# TI2306 Algoritmiek, Lab Assignment 4

Bart Ziengs
bziengs
4391799

January 22, 2017

1. If one would choose to compile each function of the three function one would obtain $(10 + 15) + ((28 \times 4 + 10)) + ((5 \times 6) + 10) = 187$

2. I would define a (minimum) cut $(A, B)$ in which all the to-compile functions $\in A$ en the others $\in B$. In addition, the following variables are used:

   $C_f$, **compile** time for a certain function $f$
   $R_f$, **run** time for a certain function $f$
   $T_f$, number of total **times** a certain function $f$ is called by a function $j$
   $T_{f \to j}$, number of total **times** a certain function $f$ calls a function $j$
   $o$, The overhead costs for when a compiled function calls a non-compiled one and vice versa.

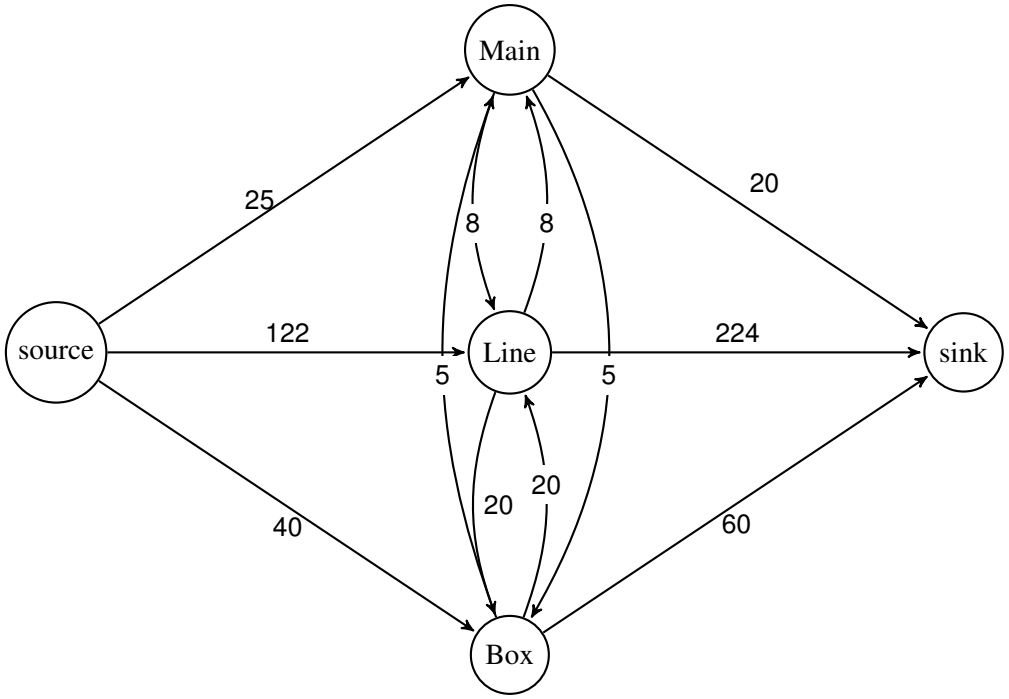
   My formula for what we aim to optimize would be:

   $$\underbrace{\sum_{f \in A} C_f + (R_f \times T_f)}_{\text{compiled function}} \quad + \quad \underbrace{\sum_{f \in B} R_f \times T_f}_{\text{uncompiled functions}} \quad + \quad \underbrace{\sum_{f \in A, g \in b} T_{f \to g} \times o}_{\text{total overhead}}$$

3. To properly model this as a network flow problem, I see it like the situation described in section 7.10 of the book. The following would need to hold:

   - Construct for each process a node and add these to a graph.
   - Add an source node $source$ and a sink node $sink$ to the graph.
   - Construct for each node $n$ an edge $\{source, n\}$ with capacity $C_f + R_{f,compiled} \times T_f$ as defined above.
   - Construct for each node $n$ an edge $\{n, sink\}$ with capacity $R_{f,uncompiled} \times T_f$ .
   - Construct for the amount times $c$ that an function $a$ calls $b$ an two edges $\{a, b\}$ and $\{b, a\}$ with capacity $c * o$. Two edges are needed because if you would obtain a minimum cut $(A, B)$ you would need to determine the capacity by summing up all edges that go from $A$ to $B$ so you need a double edge.
   - Now the minimum cut $(A, B)$ can be determined. The capacity of this cut will be the minimal outcome of the formula derived at **2.**. This is also equal to the maximum possible flow in this specific flow network.

An example is given in the figure below, this is the situation from question **1.** and has three functions




It can be seen that the minium cut $\{\{source\}, \{main, line, box\}\}$ is equal to 187.

4. The pseudo code is shown below. For each process a node is constructed with the capacities as defined in the previous question. Because the main function is always called once, we need to increment the calls to main with one. Subsequently we can deal with all the function calls, we create two directed edges because this is necessary when you want to calculate the maximum flow.

5. Because the number we want to minimize is: (as stated in **2.**)
   $\sum_{f \in A} C_f + (R_f \times T_f) + \sum_{f \in B} R_f \times T_f + \sum_{f \in A, g \in b} T_{f \to g} \times o$
   And this is equivalent to Maximizing:
   $\sum_{f \in B} C_f + (R_f \times T_f) + \sum_{f \in A} R_f \times T_f - \sum_{f \in A, g \in b} T_{f \to g} \times o$
   And that is exactly what we want to achieve.

6. See weblab for

7. last question.

# References

[1] J. Kleinberg, É. Tardos, Algorithm Design, Pearson Addison Wesley, 2006.

[2] Blackboard slides

**Algorithm 1** Minimum run time with maximum flow

let $g$ be a graph
add source node $s$ to $g$
add source node $t$ to $g$

**for each** function $f$ **do**
    create a node
    add edge $(s, f)$
    add edge $(t, f)$
**end for**

increment the function calls to main with 1

**for each** function call from $f$ to $g$ **do**
    add edge $(f, g)$
    add edge $(g, f)$
**end for**

Maximize the flow in $g$

**print** the path of the nodes that the maximum flow traverses
**return** $sum$