General rules for solving exercises

- This is a theory course, which means: if an exercise does not explicitly say “you do not need to prove your answer” or “justify intuitively”, then a formal proof is always required.

- All exercises and their solutions, no matter whether they are graded or regular/optional ones, are part of the material relevant for the two exams.

- Some of the exercises are marked as "in-class", which means that we do not expect you to solve them before the exercise session. Instead, your teaching assistant will solve them with you in class.

- You are highly encouraged to solve all other exercises (those not marked as "in-class") on your own and to hand in a writeup of your solutions no later than the due date. If you choose to do so, please write the name of your teaching assistant on the front sheet.

Solution 1: A Deterministic MAX-3-SAT Approximation

(a) Consider a random assignment \( \alpha \) where every variable is set independently and with probability \( \frac{1}{2} \) to either 0 or 1. The probability that a 3-clause is not satisfied by \( \alpha \) is \( \frac{1}{8} \), which is the case when all its literals evaluate to zero. Therefore, the probability that a clause is satisfied is \( 1 - \frac{1}{8} = \frac{7}{8} \). Let \( X_i \) be the indicator variable for the event that clause \( i \) is satisfied by the random assignment. We have seen that \( E[X_i] = \frac{7}{8} \) for all \( i = 1, \ldots, m \). By linearity of expectation we get

\[
E[X] = E \left[ \sum_{i=1}^{m} X_i \right] = \sum_{i=1}^{m} E[X_i] = \frac{7}{8} m.
\]

This expectation can be seen as averaging over all possible assignments. The fact that the average is \( \frac{7}{8} m \) implies that not all of the summands can be \( < \frac{7}{8} \). Therefore we know that there exists an assignment that satisfy at least \( \frac{7}{8} m \) clauses. \( \Box \)

(b) We have \( 2^2 = 4 \) possible assignments for the two variables of the 2-clause. One of these assignments will not satisfy the 2-clause but the others will. Thus the probability for the 2-clause to be satisfied is \( \frac{3}{4} \). For a 1-clause it is even simpler. To satisfy a 1-clause we have to set the variables of the clause in the right way, the other variables are not important for this clause. Therefore the probability to randomly satisfy an 1-clause is exactly \( \frac{1}{2} \). Again by linearity of expectation we get

\[
E[X] = E \left[ \sum_{i=1}^{m} X_i \right] = \sum_{i=1}^{m} E[X_i] = \frac{1}{2} m_1 + \frac{3}{4} m_2 + \frac{7}{8} m_3.
\]

\( \Box \)
(c) (i) Let us say that we have a partial assignment $\alpha^*$ to the variables (partial means here that some of the variables are set and some of the variables are not set). Partially evaluating the formula under the variables set, we can simplify clauses with set variables by using
\[ x \lor 0 = x, \quad x \lor 1 = 1. \]
We may obtain satisfied clauses (i.e. they evaluate to 1) by the partial assignment, while other clauses may shrink to 2-clauses, 1-clauses or even become unsatisfiable (i.e. all three variables of such a clause are set to the wrong value). We denote the result of this partial evaluation by $F$. Note that $F$ can be efficiently obtained (i.e., in polynomial time), and we also can efficiently count the number of satisfied clauses, as well as the remaining 1-, 2- and 3-clauses. Denote by $m_0(F)$ the number of satisfied clauses in $F$, and by $m_j(F)$, $j \in \{1, 2, 3\}$ the number of $j$-clauses in $F$. Then, using (b), we can efficiently compute the conditional expectation
\[ E[X | \alpha_1 = \alpha_1^*, \ldots, \alpha_{i-1} = \alpha_{i-1}^*, \alpha_i = k] = m_0(F) + \frac{1}{2} m_1(F) + \frac{3}{4} m_2(F) + \frac{7}{8} m_3(F). \]

