Prof. Dr. Stephan Held
Markus Ahrens, M. Sc.

Programming Exercise 2

Exercise P.1. Implement the Dijkstra-Steiner Algorithm for 3D grid graphs with future costs.

Your program will be started with a single argument that specifies an instance file. Such an instance file will be in the following format:

$$
\begin{array}{lll}
n & & \\
x_1 & y_1 & z_1 \\
x_2 & y_2 & z_2 \\
\vdots & \vdots & \vdots \\
x_n & y_n & z_n
\end{array}
$$

where all the numbers are space-separated non-negative integers. The first number $n \geq 0$ specifies the number of following terminals. Each subsequent line encodes a terminal at position $t_i := (x_i, y_i, z_i)$. Assume $c(v, w) = \ell_1(v, w)$ which will not be encoded explicitly. Your task is to find a shortest rectilinear Steiner tree in the Hanan grid on the terminals.

Your program shall check for correct input and return exit code 1 if this is not the case. In particular, you should detect unreadable instance files and instance files in a wrong format. If the input is valid, your program must compute the length of a shortest Steiner tree and print this value to the standard output.

The program should achieve the proven theoretical running time up to the running time of the heap. You do not have to implement a Fibonacci heap; you may use simpler heap structures instead (providing $\mathcal{O}(\log n)$ heap operations e.g. as available in the standard library). Your implementation has to use some future costs e.g. the bounding box to the remaining terminals. You should expect test instances of up to 20 terminals, in which case sets of terminals are most efficiently stored in `std::bitset`'s.

The program must be written in `C` or `C++` and must compile and run on Linux. You are allowed to use any `C++` standard including `C++17`. You can use any tool available in the standard library. Your program must compile with either Clang (any version $\geq$ 3.4.2) or Gcc (version $\geq$ 4.8.3) with `-Wall -Wextra -Wpedantic -Werror` and cannot link to any external library.

To achieve the maximum score, your program must not leak any memory and must be well documented. Bonus points will be rewarded for beautiful implementations and effective speedup techniques.

(36 + 10* points)

Deadline: June 25th, via email to ahrens@dm.uni-bonn.de. The websites for the lecture with all exercises and test instances can be found at:

`http://www.or.uni-bonn.de/lectures/ss20/chipss20_ex.html`