there exists C_2 such that $f_2(x) \leq C_2 \frac{1}{g(x)}$ for all $x \geq 1$.

Assume $x \geq 1$. Then

$$f_1(x) f_2(x) \leq (C_1 g(x)) \left(C_2 \frac{1}{g(x)} \right) = C_1 C_2$$

Thus:

$$\exists C_3 > 0. \forall x \geq 1, f_1(x) f_2(x) \leq C_3 * 1$$

In this case $C_3 = C_1 C_2$. From this we have shown that. $f_1(x) f_2(x) \leq \mathcal{O}(1)$.

Exercise 2.4 *Karatsuba algorithm for polynomial multiplication* **(2 points)**.

In this exercise you should design an efficient algorithm for multiplying polynomials with integer coefficients. For this exercise, we will treat addition, subtraction, and multiplication of two integers as a single elementary operation. As before, you may ignore any further bookkeeping costs.

a) How many elementary operations are necessary to add or subtract two polynomials of degree k?

Solution: Since a polynomial of degree k has $k+1$ coefficients, we need $k+1$ elementary operations to add or subtract two polynomials of degree k.

b) Assume that you already know an efficient algorithm A_k for the multiplication of two polynomials of degree at most $k-1$. Design an efficient algorithm A_{2k} for multiplying two polynomials of degree at most $2k-1$ that calls A_k at most three times as a subroutine and uses at most $\mathcal{O}(k)$ additional elementary operations. You should describe your algorithm in pseudo-code.

Hint: *Let the input polynomials $P(x), Q(x)$ of degree at most $2k-1$ be*

$$P(x) = P_0 x^0 + P_1 x^1 + \cdots + P_{2k-1} x^{2k-1},$$
$$Q(x) = Q_0 x^0 + Q_1 x^1 + \cdots + Q_{2k-1} x^{2k-1}.$$

Then write the polynomials as $P(x) = x^k P^*(x) + P^{**}(x)$ and $Q(x) = x^k Q^*(x) + Q^{**}(x)$, where P^*, P^{**}, Q^* and Q^{**} are polynomials of degree at most $k-1$. Now use the same trick as in Karatsuba's algorithm.

Solution:

Algorithm 4 $A_{2k}(P, Q)$

for $i \in \{0, \ldots, k-1\}$ **do**
  $P_i^* \leftarrow P_{k+i}$
  $P_i^{**} \leftarrow P_i$
  $Q_i^* \leftarrow Q_{k+i}$
  $Q_i^{**} \leftarrow Q_i$
$L \leftarrow A_k(P^*, Q^*)$
$R \leftarrow A_k(P^{**}, Q^{**})$
$M \leftarrow A_k(P^* + P^{**}, Q^* + Q^{**}) - L - R$
return $x^{2k} L + x^k M + R$

c) Provide a bound for the amount of elementary operations $T(2k)$ required by your algorithm A_{2k} in terms of the amount of elementary operations $T(k)$ required by the algorithm A_k. That is, give a bound of the form $T(2k) \leq a \cdot T(k) + b(k)$. Give as precise bound as possible.

Solution: We call A_k three times, so $a = 3$. We perform $2k$ additions of integers to compute $P^* + P^{**}$ and $Q^* + Q^{**}$. Also we perform $2k-1$ subtractions of integers to compute $A_k(P^* + P^{**}, Q^* + Q^{**}) - L$ and $2k-1$ subtractions of integers to subtract polynomial R. We can compute $x^k M + R$ using $3k-1$ additions and finally we can compute the result using $4k-1$ additions. Hence we can take $b(k) = 2k + 2k - 1 + 2k - 1 + 3k - 1 + 4k - 1 = 13k - 4$.

d) Let $k = 2^s$, for $s \in \mathbb{N}_0$ (where $\mathbb{N}_0 = \{0, 1, 2, 3, 4, \ldots\}$). Find an algorithm Product(k, P, Q) for the efficient multiplication of two polynomials P, Q of degree at most $k - 1$. Describe you algorithm using pseudo-code.