(ii) We claim that for all $i$, the expected number of satisfied clauses on condition of the partial assignment $\alpha^*$ is at least $\frac{7}{8} m$. We prove this claim by induction. The base case, $i = 0$, holds by (a). As for the induction step, we assume that we have
\[ E[X | \alpha_1 = \alpha_1^*, \ldots, \alpha_{i-1} = \alpha_{i-1}^*, \alpha_i = 0] \geq \frac{7}{8} m \]
and we want to show that this is still true for $(i - 1) \rightarrow i$. By the law of total expectation we have
\[ \frac{1}{2} E[X | \alpha_1 = \alpha_1^*, \ldots, \alpha_{i-1} = \alpha_{i-1}^*, \alpha_i = 0] + \frac{1}{2} E[X | \alpha_1 = \alpha_1^*, \ldots, \alpha_{i-1} = \alpha_{i-1}^*, \alpha_i = 1] \]
\[ = E[X | \alpha_1 = \alpha_1^*, \ldots, \alpha_{i-1} = \alpha_{i-1}^*] \geq \frac{7}{8} m. \]
Therefore one of the two left-hand side summands is at least $\frac{7}{8} m$. As in the algorithm, we choose the larger summand, we select the assignment such that the expected number of satisfied clauses will be at least $\frac{7}{8} m$. This shows the induction step. Therefore in the end, we get a total assignment (every variable is set) which satisfies at least $\frac{7}{8} m$ clauses. The maximum of clauses which can be satisfied is clearly at most $m$. Therefore the algorithm at hand is a deterministic polynomial-time $\frac{7}{8}$-approximation algorithm for max-SAT applied to proper 3-CNF formulas.

Solution 2: Linear Functions
Assume for contradiction that there are two different vectors $a, b$ which satisfy the inequality. We know that in more than $\frac{3}{4}$ of all vectors $x \in \{0, 1\}^n$, the function $T(x)$ agrees with the linear function $a^T x$, and analogously for more than $\frac{3}{4}$ of all vectors $x$, it agrees with $b^T x$. Let us visualize this by shading the vectors $x$ for which $T(x)$ agrees with one of the linear functions, e.g.
After sorting the vectors as in the lower picture, we see that there is a shaded block in the middle which is more than half of the whole length where we have \( a^T x = T(x) = b^T x \). Indeed, the two distinct linear functions \( a, b \) must be equal in more than half of the values - written down formally:

\[
1 \geq \Pr [a^T x = T(x) \text{ or } b^T x = T(x)] \\
= \Pr [a^T x = T(x)] + \Pr [b^T x = T(x)] - \Pr [a^T x = T(x) = b^T x] \\
> \frac{3}{4} + \frac{3}{4} - \Pr [a^T x = T(x) = b^T x].
\]

By rearranging the terms it follows that

\[
\Pr [a^T x = b^T x] \geq \Pr [a^T x = T(x) = b^T x] > \frac{1}{2}.
\]

But this is a contradiction to the fact that two distinct linear functions agree in exactly \( \frac{1}{2} \) of all \( x \). This can be seen as follows: Choose \( j \in \{1, \ldots, n\} \) such that \( a_j \neq b_j \). We then have

\[
a^T x - b^T x = (a - b)^T x = \sum_{i \in [n]} (a_i - b_i)x_i + (a_j - b_j)x_j
\]

\[
= S + (a_j - b_j)x_j
\]

For \( x \in \text{u.a.r.} \{0, 1\}^n \), consider \( \Pr [a^T x = b^T x] = \Pr [S + (a_j - b_j)x_j = 0] \). Under this probability distribution, we have

\[
\Pr [(a_j - b_j)x_j = 0] = \Pr [(a_j - b_j)x_j = 1] = \frac{1}{2}.
\]

Thus, we also have

\[
\Pr [a^T x = b^T x] = \Pr [S + (a_j - b_j)x_j = 0] = \Pr [S + (a_j - b_j)x_j = 1] = \frac{1}{2}
\]

(cf. Exercise 4.1 on error detection in matrix multiplication).

This confirms the above-stated contradiction, and we can conclude that there is at most one \( a \) with properties as listed in the exercise description. \( \square \)
Solution 3: PCP inapproximability version: Theorem 7.5 $\implies$ Theorem 7.1’

Let $\varepsilon > 0$, and assume that there is a polynomial time algorithm $A$ which approximates 3-SAT with ratio $7/8 + \varepsilon$. Let $L \in \text{NP}$ be any language. We want to show that $L \in \text{P}$. 

For this, we use the reduction given by Theorem 7.5, where we will use $\varepsilon' := \frac{\varepsilon}{2}$. This will give us the polynomial time computable function $f$ mapping the instances of our language $L$ to satisfiable 3-SAT formulas, and the instances $x \notin L$ to 3-SAT formulas where each assignment satisfies at most fraction $\frac{7}{8} + \varepsilon'$ of the clauses.

