Midterm Practice Sheet (Solutions)

Introduction to Computer Science

Professor	Teaching Assistants
Jürgen Schönwälder	Jonas Bayer
	Marco David
	Dung Tri Huynh
	Irsida Mana
	Abhik Pal

October 28, 2018

These problems have been created by the TAs for your midterm preparation. Accordingly, they neither represent the extent nor necessarily the difficulty of the midterm exam. We tried to cover all topics that are relevant, however, you should also study the course notes and revise the homework assignments in order to prepare for the exam. In particular, you should also study the shortest path algorithm and the Boyer Moore algorithm, which are not part of this practice sheet. We will upload the solutions to these problems on Moodle two days before the exam. In case you have any questions on this problem sheet, please contact us TAs.

In this document you can find sample solutions for all the practice problems. Sometimes, multiple possible solutions are given to illustrate different ways to approach a problem. However, in the exam only one solution is expected.

Problem 1 Mathematical Notation Rewrite the following statements using quantifiers:

(a) Every integer greater than 1 can be written as the product of prime numbers

(b) The function $f : \mathbb{R} \setminus \{0\} \to \mathbb{R}$ defined by

$$f(x) = \frac{1}{x}$$

is surjective.

(c) Every polynomial of degree 3 has at least one real root

Which of the statements are true?

Solution The statements can be written as follows:

- $\forall n \in \mathbb{Z}, n > 1\colon$
  $\exists p_1, p_2, \ldots, p_k$ prime: $n = p_1 \cdot p_2 \cdot \ldots \cdot p_k$
- $\forall y \in \mathbb{R}\colon \; \exists y \in \mathbb{R} \setminus \{0\}\colon f(x) = y$
- $\forall p(x) = a_3x^3 + a_2x^2 + a_1x + a_0, a_3 \neq 0 \quad \exists y\colon p(y) = 0$
  The part $a_3 \neq 0$ is necessary to ensure that the degree is actually 3.

The first and the third statement are true. The second statement is false, as the function never assumes the value 0.

Problem 2 More notation Demonstrate your understanding of the Σ-notation by writing out the sums:

$$A = \sum_{k=0}^{4} k^2 \text{ and } B = \sum_{0 \leq k \leq 2} b_{k^2}$$

Solution One can write the sums as

$$\begin{aligned} A &= 0^2 + 1^2 + 2^2 + 3^2 + 4^2 = 30, \\ B &= b_0 + b_1 + b_4. \end{aligned}$$

Problem 3 Proof by contrapositive Let P be a polynomial. Show by contrapositive that if

$$\frac{P(x)}{(x-a)}$$

is not a polynomial, then a cannot be a root of P.

Solution The contrapositive statement is that if a is a root of P, i.e. $P(a) = 0$, then $P(x)/(x-a)$ is a polynomial. A polynomial P can always be written as

$$P(x) = (x - x_1) \cdot (x - x_2) \cdots (x - x_n)$$

where the x_i are the roots of P. If a is a root of P, then one of the x_i is equal to a. Hence,

$$\begin{aligned} \frac{P(x)}{(x-a)} &= \frac{(x - x_1) \cdot (x - x_2) \cdots (x - a) \cdots (x - x_n)}{(x-a)} \\ &= \underbrace{(x - x_1) \cdot (x - x_2) \cdots (x - x_n)}_{x_i \neq a} \end{aligned}$$

which is a polynomial. Thus, the contrapositive follows which is equivalent to the claim.

Problem 4 Induction Prove that n^2+n is an even number for $n \in \mathbb{N}$ using induction. Can you also find a direct proof?

Solution 1 We can write $n^2 + n = n(n+1)$. Clearly, one of the numbers n or $n+1$ must always be even. Hence, their product $n(n+1)$ is also even which shows the claim.

Solution 2 Alternatively, we can prove the statement using induction.

Base case If $n = 0$, then $n^2 + n = 0$ which is even.

Inductive hypothesis We assume that for some n, $n^2 + n$ is even.

