Step 3: Compute a force directed layout

Application of a force When programming the forces, do not forget that they are vectors. So, to update a position of a point you perform an operation $p := p + \vec{v}$, where p is a point, and $\vec{v}$ is the vector specifying the force. The vector has a magnitude $||\vec{v}||$ and direction, usually expressed by a vector of unit length. Refer to Figure 1 and Figure 2 for a wrong and a correct application of a force, respectively.

```python
force = 2
position = np.array([0.0, 1.0])

position = position + force  # force gets converted silently to [2, 2]
```

Figure 1: Incorrect application of a force. You end up with position $== [2, 3]$. Instead, your force must be a vector.

```python
# Instead, force must be a vector (magnitude and direction)
# In this example, we're applying a force from point p_u to point p_v.

p_u = np.array([0.0, 0.5])
p_v = np.array([0.6, -1.2])

direction = p_v - p_u  # p_v - p_u is a vector *from* p_u *to* p_v.
# We need a unit vector for calculations.
direction_unit = direction / np.linalg.norm(direction)

force_magnitude = ... # Compute your force(s) here
force = force_magnitude * direction_unit

p_u = p_u + force  # feel free to include step size (delta from slides)
```

Figure 2: Correct application of a force.

Number of iterations Explain in detail how do you determine when to stop the layout algorithm. You have a few options: - observe how the layout looks like and decide on the number of iterations manually; - check when the node movement is below a given threshold; - measure some quality metric (e.g. stress), analyse them and find an iteration after which no major improvement can be seen. How do you stop your layout algorithm? Explain your choice in detail.

Constants Explain carefully the choice of the constants, such as the ideal edge length and the force constants. Check how your algorithm performs for different settings and if you find that a few of them give nice but different results, report this as well.

Reporting resulting visualizations Investigate how the layout improves with the number of iterations. Additionally to the initian and final layout, report a few intermediate results that show how your layout evolves and do not forget to annotate them with the iteration count.

Complexity Analyse the worst-case computational complexity of your implementation.