After we have run this $f$ for our input $x$, we now run the approximation algorithm $A$ on the resulting formula $f(x)$. If the assignment returned by $A$ satisfies at least a $7/8 + \varepsilon$ fraction of the clauses, then we output “$x \in L$”, otherwise we output “$x \notin L$”. The whole procedure is clearly a polynomial time algorithm.

If $x \in L$, then this algorithm will always correctly output “$x \in L$”, as the resulting formula $f(x)$ is satisfiable and $A$ is a $7/8 + \varepsilon$ approximation algorithm. If $x \notin L$, then this algorithm will output “$x \notin L$”, because the 3-SAT formula $f(x)$ has the property that no assignment satisfies more than $\frac{7}{8} + \varepsilon' < \frac{7}{8} + \varepsilon$ of the clauses. \hfill \Box

Solution 4: QP-SAT is NP-complete

Observe that QP-SAT is in NP: It can be checked deterministically and in polynomial time whether $x = (x_1, \ldots, x_n)$ is a common root. By showing that 3-SAT is polynomial time reducible to QP-SAT, we show that QP-SAT is NP-hard, i.e. at least as difficult/hard as 3-SAT which is NP-complete. The fact that QP-SAT is in NP and NP-hard makes it an NP-complete problem.

(a) The following polynomial of degree 3 is generated for the clause $C$:

$$C = (x_1 \lor \overline{x}_2 \lor x_3) \implies p_C(x_1, x_2, x_3) = (1 - x_1)x_2(1 - x_3).$$

It is easily seen that an assignment to the variables $x_i$ satisfies the clause $C$ if and only if the polynomial $p_C(x_1, x_2, x_3)$ is zero for the assigned values of the $x_i$'s. (Using the variable $x$ in the formula as well as in the polynomial might be not that clean, but it works as $x \in \{0, 1\}$ in both cases.)

(b) Since the polynomial generated in (a) is not quadratic, we introduce a new variable $y$ and replace the polynomial by two quadratic ones in the variables $x_1, x_2, x_3$ and $y$.

$$p_C(x_1, x_2, x_3) = (1 - x_1)x_2(1 - x_3) \implies$$

$$q_C(x_1, x_2, x_3, y) = y - (1 - x_1)x_2$$

$$r_C(x_1, x_2, x_3, y) = y(1 - x_3)$$

Observe that $(r_1, r_2, r_3)$ is a root of $p_C$ if and only if $(r_1, r_2, r_3, (1 - r_1)r_2)$ is a common root of $q_C$ and $r_C$.

(c) Let $n$ be the number of variables in the formula $\psi$. For every clause $C_i = (l_{i1} \lor l_{i2} \lor l_{i3})$ in $\psi$, a polynomial of degree 3 is generated by generalizing the idea of (a). The corresponding polynomial is $p_{C_i}(x_1, \ldots, x_n) = \prod_{j=1}^3 f(l_{ij})$ where

$$f(l_{ij}) = \begin{cases} 
  x_k & \text{if } l_{ij} \text{ is } x_k \\
  (1 - x_k) & \text{if } l_{ij} \text{ is } \overline{x}_k,
\end{cases}$$

and a satisfying assignment of $C_i$ corresponds to a root of $p_{C_i}$ and vice versa.
The polynomial $p_{C_i}(x_1, \ldots, x_n)$ is then replaced by $q_{C_i}(x_1, \ldots, x_n, y_i) = y_i - f(l_{i1}) \cdot f(l_{i2})$ and $r_{C_i}(x_1, \ldots, x_n, y_i) = y_i \cdot f(l_{i3})$, two polynomials of degree 2 whose common roots (without the $y_i$-value) are exactly the roots of $p_{C_i}$.

Doing this for every clause, we get $2m$ quadratic polynomials. If the formula $\psi$ is satisfiable and $\alpha$ is a satisfying assignment for it, then $\alpha$ corresponds to a common root of all $p_{C_i}$ and thus to a common root of all $2m$ polynomials $q_{C_i}, r_{C_i}$. On the other hand, given that there is a common root for the constructed $2m$ polynomials, there is also a common root for the polynomials $p_{C_i}$. This common root corresponds to a satisfying assignment for $\psi$.

Thus, the constructed $2m$ polynomials have a common root if and only if $\psi$ is satisfiable. We can therefore decide 3-SAT by deciding QP-SAT. By noting that the polynomials can be constructed in polynomial time, we get the desired NP-hard result.