Induction step Assume the Induction hypothesis (IH), i.e. that $n^2 + n$ is even. Now we need to show that $(n+1)^2 + (n+1)$ is even, too. We have that

$$\begin{aligned}(n+1)^2 + (n+1) &= n^2 + 2n + 1 + n + 1 \\ &= (n^2 + n) + 2n + 2 = \underbrace{(n^2 + n)}_{\text{Even by IH}} + 2(n+1)\end{aligned}$$

which is even.

Hence, the claim follows. $\square$

Problem 5 Cardinality of the power set Let $A = \{a_1, \cdots, a_n\}$ be a finite set. Show that the cardinality of its power set is given by $|\mathcal{P}(A)| = 2^n$. Try to give a direct proof. If you cannot find one, you can also use induction.

Solution 1 The elements of the power set of A are all possible subsets of A. Any subset of A is characterized by which of the elements $a_1, \cdots, a_n$ are contained in it. Every element can either be in or not in the subset, i.e. there are 2^n ways to form subsets, so $|\mathcal{P}(A)| = 2^n$.

Solution 2 We can also proceed by induction.

Base case Suppose that $n = |A| = 0$. Then $\mathcal{P}(A) = \{\emptyset\}$ and $|\mathcal{P}(A)| = 1 = 2^0$.

Inductive Hypothesis We assume that the hypothesis holds for a set with n elements, i.e. for $A = \{a_1, \cdots a_n\}$ we have $|\mathcal{P}(A)| = 2^{|A|} = 2^n$.

Induction step We need to show that if we add another element a_{n+1} to A, its power set has cardinality 2^{n+1}. Define $A' = A \cup \{a_{n+1}\}$. The power set of $\mathcal{P}(A')$ has twice as many elements as the power set of $\mathcal{P}(A)$, as every subset of A' can either contain the new element a_{n+1} or not contain it. Hence,

$$|\mathcal{P}(A')| = 2 \cdot |\mathcal{P}(A)| = 2 \cdot 2^n = 2^{n+1} \ .$$

Problem 6 Symmetric relations Show that the relation

$$R_k = \{(a, b) \in \mathbb{Z} \colon k|(a - b)\}$$

where $k \in \mathbb{N}$ is positive is an equivalence relation. Why is the relation not total for $k = 2$? For what k is R_k a total relation?

Solution To show that R_k is an equivalence relation, we need to prove that it is reflexive, symmetric and transitive.

Reflexivity We have that $(a, a) \in R_k$ as $(a - a) = 0$ and $k|0$ for any $k \in \mathbb{N}$.

Symmetry Suppose that $(a, b) \in R_k$. We need to show that then $(b, a) \in R_k$, too. As $(a, b) \in R_k$, we have $k|(a - b)$. Hence, also $k| - (a - b)$ which means that $k|(b - a) \implies (b, a) \in R_k$.

Transitivity Suppose that $(a, b) \in R_k$ and that $(b, c) \in R_k$. Then we have $k|(a - b)$ and $k|(b - c)$. Moreover, $k|[(a - b) + (b - c)] \implies k|(a - c)$. Hence, $(a, c) \in R_k$ which completes the claim.

As all properties apply, the relation is an equivalence relation.

To show that R_2 is not total, consider e.g. $a = 0$ and $b = 1$. Then neither $(a, b) \in R_k$ nor $(b, a) \in R_k$.

If $k = 1$, then $R_k = \{(a, b) \in \mathbb{Z} \colon 1|(a-b)\} = \{(a, b) \in Z\}$, as the difference of any two integers divides 1. Accordingly, the relation is clearly total as $(a, b) \in R_k$ for any a, b.

Problem 7 Irreflexive relations Let R be a relation. Using proof by contradiction, show that if R is irreflexive, it is not total.

Solution Suppose R is total. Then for any a, b it holds that always $(a, b) \in R$ or $(b, a) \in R$. In particular, for any a it must hold that $(a, a) \in R$. But this contradicts that R is irreflexive. Hence, if R is irreflexive it is not total.

