Exercise Set 8

Exercise 8.1:
Consider the Placement Legalization Problem with \( y_{\text{max}} - y_{\text{min}} = 1 \). We are given an infeasible placement \( \hat{x} : C \to \mathbb{R} \). Show that there are instances for which there is no optimum solution which is consistent with \( \hat{x} \), i.e. such that \( x(C) < x(C') \Rightarrow \hat{x}(C) \leq \hat{x}(C') \).

(4 points)

Exercise 8.2:
Consider the following variant of the Single Row Placement With Fixed Ordering problem, in which we minimize the bounding box net length:

\[
\text{Input: } \text{A set } C = \{C_1, \ldots, C_n\} \text{ of circuits, widths } w(C_i) \in \mathbb{R}^+, \text{ an interval } [0, w(\Box)], \text{ s.t. } \sum_{i=1}^n w(C_i) \leq w(\Box). \text{ A netlist } (C, P, \gamma, N) \text{ where the offset of a pin } p \in P \text{ satisfies } x(p) \in [0, w(\gamma(p))]. \text{ Weights } \alpha : N \to \mathbb{R}_+. \\
\text{Task: } \text{Find a feasible placement given by a function } x : C \to \mathbb{R} \text{ s.t. } 0 \leq x(C_1), x(C_i) + w(C_i) \leq x(C_{i+1}) \text{ for } i = 1, \ldots, n-1 \text{ and } x(C_n) + w(C_n) \leq w(\Box), \text{ that minimizes } \\
\sum_{N \in N} \alpha(N) \cdot \text{BB}(N).
\]

Show that there exist \( f_i : [0, w(\Box)] \to \mathbb{R}, i = 1, \ldots, n \), piecewise linear, continuous and convex, such that we can solve this problem by means of the Single Row Algorithm.

(4 points)

Exercise 8.3:
Give a detailed proof of Corollary 3.27, i.e. show that the variant of the Placement Legalization Problem in which the goal is to minimize \( \sum_{C \in C} |x(C) - \hat{x}(C)| \) is strongly NP-hard even if \( S = \emptyset \) and \( y_{\text{max}} - y_{\text{min}} = 1 \).

(3 points)

Exercise 8.4: Programming exercise
Implement the dynamic programming buffering algorithm proposed by van Ginneken, with runtime \( O(\|L\|^2 \|V(A)\|^2) \). Your program will take one argument, the path to a file which contains the input, and print the result to the standard output. The task is to find a buffering that maximizes the worst slack. Buffers can be placed only at vertices \( v \) with \( |\delta^+(v)| = |\delta^-(v)| = 1 \).

\text{Input: the first line of the input contains the number of vertices } n = |V(A)|. \text{ The set of vertices is } \{0, \ldots, n-1\}. \text{ Each of the following } n-1 \text{ lines contains four natural numbers } v \ w \ C_e \ R_e, \text{ encoding an edge } e = (v, w) \text{ with capacitance } C_e \text{ and resistance } R_e. \text{ You may assume that this graph is an arborescence rooted at } 0, \text{ in which every leaf is a sink.}

Turn page!
The next line contains the number of buffer types $|L|$ in the library. Each of the following lines contains two natural numbers $C_l$ $R_l$, where $C_l$ is the input capacitance and $R_l$ the resistance of a buffer $l \in L$. The delay of a circuit (buffer or circuit at the root) with resistance $R$ is $R \cdot dc$, where dc is the downstream capacitance. For every sink $v$, assume $\text{rat}(v) = 0$, and input capacitance $C_v = 1$. Assume that the root has resistance $R_0 = 1$.

**Output:** the output must start with a line containing the worst slack and the number of buffers $b$. This should then be followed by $b$ lines, consisting of two integers $v$ $l$, which means that a buffer of type $l$ is used at vertex $v$.

The program must be written in C or C++ (you are allowed to use up to C++11) and must compile and run on Linux. To achieve the maximum score, your program must not leak any memory and must be well documented. It must compile with either Clang $\geq 3.4.2$ or Gcc $\geq 4.8.3$, with `-Wall` `-Wpedantic` `-Werror`, and it cannot link to any other library. You can use any tool available in the standard library.

**The deadline for the programming exercise is July 3\textsuperscript{rd}, 12:00.** Deliver your source code by email.

(20 points)