**Report: Exercises 4**

Sevde Yanik & Sima Esmaeili

**Problem 1**

Using MPI in C++, the Vogel-Strauß algorithm is implemented. The algorithm first puts all messages into network using asynchronous send operations and then it receives everything that arrives. The implementation is tested by running it on multiple processes where each process sends a unique message.

The implementation is explained step-by-step:

1. At the beginning MPI is initialized and the rank and size of the MPI communicator is retrieved.
2. A unique message is generated for each process based on its rank.
3. Using a loop, each process asynchronously sends its message to all other processes, except for itself.
4. Another loop is used to receive messages from other processes. Again, each process receives messages from all other processes, except for itself.
5. All messages are printed to the console together with the rank of the process that retrieved it.
6. MPI is finalized.

**Problem 2**

Run the Hypercube Gossiping algorithm with 23 = 8 processing elements represented in binary as (000, 001, 010, 011, 100, 101, 110, 111). Assume that each processing element starts with a different character from {A,B,C,D,E,F,G,H}. Show all steps of the execution until all processing elements have all the available characters.

Start:

|  |  |
| --- | --- |
| PE | Character |
| 000 | A |
| 001 | B |
| 010 | C |
| 011 | D |
| 100 | E |
| 101 | F |
| 110 | G |
| 111 | H |

A picture containing text, screenshot, font, algebra

Description automatically generated

Running the Hypercube Gossiping algorithm using the union operation provided, Step 1:

1. PE 000(A) receives B from PE 001, A . B = AB
2. PE 001(B) receives A from PE 000, B . A = BA
3. …

Result:

|  |  |
| --- | --- |
| PE | Character |
| 000 | AB |
| 001 | BA |
| 010 | CD |
| 011 | DC |
| 100 | EF |
| 101 | FE |
| 110 | GH |
| 111 | HG |

Step 2:

1. PE 000(AB) receives CD from PE 010, AB . CD = ABCD
2. PE 001(BA) receives DC from PE 011, BA . DC = BADC
3. …

Result:

|  |  |
| --- | --- |
| PE | Character |
| 000 | ABCD |
| 001 | BADC |
| 010 | CDAB |
| 011 | DCBA |
| 100 | EFGH |
| 101 | FEHG |
| 110 | GHEF |
| 111 | HGFE |

Step 3:

1. PE 000(ABCD) receives EFGH from PE 100, ABCD . EFGH = ABCDEFGH
2. PE 001(BADC) receives FEHG from PE 101, BADC . FEHG = BADCFEHG
3. …

Result:

|  |  |
| --- | --- |
| PE | Character |
| 000 | ABCDEFGH |
| 001 | BADCFEHG |
| 010 | CDABGHEF |
| 011 | DCBAHGFE |
| 100 | EFGHABCD |
| 101 | FEHGBADC |
| 110 | GHEFCDAB |
| 111 | HGFEDCBA |

At this point, all processing elements have all the available characters. The final state is:

* PE 000: ABCDEFGH
* PE 001: BADCFEHG
* PE 010: DCABGHFE
* PE 011: DCBAHGFE
* PE 100: EFGHABCD
* PE 101: FEHGBADC
* PE 110: GHFEDCAB
* PE 111: HGFEDCBA

The Hypercube Gossiping algorithm using the union operation has finished running. Now, every processing element has gathered all the messages that were initially given to them.

**Problem 3**

Consider a directed graph, where the adjacency arrays store the outgoing edges for each vertex. Give pseudo-code to build a reverse graph adjacency array using the map-reduce paradigm (the reverse graph should have the same set of vertices, but the edges should be reversed).

**Problem 4**

Consider an undirected graph G = (V,E) and a partitioning function Π : V → {1, 2, . . . , k} that assigns each node to one of k blocks. Provide pseudo-codes to implement each of the following tasks using the map-reduce paradigm.

1. a  Compute the edge-cut, i.e., the total weight of edges that run between blocks. (2 points)
2. b  Compute the cardinality of each block. (2 points)
3. c  List all boundary nodes in each block, i.e., the nodes contained in a block that have at least one edge to a node contained in another block. (3 points)
4. d  Calculate the weight of the edges of the quotient graph associated with the partition. (3 points)