University of Zurich
Institute of Computer Science
Dr. Alexander Souza

Spring Term 2015
Sheet 01

Advanced Algorithms: Design and Analysis

Exercise 1 (Running Times)

Suppose you are given an algorithm, which receives just a single number n as its input in binary encoding. Assume that the algorithm executes always exactly n^2 many elementary operations on input n. Is this a polynomial time algorithm?

Exercise 2 (Spanning Trees)

Let G be a connected undirected graph on n vertices. Show that any spanning tree T of G has exactly $n-1$ edges.

Exercise 3 (Edges in Minimum Spanning Trees)

Let G be a connected undirected graph on the vertex set V and the edge set E. Furthermore, we are given a cost function $c : E \to \mathbb{R}$ on the edges. Prove the following:

(a) Let e be an edge in G with minimal cost. Then there is a minimum spanning tree T which contains e.

(b) Let e be an edge in G with maximal cost. Then there is a minimum spanning tree T which does not contain e.

Exercise 4 (Maximum Spanning Trees)

Let G be a connected undirected graph on the vertex set V and the edge set E. Furthermore, we are given a cost function $c : E \to \mathbb{R}$ on the edges. A *maximum spanning tree* is a spanning tree in G having largest total edge cost (among all spanning trees).
Modify the algorithms Kruskal and Prim from the lecture such that maximum spanning trees are computed. Prove the correctness of your modification.

Programming 5 (First Steps with CPLEX)

Download ILOG CPLEX from our website. Follow the instructions for installation provided there. Also download and read some of the provided tutorials.

University of Zurich
Institute of Computer Science
Dr. Alexander Souza

Spring Term 2015
Sheet 02

Advanced Algorithms: Design and Analysis

Exercise 1 (Set Cover Greedy)

Give an instance showing that the Greedy algorithm for Set Cover is a H_n-approximation.

Exercise 2 (Vertex Cover)

The problem Vertex Cover is defined as follows: Given a simple undirected Graph $G = (V, E)$ find a set $C \subseteq V$ with minimal cardinality, such that each edge in E has at least one endvertex in C.

(a) Formulate Vertex Cover as a Set Cover problem.

(b) Give a 2-approximation for Vertex Cover (not using (a)).
Hint. Consider the edges of the graph. For some of the edges, include both endvertices.

Exercise 3 (Multi-Source Multi-Sink)

The Maximum Flow problem can be generalized to multiple sources $s_1, \ldots, s_p$ and sinks $t_1, \ldots, t_q$. The value of a flow f is then defined as $\text{value}(f) = \sum_{1 \leq i \leq p} \text{bal}_f(s_i)$. Show that the Multi-Source Multi-Sink Maximum Flow problem can be reduced to the ordinary Maximum Flow problem.

Exercise 4 (Guest Shuffle)

Suppose you are organizing a dinner and lay n tables. You invite m families to join the dinner and family i has a_i members. Furthermore table j has b_j seats. In order to boost the inter-family-communication you want to make sure that no two members of the same family are at the same table (if this is possible). Formulate this seating arrangement problem as a Maximum Flow problem.

Programming 5 (OPL Formulations)

Give formulations of Set Cover and Vertex Cover in OPL. Run your formulations in ILOG CPLEX with a few examples.

University of Zurich
Institute of Computer Science
Dr. Alexander Souza

Spring Term 2015
Sheet 03

Advanced Algorithms: Design and Analysis

Exercise 1 (Greedy Algorithm for Knapsack)

Give an instance which shows that the approximation guarantee of 1/2 for the Greedy algorithm for Knapsack is tight.

Exercise 2 (Fractional Knapsack)

Let $c, w \in \mathbb{R}^n$ be non-negative vectors with $c_1/w_1 \geq c_2/w_2 \geq \cdots \geq c_n/w_n$. The Fractional Knapsack problem is the following mathematical program:

$$\begin{aligned} \text{maximize} \quad & \sum_{j=1}^{n} c_j x_j, \\ \text{subject to} \quad & \sum_{j=1}^{n} w_j x_j \leq W, \\ & 0 \leq x_j \leq 1 \quad j = 1, \ldots, n. \end{aligned}$$

Let $k = \min\left\{ j \in \{1, \ldots, n\} : \sum_{i=1}^{j} w_i > W \right\}$. Show that an optimum solution for the Fractional Knapsack problem is given by the vector x with

$$\begin{aligned} x_j &= 1 && \text{for } j = 1, \ldots, k-1, \\ x_j &= \frac{W - \sum_{i=1}^{k-1} w_i}{w_k} && \text{for } j = k, \text{ and} \\ x_j &= 0 && \text{for } j = k+1, \ldots, n. \end{aligned}$$

Exercise 3 (Fractional Multi-Knapsack)

In the Multi-Knapsack problem, we are given m knapsacks, each having a capacity W_i for $i = 1, \ldots, m$, n items each having weight w_j for $j = 1, \ldots, n$, and costs c_{ij} when item i is packed into knapsack j. We may assume that $\sum_{i=1}^{m} W_i \geq \sum_{j=1}^{n} w_j$. The task is to pack *all* items into knapsacks such that all knapsack capacities are obeyed and the total cost is minimized.
In the Fractional Multi-Knapsack problem, it is allowed to assign parts of the items to the knapsacks.
Give a combinatorial polynomial-time algorithm that solves this problem.
Hint. Think of the Minimum Cost Flow problem.