Solution:

Algorithm 5 Product(k, P, Q)

if $k = 1$ **then**
  return $P_0 \cdot Q_0$
else
  for $i \in \{0, \ldots, k/2 - 1\}$ **do**
    $P_i^* \leftarrow P_{k/2+i}$
    $P_i^{**} \leftarrow P_i$
    $Q_i^* \leftarrow Q_{k/2+i}$
    $Q_i^{**} \leftarrow Q_i$
  $L \leftarrow$ Product$(k/2, P^*, Q^*)$
  $R \leftarrow$ Product$(k/2, P^{**}, Q^{**})$
  $M \leftarrow$ Product$(k/2, P^* + P^{**}, Q^* + Q^{**}) - L - R$
  return $x^k L + x^{k/2} M + R$

e) Provide the asymptotic amount of operations required by your algorithm Product(k, P, Q) in terms of k (i.e., in $\mathcal{O}$-Notation with respect to the degree k). Prove your answer with induction.

Solution: Let $T(k)$ be the number of elementarty operations that Product(k, P, Q) performs in order to to multiply polynomials P and Q of degree at most $k - 1$. Let's prove by induction that $T(k) \leq 100k^{\log_2 3} - 20k$, where $k = 2^s$ for $s \in \mathbb{N}_0$.

- **Base Case.**
  Let $s = 0$ and $k = 1$. Then $T(k) = 1 \leq 80 = 100k^{\log_2 3} - 20k$.
- **Induction Hypothesis.**
  Assume that the property holds for $s \in \mathbb{N}_0$ and $k = 2^s$. That is, $T(k) \leq 100k^{\log_2 3} - 20k$.
- **Inductive Step.**
  We must show that the property holds for $s + 1$ and $2k = 2^{s+1}$.

  $$T(2k) \leq 3T(k) + 13k \overset{\text{I.H.}}{\leq} 300k^{\log_2 3} - 60k + 13k \leq 300k^{\log_2 3} - 47k \leq 100 \cdot (2k)^{\log_2 3} - 40k.$$

  By the principle of mathematical induction, this is true for any $s \in \mathbb{N}_0$ and $k = 2^s$.

Hence $T(k) \in \mathcal{O}(k^{\log_2 3})$.

f) Prove the correctness of your algorithm Product(k, P, Q) via mathematical induction. In other words: show that Product$(k, P, Q) = P \cdot Q$ for all $k \in \mathbb{N}$ such that $k = 2^s$, where $s \in \mathbb{N}_0$.

Solution:

- **Base Case.**
  Let $s = 0$, $k = 2^s = 1$. Then P, Q have degree at most 0 and Product$(k, P, Q) = P \cdot Q$.
- **Induction Hypothesis.**
  Assume that the property holds for some integer $s \geq 0$ and $k = 2^s$. That is, Product$(k, P, Q) = P \cdot Q$ for polynomials P and Q of degree at most $k - 1$.

- **Inductive Step.**
  We must show that the property holds for $s+1$ and $2k = 2^{s+1}$.

$$\begin{aligned}
\text{Product}(2k, P, Q) &= x^{2k}L + x^{k}M + R \\
&= x^{2k}\text{Product}(k, P^{*}, Q^{*}) + \\
&\quad + x^{k}(\text{Product}(k, P^{*} + P^{**}, Q^{*} + Q^{**}) - \text{Product}(k, P^{*}, Q^{*}) - \\
&\quad - \text{Product}(k, P^{**}, Q^{**})) + \text{Product}(k, P^{**}, Q^{**}) \\
&\overset{\text{I.H.}}{=} x^{2k}P^{*} \cdot Q^{*} + x^{k}(P^{*} + Q^{*}) \cdot (P^{**} + Q^{**}) - x^{k}P^{*} \cdot Q^{*} - x^{k}P^{**} \cdot Q^{**} + P^{**} \cdot Q^{**} \\
&= x^{2k}P^{*} \cdot Q^{*} + x^{k}P^{*}Q^{**} + x^{k}P^{**}Q^{*} + P^{**} \cdot Q^{**} \\
&= (x^{k}P^{*} + P^{**}) \cdot (x^{k}Q^{*} + Q^{**}) \\
&= P \cdot Q
\end{aligned}$$

  By the principle of mathematical induction, this is true for any integer $s \geq 0$ and $k = 2^{s}$.