1. (3 Punkte) Ein Rechner schafft 10^9 Schritte pro Sekunde.
   (a) Wie lange brauchen Algorithmen mit n^2, n^3 und 2^n Schritten für $n = 20$ und für $n = 40$?
   (b) Um welchen Faktor wächst die Laufzeit jeweils, wenn sich n verdoppelt?

2. (3 Punkte) Formuliere die folgenden Optimierungsprobleme als Entscheidungsprobleme.
   (a) Finde die kürzeste Rundreise durch 20 Städte.
   (b) Wie viele Farben braucht man mindestens, um eine Landkarte zu färben?
   (c) Packe Gegenstände mit möglichst großem Wert in einen Rucksack mit begrenzter Tragfähigkeit.

   Wie kann man mit einem schnellen Algorithmus für das Entscheidungsproblem in (a) die Länge der kürzesten Rundreise bestimmen?

3. (4 Punkte) Gib jeweils an, in welcher der Klassen P, NP, R das Problem liegt, soweit man es weiß, und begründe.
   (a) Gibt es in einem Graphen einen Weg von u nach v?
   (b) Hat dieses Sudoku eine Lösung? (verallgemeinert auf beliebig große Felder)
   (c) Hat die Zahl n einen Teiler zwischen 2 und k?
   (d) Hält dieses Programm mit dieser Eingabe an?

4. (4 Punkte) Gegeben ist die Formel

$$(\overline{x_1} \vee x_2 \vee x_3) \wedge (x_2 \vee \overline{x_3} \vee x_4) \wedge (x_1 \vee \overline{x_2})$$

   (a) Benenne die Klauseln und Literale.
   (b) Werte die Formel für $x_1 = True$, $x_2 = False$, $x_3 = True$, $x_4 = False$ aus und für $x_1 = True$, $x_2 = True$, $x_3 = False$, $x_4 = True$.
   (c) Wie viele Belegungen muss ein Brute-Force-Algorithmus höchstens prüfen? Wie viele sind es bei 30 Variablen?
   (d) Warum ist SAT in NP?

5. (3 Punkte) Zeige, dass die Formel

$$(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2})$$

nicht erfüllbar ist. Warum ist es für ein Entscheidungsverfahren schwieriger, die Antwort Nein zu begründen als die Antwort Ja?

6. (4 Punkte) Der Graph hat die Knoten A, B, C und die Kanten A–B und B–C. Die Frage, ob er sich mit 2 Farben färben lässt, soll in eine SAT-Formel übersetzt werden. Die Variable $x_{v,c}$ bedeutet: Knoten v hat Farbe c ($c = 1, 2$).
   (a) Schreibe die Klauseln auf: Jeder Knoten hat mindestens eine Farbe, kein Knoten hat zwei Farben, benachbarte Knoten haben nicht dieselbe Farbe. Wie viele Variablen und Klauseln sind es?
   (b) Gib eine erfüllende Belegung an und die zugehörige Färbung.
   (c) Warum muss die Übersetzung selbst schnell (in polynomieller Zeit) gehen?

7. (3 Punkte) Nach dem kleinen Satz von Fermat gilt für jede Primzahl p und $1 \leq a < p$: $a^{p-1} \,\%\, p = 1$.
   (a) Berechne $2^8 \,\%\, 9$ und $2^{10} \,\%\, 11$. Was folgt daraus für 9 und 11?
   (b) Es gilt $2^{560} \,\%\, 561 = 1$, aber $561 = 3 \cdot 11 \cdot 17$. Was bedeutet das für einen Primzahltest, der nur $2^{n-1} \,\%\, n$ berechnet?
   (c) Warum ist der Fermat-Test ein Beispiel für „Suchen ohne zu suchen“?

8. (4 Punkte) (a) Was bedeutet es, dass ein Problem NP-vollständig ist?
   (b) Was hat Stephen Cook 1971 bewiesen und warum ist das wichtig?
   (c) Was würde folgen, wenn jemand einen polynomiellen Algorithmus für SAT findet?
   (d) Warum benutzt man für NP-vollständige Probleme in der Praxis oft Näherungsverfahren?

9. (4 Punkte) Der Graph hat die Knoten a1, a2, a3, b1, b2, b3. Jedes a_i ist mit jedem b_j verbunden, außer wenn $i = j$ ist. Greedy färbt die Knoten in einer Reihenfolge und gibt jedem Knoten die kleinste Farbnummer (0, 1, 2, ...), die keiner seiner schon gefärbten Nachbarn hat.
   (a) Färbe in der Reihenfolge a1, b1, a2, b2, a3, b3. Wie viele Farben braucht Greedy?
   (b) Färbe in der Reihenfolge a1, a2, a3, b1, b2, b3. Wie viele Farben braucht Greedy jetzt?
   (c) Was folgt daraus für Greedy als Verfahren?

10. (3 Punkte) Ein Graph hat einen Mittelpunkt M mit vier Nachbarn A, B, C, D. Jeder davon hat noch einen weiteren Nachbarn: A–E, B–F, C–G, D–H. Gesucht ist eine möglichst kleine dominierende Menge: Jeder Knoten soll in der Menge liegen oder einen Nachbarn in der Menge haben.
   (a) Wende Greedy an: Wähle immer den Knoten, der die meisten noch nicht abgedeckten Knoten abdeckt (sich selbst und seine Nachbarn). Bei Gleichstand den alphabetisch ersten.
   (b) Gib eine kleinere dominierende Menge an.

11. (4 Punkte) Schreibe eine Funktion `greedy_nach_grad(G)`, die einen Graphen (Dictionary mit Nachbarlisten) mit Greedy färbt und die Knoten dabei nach der Zahl ihrer Nachbarn absteigend sortiert, bei Gleichstand alphabetisch. Sie soll die Färbung zurückgeben. Teste sie mit dem Graphen aus der Aufgabe zur Greedy-Färbung.