RWTH Aachen University **WS 2010/11**
Lehrgebiet Theoretische Informatik Tutorial 5
Kneis—Langer—Sikdar 16.11.2010

Tutorial Exact Algorithms

Exercise T8 Consider the following algorithm A for Strange Set:

```
if V = ∅ return 1;
if (there is v ∈ V with deg(v) = 1) return A(G \ {v});
pick v of maximum degree;
if (deg(v) ≥ 5) return A(G \ {v})+1;
return min{A(G \ {u ∈ N[v] | deg(u) = deg(v)}) + 2, A(G \ {v}) + 1}
```

Show that the running time of Algorithm A is less than $O^*(2^n)$.

Exercise T9

Consider the following scheduling problem. An input consists of a set of n jobs $J_1, \ldots, J_n$ with execution times $\tau_1, \ldots, \tau_n$ and costs $c_1(t), \ldots, c_n(t)$. Job J_i requires time τ_i for its completion and the cost associated on completing it at time t is $c_i(t)$.

- The jobs are to be executed on a single machine which is in use constantly.
- There is no *preemption*, that is, one cannot schedule a job on the machine if the current job has not been executed fully.
- Processing starts at time $t = 0$.

A *schedule* is the order in which the jobs are executed, that is, it is simply a permutation of $\{1, \ldots, n\}$. The *termination time* of a job $J_{\pi(i)}$ executed in the schedule π is

$$t_{\pi(i)} = \sum_{j=1}^{i} \tau_{\pi(j)}.$$

The *total cost* associated with schedule π is

$$\sum_{i=1}^{n} c_{\pi(i)}(t_{\pi(i)}).$$

Find out a schedule that minimizes the total cost.

Homework Assignment H9 (10 Points+10 Bonus Points)

Remember that Independent Set can be solved with a branching vector of

$$\left(\alpha_{\deg(u)} + \sum_{i=3}^{\Delta(G)} c_i(\alpha_i - \alpha_{i-1}), \alpha_{\deg(u)} + \sum_{i=3}^{\Delta(G)} c_i\alpha_i + \deg(u)\min\{\alpha_3/3, \alpha_i - \alpha_{i-1} \mid 3 \leq i \leq \deg(u)\}\right)$$

for $\deg(u) \geq 4$.

Construct all branching vectors for $\deg(u) = 5$.

Let $\alpha_i = 1$ for $i \geq 5$ and $\alpha_i = 0$ for $i \leq 2$. Find values for α_3, α_4 such that these branching values yield branching number less than $\tau(1, 6)$.

Bonus Points: Find values for α_3, α_4 such that the branching number is at most 1.26 for any degree.

Homework Assignment H10 (10 Points)

Consider the KNAPSACK problem defined as follows: An input to the problem consists of n items $\{1, \ldots, n\}$ such that item i has weight $w_i \in \mathbf{N}$ and profit $p_i \in \mathbf{R}^+$, and an integer W. The goal is to obtain a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} p_i$ is a maximum subject to the condition $\sum_{i \in S} w_i \leq W$. Intuitively, one has a knapsack that can hold items of total weight at most W and one has to pack the knapsack trying to maximize the total profit. Use dynamic programming to obtain an algorithm for KNAPSACK with running time $O(nW)$. Does this running time contradict the fact that the KNAPSACK problem is NP-complete?

Solution For $1 \leq w \leq W$, define $\mathrm{OPT}[w; j]$ to be the maximum profit that can be earned when the knapsack capacity is w and the items are to be picked from $\{1, \ldots, j\}$. We seek the value $\mathrm{OPT}[W; n]$. Then $\mathrm{OPT}[w; 1] = p_1$ if $w_1 \leq 1$; else 0;

$$\mathrm{OPT}[w; j] = \max\{\mathrm{OPT}[w - w_j; j - 1] + p_j, \mathrm{OPT}[w; j - 1]\}.$$

The first case arises only if $w_j \leq w$. It is easy to verify that $\mathrm{OPT}[W; n]$ represents the optimal value and that the computation can be carried out in time $O(nW)$. Note that a bit representation of an instance requires $O(\log n + n \log \max_i\{w_i, p_i\} + \log W)$ bits. Therefore a running time of $O(nW)$ is not polynomial in the *input size* and does not contradict the NP-completeness of the problem.