

## Algorithms and Complexity. Exercise session 5

### Flows. Reductions

#### Altered flow

- a) Describe an efficient algorithm that finds a new maximum flow if the capacity of a particular edge *increases* by one unit.  
Algorithm time complexity should be linear, ie  $O(|V| + |E|)$ .
  - b) Describe an efficient algorithm that finds a new maximum flow if the capacity of a particular edge *decreases* by one unit.  
Algorithm time complexity should be linear, ie  $O(|V| + |E|)$ .
- 

**Quick bin packing** *Bin packing* is the following problem. You are given  $n$  objects, each weighting between 0 and 1 kg, and a number of boxes to place the objects in. The goal is to find the minimum number of boxes needed to store  $n$  objects with no box containing more than 1 kg.

This is a well-known problem which is difficult to solve exactly (it's a so called *NP-complete problem*). Therefore, it might be sufficient to find a solution which is not optimal, by using the following simple algorithm:

Assume both objects and boxes are numbered from 1 to  $n$ . Pick one object at a time (sequentially) and put it in the first box which can handle it (ie the box with a remaining weight which can handle the object).

Your task is to describe how this algorithm can be implemented so that it runs in time  $O(n \log n)$  (in the worst case with unit cost). To achieve this, you will have to build a heap-like data structure in which you can quickly look up the first box that should hold the current object.

---

**Negative reduction** In previous exercise we described an algorithm that finds an approximate solution to bin packing problem. The algorithm works by placing each object in the first box that can handle it. The goal was to implement the algorithm in time  $O(n \log n)$ . Show that  $\Omega(n \log n)$  is a lower bound for the algorithm time complexity.

---

**Positive reduction** A useful way to solve problems is by finding a reduction to a problem which you already know how to solve. You should use this method to solve the following problem.

INPUT: A connected undirected graph  $G = (V, E)$  and a positive integer  $K$  between 1 and  $|V|$ .

PROBLEM: Is it possible to remove  $K$  edges from graph  $G$  to make it disconnected (ie. divided into connected components)?

---

**Reduction between decision-, optimization- and construction problems** Assume that the algorithm  $\text{GraphColouring}(G,k)$  in time  $T(n)$  (where  $n$  is the number of vertices in  $G$ ) is 1 iff the vertices of  $G$  can be colored with  $k$  colors and no edge has both ends of the same color.

- a) Construct an algorithm that given a graph  $G$  with  $n$  vertices determines the minimum number of colors needed to color  $G$ . The time complexity should be  $O(\log n \cdot T(n))$ .
  - b) Construct an algorithm that given a graph  $G$  with  $n$  vertices colors each vertex with the minimum number of colors in time  $O(P(n)T(n))$ , where  $P(n)$  is a polynomial.
-