Implementation of Johnson's algorithm for shortest path with negative weights

Analysis

- `Dijstra` algorithm doesn't work with negative weights always, and for certain structure of the graph it fails.
- `Bellman-Ford` does work with negative weights but is slower than `Dijkstra` and is not convenient for large graphs.
- `Johnson` algorithm uses `Dijkstra` and `Bellman-Ford` algorithm as subroutines
- `Johnson` algorithm was designed for negative edge weights
- The key of `Johnson's` algorithm is to re-weight all the weights into positive territory.

Solution

- Given that petgraph already implements `Dijstra` and `Bellman Ford`, the only task that is left is to integrate both algorithms to assemble `Johnson's` algorithm.
- Using negative weights is a requirement for testing our solution

Implementation details

Source code: https://github.com/nickhash/johnson

Tests

For more effective testing, we search for already existing tests of `Johnson's` algorithm. Here are three links that have sample graphs and every step of the algorithm is explained, providing sample values:

https://www.geeksforgeeks.org/johnsons-algorithm-for-all-pairs-shortest-paths-implementation/
https://www.javatpoint.com/johnsons-algorithm (*with typos*)
https://algs4.cs.princeton.edu/44sp/

Our implementation generates the same result as in the above links.

Note: The second link has many errors in the pictures (which are evident after following the chain of mathematical operations presented on the webpage), so any mismatch with our own values is only due to human errors of the webmaster.

First two cases have been coded as part of Unit tests and can be executed with command:

```
[user@host]$ cargo test
```

References

- Pethgraph overview: https://thobbs.cz/rust-play/petgraph_review.html
- Internals of Pethgraph: https://thobbs.cz/rust-play/petgraph-internals.html