Problem 8 Domain, codomain, and restrictions Recall that the properties injectivity and surjectivity always depend on the domain and codomain of a function. For example, the function $f : \mathbb{R} \to \mathbb{R}$ defined by

$$f(x) = x^2$$

is neither surjective nor injective. However, if one defines $f : \mathbb{R} \to \mathbb{R}^+$ where $\mathbb{R}^+$ is the set of non-negative real numbers f becomes surjective. On what restriction of its domain is f injective?

Solution If one restricts the domain $\mathbb{R}$ to $\mathbb{R}^+$, i.e. $f : \mathbb{R}^+ \to \mathbb{R}$, the function becomes injective. One can also restrict the domain to $\mathbb{R}^-$.

Problem 9 Make any function surjective Let $f : A \to B$ be a function. Show that f can be made surjective by restricting its codomain. That is, show that there exists a set $B' \subset B$ such that $f : A \to B'$ is surjective.

Solution If $f : A \to B$ is already surjective, then $B' = B$ and the claim holds. Now suppose that f is not surjective, i.e. there are $b \in B$, such that there is no $a \in A$ with $f(a) = b$. Then define $C := \{b \in B : \nexists a \in A : f(a) = b\}$. Then the set $B' := B \setminus C$ contains exactly the elements that are all mapped to. That is, the function $f : A \to B'$ is surjective.

Problem 10 Injectivity and surjectivity example Define two functions $f : \mathbb{R} \to \mathbb{R}^+$ and $g : \mathbb{R}^+ \to \mathbb{R}^+$ by:

$$f(x) = x^2 \quad \text{and} \quad g(x) = \sqrt{x}$$

Calculate the function composition $h = g \circ f$. Is it surjective? Is it injective? If it is not injective: On what restriction of its domain is h injective?

Solution Solution: The domain of h is the domain of f and its codomain is the codomain of g, i.e. $h : \mathbb{R} \to \mathbb{R}^+$. We can calculate

$$h(x) = g(f(x)) = \sqrt{x^2} = |x| \ .$$

Note that h does not map x to x but to its absolute value $|x|$! The absolute value function maps to all numbers in $\mathbb{R}^+$, so h is surjective. However, it is not injective, as e.g. $h(1) = |1| = 1 = |-1| = h(-1)$. If one restricts h to non-negative numbers only, it becomes injective:

$$h|_{\mathbb{R}^+} : \mathbb{R}^+ \to R^+ \ .$$

Alternatively one can choose the non-positive numbers.

Problem 11 Number conversions Convert the following numbers into binary numbers.

(a) 59_{10}

(b) beef_{16}

(c) 717_8

Convert the following numbers into decimal.

(d) 101010_2

(e) 101010_3

(f) 224fe_{16}

Problem 12 Two's complement Working with fixed size integer representations, use a number system with b-complement notation, base $b = 9$ and $n = 4$ digits.

(a) What is the smallest number that can be represented in this number system? State the number in decimal, and also express it in this number system.

(b) What is the largest number that can be represented in this number system? State the number in decimal, and also express it in this number system.

(c) Express the numbers 16_{10} and 121_{10} in this number system.

(d) Express the number -16_{10} and -121_{10} in this number system. Verify your representations by calculating $-(-16_{10})$ and $-(-121_{10})$.

Now consider a number system with $(b-1)$-complement notation. Review questions a) and b). How do the answers change, and why? How many different values can be represented in the two number systems, respectively? You do not need to express your answers in the $(b-1)$-complement system.

Problem 13 Units and Prefixes What is the difference between a *Gigabyte* and a *Gibibyte*? How is each prefix defined?

Problem 14 The International Unit System (SI) Name three examples of SI base units. What fundamental physical concepts are used to define them? Also name two or three derived, or compound units. (You do not need to give precise definitions of neither base or derived units.)

