

## Übungen zu Theoretische Informatik III

Abgabe bis zum 31.05.05

### Aufgabe 4.1      Backtracking

8 Punkte

Eine aussagenlogische Formel  $\phi$  ist in 3-KNF, falls sie in konjunktiver Normalform ist und jede Klausel aus höchstens drei Literalen besteht (abweichend vom ersten Übungsblatt).

Wir bezeichnen aussagenlogische Variablen fortlaufend mit  $x_1, x_2, x_3, \dots$

Dann ist  $\phi[x_i := 0]$  die aussagenlogische Formel, welche aus  $\phi$  entsteht, indem jedes Vorkommen von  $x_i$  in  $\phi$  durch 0 ersetzt wird, entsprechend für  $\phi[x_i := 1]$ . Für ein Literal  $L$  bezüglich der Variablen  $x_i$  setzen wir entsprechend

$$\phi[L := t] := \begin{cases} \phi[x_i := t] & , \text{ falls } L \equiv x_i \\ \phi[x_i := 1 - t] & , \text{ falls } L \equiv \neg x_i \end{cases} \quad \forall t \in \{0, 1\}.$$

Wie Sie wissen, ist das Problem *3-KNF-SAT* **NP**-vollständig. Ein einfacher, exponentieller Algorithmus, welcher eine Formel  $\phi$  in 3-KNF auf Erfüllbarkeit testet, sieht wie folgt aus:

```
bool 3_KNF_SAT( Formel  $\phi$  ) {
  if (  $\phi$  enthält keine Variablen mehr ) {
     $r$  = Wahrheitswert von  $\phi$  berechnen;
    return  $r$ ;
  }
  else {
     $i$  = niedrigster Index einer Variable in  $\phi$ ;
    if ( 3_KNF_SAT(  $\phi[x_i := 0]$  ) == 1 )
      return 1;
    else
      return 3_KNF_SAT(  $\phi[x_i := 1]$  ); } }
```

Der Algorithmus betrachtet offensichtlich im schlimmsten Fall alle  $2^n$  relevanten passenden Belegungen, wenn  $n$  die Anzahl der in  $\phi$  vorkommenden Variablen ist, und wertet die Formel  $\phi$  unter jeder der betrachteten Belegungen in Linearzeit aus. Die Laufzeit kann also nach oben durch  $2^n \cdot \mathcal{O}(n)$  abgeschätzt werden.

Ihre Aufgabe ist es, einen Algorithmus zu entwerfen, dessen Laufzeit durch den Term  $(1.84)^n$  dominiert wird.

Sie dürfen hierfür verwenden, dass eine Formel in 2-KNF (d.h., jede Klausel hat höchstens zwei Literale) effizient in Polynomialzeit auf Erfüllbarkeit getestet werden kann. Ihnen steht hierfür eine Prozedur `2_KNF_SAT` zur Verfügung, welche genau dann 1 zurückgibt, wenn die übergebene 2-KNF-Formel erfüllbar ist.

Sei dann  $\phi$  eine beliebige Formel in 3-KNF. Reduzieren Sie rekursiv in  $\phi$  die Anzahl der Klauseln über drei Literalen, bis Sie schließlich bei einer 2-KNF-Formel ankommen. Nutzen Sie für eine Klausel  $K = L_1 \vee L_2 \vee L_3$  aus, dass diese erfüllt sein muss, damit  $\phi$  erfüllt werden kann. Sie sollten maximal drei rekursive Aufrufe für eine solche Klausel  $K$  benötigen.

Stellen Sie dann eine Rekursionsgleichung für die maximale Anzahl  $R(n)$  der Aufrufe von `2_KNF_SAT` auf, welche Ihr Algorithmus für eine 3-KNF-Formel mit  $n$  verschiedenen (unbelegten) Variablen tätigt. Überprüfen Sie, dass  $R(n) \approx 1.84^n$  eine Lösung dieser Rekursionsgleichung ist.

### Aufgabe 4.2

8 Punkte

Unter dem Rucksack-Problem versteht man folgende Aufgabenstellung:

Gegeben sind ein Zielgewicht  $b \in \mathbb{N}$  und  $n$  Objekte  $\{1, \dots, n\}$ , welchen ein Gewicht  $g(i) \in \mathbb{N}$  zugeordnet ist.

Gesucht ist eine Auswahl  $I \subset \{1, \dots, n\}$ , deren Gesamtgewicht

$$G(I) := \sum_{k \in I} g(k)$$

gerade gleich dem Zielgewicht  $b$  ist.

Wie Sie u.U. aus TI2 wissen, ist dieses Problem ebenfalls **NP**-vollständig.

- (a) Geben Sie eine möglichst effiziente Implementierung des trivialen Algorithmus samt Laufzeitabschätzung an, welche einfach alle möglichen Indexmengen  $I \subset \{1, \dots, n\}$  ausprobiert.

- (b) Wandeln Sie Ihren Algorithmus aus Aufgabe (a) so ab, dass er Ihnen die Menge  $\{(I, G(I)) \mid I \subseteq \{1, \dots, n\}\}$  berechnet. Wie viel Zeit benötigt Ihr Algorithmus?
- (c) Ihre weitere Aufgabe ist es nun, einen Algorithmus für das Rucksackproblem zu entwerfen, dessen Laufzeit nur durch den Term  $2^{\frac{n}{2}}$  dominiert ist.

Wenden Sie Ihren Algorithmus aus Teilaufgabe (b) auf die Mengen  $\{1, \dots, \lfloor \frac{n}{2} \rfloor\}$  und  $\{\lfloor \frac{n}{2} \rfloor + 1, \dots, n\}$  an. Überlegen Sie sich nun, wie Sie möglichst effizient aus den erhaltenen Teilergebnissen eine mögliche Lösung zusammensetzen können, so dass die resultierende Gesamtlaufzeit durch  $2^{\frac{n}{2}} \cdot p(n)$  beschränkt ist, wobei  $p$  ein Polynom ist.