Technische Universität München (I7) **Summer term 2016**
Prof. J. Křetínský / Dr. C.H. Broadbent **29. Juni 2016**

Computational Complexity – Homework 11

Discussed on TBA: (Please see Website News section for information as an email needs to be sent to me to arrange next week's tutorial time(s)).

Exercise 11.1

(a) Modify the approximation algorithm for vertex cover shown in the lecture (lecture 18, slide 19) such that it always computes an optimal solution if the given graph is a disjoint union of linear chains.

(b) Consider the following algorithm for approximating an optimal vertex cover:

While G has edges choose any node v of maximal degree of G; add it to C; and remove v and all edges connected to it from G.

- Quantify the approximation this algorithm obtains on the following graph:

  $V := \{a_1, a_2, a_3, a_4\} \cup \{b_1, b_2\} \qquad E := \{\{a_i, a_i\} | 1 \le i \le 4\} \cup \{\{a_i, b_j\} | 1 \le i \le 4, 1 \le j \le 2\}$

- Can you generalize the graph from above to show that the approximation error can be as large as $\approx \ln |V|$?

Exercise 11.2

Show that, if $\textsc{Sat} \in \mathbf{PCP}(r(n), 1)$ for some $r(n) = o(\log n)$, then $\mathbf{P} = \mathbf{NP}$.

Exercise 11.3

Prove that $\textsc{QuadEq}$ is **NP**-complete.

Exercise 11.4

Consider the following problem:

Input: A matrix $A \in \mathbb{Q}^{m \times n}$, a vector $b \in \mathbb{Q}^m$.
Target: Determine the maximal number of equations in $Ax = b$ which can simultaneously be satisfied by some $x \in \mathbb{Q}^n$.

Show that there is a constant $\rho < 1$ such that approximating the maximal size is **NP**-hard.

Exercise 11.5

We consider the optimization variant of the KNAPSACKPROBLEM:

Input: Values $v_1, \ldots, v_n$, weights $w_1, \ldots, w_n$ and a weight bound W, all natural numbers representable by n bits.

Target: Compute the maximal total value attainable by any selection S of total weight at most W, i.e.,

$$v_{\text{opt}} := \max\{\sum_{i \in S} v_i \mid S \subseteq \{1, 2, \ldots, n\} \wedge \sum_{i \in S} w_i \leq W\}.$$

(a) In Exercise 3.2(c) we have discussed a pseudo-polynomial algorithm which solves this problem in time $\mathcal{O}(nW)$. Similarly, design an algorithm which finds the maximal total value by computing an array A with

$$A[j, v] = \min\{W + 1, \sum_{i \in S} w_i \mid S \subseteq \{1, 2, \ldots, j\} \wedge \sum_{i \in S} v_i = v\}.$$

Your algorithm should be polynomial in n and $V := \sum_{i=1}^{n} v_i$.

(a') Modify your algorithms so that it runs in time polynomial in n and v_{opt}.

(b) Assume you replace all values v_i by $v'_i := \lfloor v_i/2^k \rfloor$ for some fixed $k \geq 0$, i.e., you remove the k least significant bits. The weights w_i and the weight limit W stay unchanged. Let v_{opt}, resp. v'_{opt} be the optimal value for the original resp. reduced instance.

We take $v'_{\text{opt}} \cdot 2^k$ as an approximation for v_{opt}.

- Show that $v_{\text{opt}} \geq v'_{\text{opt}} 2^k$. What is the approximation error in the worst case?
- Choose k s.t. the approximation error is at most $\epsilon > 0$. Show that for this k the algorithm runs in time polynomial in n and $1/\epsilon$.