Problem 15 The Unicode Transformation Format (UTF) The UTF-32 encoding assigns a unique 32-bit code to every character, allowing for the possibility to encode 2^{32} different characters. From this point, what was the main motivation to introduce the UTF-8 encoding? How does this encoding, which is commonly used today, fundamentally differ from UTF-32?

Problem 16 ISO-8601 Write today's date in ISO-8601 format. What are the advantages of this format? Compare also to (U.S.-)American conventions.

Problem 17 Boolean equivalence Using equivalence laws for boolean formulas show that the following two formulas are equivalent:

$$\begin{aligned} \varphi: &\quad ((A \lor (B \lor C)) \land (C \lor \neg A)) \\ \psi: &\quad ((B \land \neg A) \lor C) \end{aligned}$$

Solution

$$\begin{aligned}
\varphi &\equiv ((A \lor (B \lor C)) \land (C \lor \neg A)) && \\
&\equiv (((A \lor B) \lor) \land (C \lor \neg A)) && \text{(Associativitiy)} \\
&\equiv ((C \lor (A \lor)) \land (C \lor \neg A)) && \text{(Commutativity} \\
&\equiv (C \lor ((A \lor B) \land \neg A)) && \text{(Distributivity)} \\
&\equiv (C \lor (\neg A \land (A \lor B))) && \text{(Commutativity)} \\
&\equiv (C \lor (\neg A \land A) \lor (\neg A \land B)) && \text{(Distributivity)} \\
&\equiv (C \lor (\neg A \land B)) && \text{(Unsatisfiability law)} \\
&\equiv (C \lor (B \land \neg A)) && \text{(Commutativity)} \\
&\equiv ((B \land \neg A) \lor C) && \text{(Commutativity)} \\
&\equiv \psi.
\end{aligned}$$

Problem 18 Normal Forms Given the following formula

$$((\neg A \implies B) \lor ((A \land \neg C) \iff B))$$

construct an equivalent formula in CNF and one in DNF.

Solution First we conveniently split the formula into smaller sub-formulas:

$$\varphi := (\underbrace{(\neg A \implies B)}_{=:\varphi_0} \lor \underbrace{((A \land \neg C) \iff B)}_{=:\varphi_1}).$$

We now construct a truth table[1] for φ:

$$
\begin{array}{cccccc}
A & B & C & \varphi_0 & \varphi_1 & \varphi
\end{array}
$$
$$
\begin{pmatrix}
1 & 1 & 1 & 1 & - & 1 \\
1 & 1 & - & 1 & 1 & 1 \\
1 & - & 1 & 1 & 1 & 1 \\
1 & - & - & 1 & - & 1 \\
- & 1 & 1 & 1 & - & 1 \\
- & 1 & - & 1 & - & 1 \\
- & - & 1 & - & 1 & 1 \\
- & - & - & - & 1 & 1
\end{pmatrix}
$$

The CND and DNF representations can be read from the table (we use "+" to denote conjunction,"·" disjunction, and a bar negation):

$$
\begin{aligned}
CNF(\varphi) &= (A \cdot B \cdot C) + (A \cdot B \cdot \bar{C}) + (A \cdot \bar{B} \cdot C) + (A \cdot \bar{B} \cdot \bar{C}) \\
&\quad + (\bar{A} \cdot B \cdot C) + (\bar{A} \cdot B \cdot \bar{C}) + (\bar{A} \cdot \bar{B} \cdot C) + (\bar{A} \cdot \bar{B} \cdot \bar{C}), \\
DNF(\varphi) &= (A + B + C) \cdot (A + B + \bar{C}) \cdot (A + \bar{B} + C) \cdot (A + \bar{B} + \bar{C}) \\
&\quad \cdot (\bar{A} + B + C) \cdot (\bar{A} + B + \bar{C}) \cdot (\bar{A} + \bar{B} + C) \cdot (\bar{A} + \bar{B} + \bar{C}).
\end{aligned}
$$

Problem 19 Pascal's triangle in Haskell The binomial coefficient can be defined recursively by

