Coding Task

Quantinuum

Introduction. A group of labs wish to collaborate on synthesising some complex materials. These are synthesised via a step by step process, each step combining two components in one of the labs. Your task is to design and implement an algorithm that schedules these steps across the different labs, with the goal of reducing the time it takes to synthesise the material.

Problem definition. You will be provided an undirected graph describing the steps required to synthesise the material – we will refer to this graph as the *blueprint*. An example blueprint is provided in Figure 1. The vertices of the graph identify the different components. An edge between two vertices indicates that the corresponding components need to be combined. When two components are combined we update the blueprint as shown in Figure 2, merging the components to create a new one. The time required to combine two components a and b can be accurately estimated as:

$$T(a, b) = 2^{\delta(a)+\delta(b)-\gamma(a,b)} \quad (1)$$

where $\delta(a)$ is the degree of vertex a and $\gamma(a, b)$ is the number of edges a and b share. For instance, the combination depicted in Figure 2 would take $T(a, b) = 2^{4+5-2}$.

- Each combination must be assigned to a single lab.
- Each lab can carry out a single combination at a time.
- A combination can only begin once its two components have been created.
- Combinations that do not share components may be processed in parallel by different labs.
- All labs are equally capable.
- There is no other contribution to time apart from equation (1) – you can assume that transportation of components between labs is instantaneous.

Goal. Design and implement an algorithm under the following specifications:

- *Input*. A number of labs N and an undirected graph G – the blueprint.
  - A collection of example graphs are provided in the folder blueprints. Each one is provided in a separate txt file as a collection of edges.
  - All blueprints were generated by the script blueprint_generator.py.
- *Output*. A schedule of combinations and the lab each combination is assigned to.
  - A schedule is valid only if the final state of the blueprint (updated as in Figure 2 after each combination) contains a single vertex.
- *Metric*. Minimise the time it would take to complete the schedule.

You may implement the algorithm in Python, C/C++ or Rust. You may use any library you wish, but make sure there is enough original code for us to have a meaningful conversation during the interview.

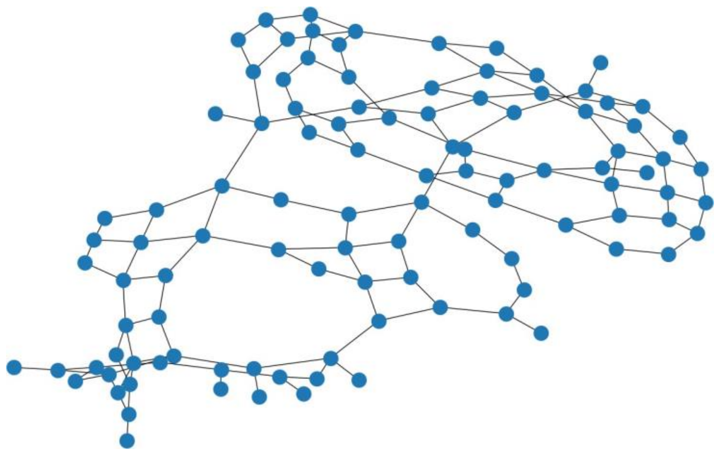
Figure 1: An example blueprint.

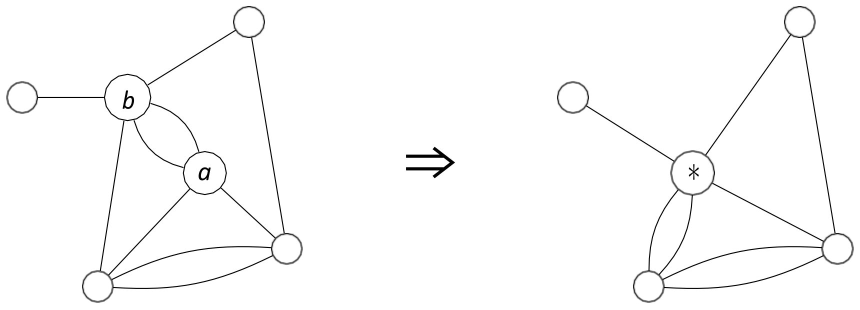

Figure 2: Combination of components a and b in a blueprint. All edges between a and b are removed. All edges connected to a or to b now connect to the new merged vertex $*$.