SSSP

Single Source Shortest Paths

- The SSSP (abbr. for Single Source Shortest Paths) algorithm described in the Google Pregel paper was used.
- Introduces IO usage, partitioning based on hashing of vertextID, and collective communication.

Short summary of the algorithm

- Initialize all vertices' cost to reach it to INFINITY, just the start vertex will have cost 0.
- Initially send the new cost to all adjacent vertex containing the new cost plus the edge weight between them
- Reviewing messages: if the cost coming from a message is lower than the actual cost, update the cost and send a message to the adjacent vertices, containing the new cost plus the edge weight between them (similar to the last step)
- Repeat the last step until no updates can be made anymore.

Usage

```
bin/hama jar ../hama-0.x.0-examples.jar sssp <start vertex> <input path> <output path> [number of tasks]
```

You need to provide a start vertex name from where the computation should start calculating the shortest paths, scroll down how to provide an input file for it.

Submit your own graph

You can transform your graph as a adjacency list to fit into the input which Hama is going to parse and calculate the SSSP.

The file that Hama can successfully parse is a TextFile that has the following layout:

```
Berlin\tFrankfurt:20\tMunich:50
Frankfurt\tBerlin:20\tMunich:10
Munich
```

This piece of text will adjacent Berlin to Frankfurt (with edge weight of 20) and Munich (with edge weight of 10). Munich is a dangling node, it has no outlinks. As you can see a vertex is always on the leftmost side (we call it the key-site), and the outlinks (to which other vertex it is connected to) are seperated by tabs (\t) as the following elements. SSSP needs edge weights, you must provide them by separating the name of the vertex with a colon ":". The weight must be an integer.

Make sure that every vertex's outlink can somewhere be found in the file as a key-site. Otherwise it will result in weird NullPointerExceptions.

Then you can run sssp on it with:

```
bin/hama jar ../hama-0.x.0-examples.jar sssp Berlin /tmp/input.txt /tmp/sssp-output
```

Note that based on what you have configured, the paths may be in HDFS or on local disk.

Output

After the job ran you can see a small snapshot of what the algorithm calculated, for the textfile above you should see:

```
12/02/24 16:47:48 INFO bsp.BSPJobClient: Current supersteps number: 5
12/02/24 16:47:48 INFO bsp.BSPJobClient: The total number of supersteps: 5
Berlin | 0
Munich | 30
Frankfurt | 20
Job Finished in 4.018 seconds
```

On the left side you see your vertex name and on the right the cost which is needed to get to that vertex.