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

with base cases

$$\binom{n}{n} = \binom{n}{0} = 1 \ .$$

(a) Implement a recursive function `binom` in Haskell for the binomial coefficient. What is its signature?

```
> binom 1 0
1
> binom 6 3
20
```

Solution The following code is an implementation of the binomial function with signature `binom :: Int -> Int -> Int`.

```
binom :: Int -> Int -> Int
binom n 0 = 1
binom n k | n == k    = 1
          | otherwise = binom (n-1) (k-1) + binom (n-1) k
```

[1] "1" represents a true value and "−" a false value

(b) Using your `binom` function, implement a function `row :: Int -> [Int]` such that `row i` returns the i'th row of Pascal's triangle. Remember that the element in the n'th row and the k'th column of Pascal's triangle is $\binom{n}{k}$. For example,

```
> row 1
[1, 1]
> map row [0..2]
[[1],[1,1],[1,2,1]]
```

Solution The following code is an implementation of the row function:

```
row :: Int -> [Int]
row n = map (binom n) [0..n]
```

(c) How does Haskell evaluate the expression

```
filter odd $ map length $ map row [0..10]
```

First come up with a solution on paper and then test whether it is correct using the function that you implemented.

Solution The expression gets evaluated to `[3,5,7,9,11]`. The part `map row [0..10]` creates a list that contains row 1 to 10 of Pascal's triangle. The expression `map length` is then applied to this list and returns the length of all these rows. As row 0 of Pascal's triangle has length 1, row 1 has length 2 etc. this is evaluated to `[1, 2, 3,..., 11]`. Finally, `filter odd` returns the odd elements of that list, so the result `[3, 5, 7, 9, 11]` is obtained.

Problem 20 Deep Recursion While compiling source code, a compiler needs to handle many levels of recursion. Hence, while bench-marking a compiler, one could use functions that are *deeply recursive*. Consider the following deep recursive function implemented in Haskell:

```
ack :: Int -> Int -> Int
ack m n | m == 0           = n + 1
        | m > 0 && n == 0 = (ack (m-1) 1)
        | m > 0 && n > 0  = (ack (m-1) (ack m (n-1)))
        | otherwise        = 0
```

(a) Compute the value of `ack m n` for the given values of m and n. Also mention the total number of function calls needed to compute each value (this includes the initial call to the function):

- $m = 1, n = 1$ — 3 (4 calls)

- $m = 1, n = 2$ — 4 (6 calls)
- $m = 2, n = 2$ — 7 (27 calls)

(b) For the special case $m = 1$ write a non-recursive (you are **not** allowed to internally call `ack`) Haskell function `ack' :: Int -> Int` such that `ack' n == ack 1 n`.

Solution The required function is $A'(n) = n + 2$. In Haskell:

```
ack' :: Int -> Int
ack' n | n >= 0    = n + 2
       | otherwise = 0
```

(c) (Bonus) Do the previous exercise for $m = 2$

Solution The required function is $A'(n) = 2n + 3$. In Haskell:

```
ack' :: Int -> Int
ack' n | n >= 0    = n + n + 3
       | otherwise = 0
```

Problem 21 Colored ducks (Bonus) Induction can be quite powerful. For instance, given a set of n ducks, we can show using induction that all of them have the same color.

Base case For $n = 1$, we have one duck then it's the same color as itself.

Inductive hypothesis Numbering ducks from $1, \dots, n$, assume that all of them have the same color.

Inductive step Say we add another duck. From the inductive hypothesis, we know that the first n ducks (numbered $1, 2, \dots, n$) are of the same color. Further, the last n ducks (numbered $2, 3, \dots, (n+1)$) are also of the same color. Now, both of these sets of ducks have the ducks numbered $2, \dots, n$ in common, so the color of duck 1 is the same as any of the middle ducks, which in turn have the same color as the duck $n + 1$. Thus, the first duck and the newly added duck have the same color.

We can conclude, hence, that all ducks are of the same color! QED! Comment on what, if anything, is wrong with this argument?

Solution For $n = 2$ there are no ducks in the middle and the argument doesn't work anymore.