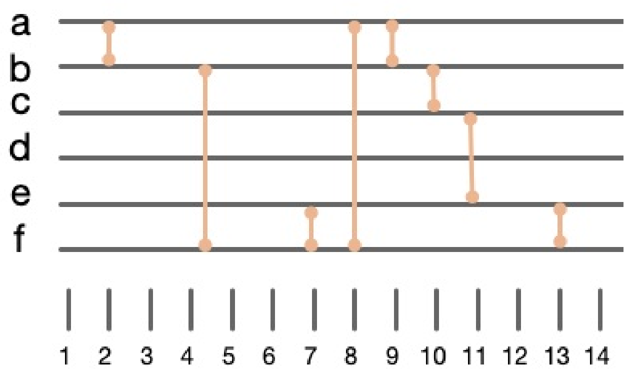
Experimenting with Dynamic networks

Several libraries exist to manipulate temporal networks, but most of them are not fully mature. I propose to use `tnetwork`, that can be installed using `pip`.
Documentation and examples: https://tnetwork.readthedocs.io.

1. Paths in dynamic networks



   (a) What is/are the **shortest** path from (a,1) to (e,14)?

   (b) What is/are the **foremost** path from (a,1) to (e,14)?

   (c) What is/are the **fastest** path from (a,1) to (e,14)?

2. Characterizing a dynamic network as a sequence of snapshots

   (a) Using `tnetwork`, load the dynamic network of interactions between individuals in a hospital, collected by the sociopatterns project as a sequence of snapshots. (1 snapshot=20s, analysis over a few days)

   You can do this with: `g = tn.graph_socioPatterns_Hospital(format=tn.DynGraphSN)`

   You can read about this dataset and other information about the sociopatterns project on the following page: http://www.sociopatterns.org/datasets/hospital-ward-dynamic-contact-network/

   (b) Count the number of snapshots. You can obtain the list of times at which snapshots occur with `g.snapshots_timesteps()`. Count the number of seconds between the first and last snapshots, and deduct the number of days of the analysis.

   (c) Obtain the graph corresponding to the first snapshot. You can use the `g.snapshots()` method which return a sorted dictionary, in which keys are times of snapshots and values are snapshots represented by a `networkx` graph object.

   (d) Compute the number of nodes and edges in this snapshot, and plot it.

   (e) What is the total number of interactions among all nodes at all time (edge-time)? What is the total number of different nodes (not node_time)? Deduct the average number of interactions per node. (you can use a `for` loop, and `set` or `collections.Counter` to handle repeated elements.

   (f) Plot the evolution of the number of nodes, number of edges and density along time. Plot the evolution of the degree of one particular node. Would you say that this network is rather stable or unstable? Be careful in your interpretation, that some snapshots are missing, if they correspond to a timestep with no nodes and no edges.

(g) You can plot several graphs simultaneously using the following function:
`graph = tn.plot_as_graph(g,ts=[1254386420,1254386440,1254386460])`, with ts being a list of timestamps.

(h) You can plot the presence of nodes in snapshots with the `tn.plot_longitudinal(g,to_datetime=True)`. The `to_datetime` parameter allows to transform datetime to their corresponding date. Be careful, it might take about 30s on the full graph. (The plot appears twice in Google colab, known bug)

(i) Compute the aggregated graph using `g.cumulated_graph()`. Plot this graph, compute its number of nodes, edges, density.

(j) Compute snapshots aggregating activity every hour using `g2 =g.aggregate_time_period("hour")`. And then by day `"day"`. Analyze the resulting networks in a way similar to the first graph.

(k) For the 3 versions, compute the average edge similarity between each successive snapshot, using Jaccard similarity (Intersection of edges divided by union of edges. You can use python set operators.)

(l) Would you say the the original data is stable or unstable ? What about aggregated versions ?

(m) Write a function that compute (you have to write the code yourself) the dynamic version of "number/quantity" of nodes N, "number/quantity" of edges L for a given period (all snapshots between an initial and a final timestamps)

(n) write another function for the density d, in the L^2_{max} and in the L^3_{max} variant. Compare.

(o) Compute these quantities for the whole period, with the original data and with the hourly version.

(p) Compute it now hour per hour, in the original data. Compare.

3. Going further: **Dynamic Communities**

(a) Read the documentation of `tnetwork` about the detection of dynamic communities: https://tnetwork.readthedocs.io/en/latest/notebooks/demo_DCD.html.

(b) Apply two methods on the primary school dynamic network aggregated every hour and compare the results. (`tnetwork.graph_socioPatterns_Primary_School(format=tn.DynGraphSN)`

(c) Compute static communities on the cumulated graph (aggregated over the whole period). Compute communities on the graph in its original form (no aggregation). What do you think of the communities found using those three different approaches?