For many, there is very little meaning or beauty in a screen of computer code [7]. By converting order of execution into a directed cyclic network, I hope to make the comparison of algorithms more accessible while maintaining the depth to generate insights [9].

Code execution is rarely a linear process, often jumping to different parts of the code to repeat tasks and reuse functionality [3]. A network reveals these jumps while remaining accessible to a wider audience. Running six classic algorithms in Python, I gathered line-execution order and duration, converting them into network variables for display on a D3.js website. [11].

Beyond showing the order of instructions, I wanted to highlight the most important ones. For this, I turned to visual variables, following Tufte's guidelines for "graphical excellence", achieving a dense representation of many values [10] [8]. Pairs of nodes which are often executed together have a shorter edge length. I used node colour to represent execution duration, using darker tones for clearer differentiation [4]. The size of the node represents the number of times a line was executed, indicating the most important lines.

The resulting network can also be considered a unique portrait of an otherwise abstract algorithm. By selecting a network representation, I hope to mirror the approaches seen with Du Bois' and Lombardi's work who combined complex data with striking visualisations [1] [6] . With my network visualisations I aim to achieve the same by highlighting the beauty behind otherwise mundane blocks of code.

An alternative visualisation which I considered was the scatter graph as it would make reading off precise values easier. While suitable for experienced engineers, a scatter plot would be harder for wider audiences due to the lack of ordering.

One difficulty when representing values with visual variables is representing the precise value. For example, runtime percentage is hard to judge from colour alone. I resolved this by developing an interactive vectorised representation [12]. This made the data easier to explore with high resolution zoom while providing precise values when hovering the mouse.

Another difficulty I encountered was how to represent the connection between lines of code and the nodes. One option could have been to overlay each node with the corresponding line but this would have made the graph noisy and more difficult to read. Instead, I leveraged the interactive medium and highlighted corresponding code when hovering over nodes. Using a bright yellow on the darker backgrounds stood out clearly in the visual hierarchy while following Imhof's first rule [13] [14]. When highlighting the connection

from either the code to the network or vice versa, I used the same yellow, following the Principle for Interaction Design with consistency [2].

While I believe that using a network was the right decision, the nodes and connections can be overwhelming at first. The use of animation could help to build up the network and make the path between nodes clearer. Some code knowledge is still required, making the visualisation more suited to beginner programmers.

References

[1] Creating an initial impression from a glance instead of focus on precise comparisons Du Bois. My article, 2006.

[2] Highlighting code and nodes:Creates a visual heirachy. My article, 2006.

[3] https://arxiv.org/html/2404.02377v1. Exploring the impact of source code linearity on the programmers' comprehension of api code examples, 2006.

[4] Increments and decrements: Luminance discrimination. light and dark, 2006.

[5] Nobody Jr. My article, 2006.

[6] Lombardi. My article, 2006.

[7] What makes code hard to understanad. My article, 2006.

[8] Dense representation which is not possible in any other format Source: Present many numbers in a small space Tufte coined the term "graphical excellence". My article, 2006.

[9] The Functional Art. An Introduction to Information Graphics Source: Cairo, Alberto and Fig 3.2. Visualization (2013), 51. My article, 2006.

[10] Achieved this dense representation of values with Source: Fig 01: Bertin's visual variables. My article, 2006.

[11] 4 types of data nominal ordinal interval ratio. My article, 2006.

[12] The use of vectorised graphics allows for easy zoom. My article, 2006.

[13] Good website design (clear buttons and consistency in colours). Source: Principles for interaction design, 2006.

[14] Highlight with bright Imhof rule 1 2007. My article, 2006.