

Assignment #1 (50 points)  
 Assigned: Friday July 12, 2024  
Due: Friday July 26, 2024

Programming Assignment #1 — FM Partitioning

### Abstract

Write a C*++/*Java*/*Pythonprogram that partitions a hypergraph *G* = (*V*, *E*) into 2 partitions.

### Outcomes

After successfully completing this assignment, you should be able to:

* Explain the steps of the FM partitioning algorithm.
* Explain the run-time complexity of the FM partitioning algorithm.
* Implement the FM partitioning algorithm in the language of your choice to partition a hypergraph into 2 partitions with the minimum cut size as the objective.
* Experiment different strategies in the FM partitioning algorithm to explore their pros and cons.

### Before Starting

* Read Chapter 2 of the textbook and the lecture slides for Graph Partitioning.
* Familiarize the high-level programming language and the programming development environment of your choice.
* Think about an efficient data structure to store a hypergraph and traverse it fast.

### The Assignment

Write a computer program that takes a netlist represented by a weighted hypergraph and partitions it intotwo partitions. Each node is associated with an area value and each edge has an edge cost. Your program should minimize the total cost of the cut set, while satisfying the area constraint that the total area of partition 1 should satisfy the balance criteria as described in the class. That is, if the area sum of all the nodes is *A*, then the area of partition 1 should be greater than or equal to *ratio\_factor* ⋅A – *a*max and less than or equal to *ratio\_factor* ⋅A + *a*max, where *a*max is the maximum value among all cell areas. The program should prompt the user for the value of *ratio\_factor*.

#### Assumptions and Requirements of the Implementation

1. Your program should not have any limitation on the maximum number of nodes and the edges of the hypergraph. Each hyperedge could connect any subset of nodes in the hypergraph.
2. Each node area is a non-negative integer, and each edge cost is a non-negative floating- point value.
3. All the ids are 0-based. Namely, the id of the first element is 0, instead of 1.
4. The output of each partition should include the list of node ids, sorted in the ascending order.
5. The partition with the smaller minimum node id is listed first in the output.
6. Use balance criteria as the tiebreaker when there are multiple cell moves giving the maximum gain, as described in the class.
7. Use the input and output formats given in the Sample Test Cases section.

### Sample Test Cases

Test1:

Please enter the number of nodes: 4

Please enter each of the 4 nodes with its id and the node area:

0 1

1 1

2 1

3 1

Please enter the number of edges: 3

Please enter each of the 3 edges with the number of connected nodes and their node ids, followed by the edge cost:

2 0 1 1

2 1 2 3

2 2 3 1

Please enter the percentage of the ratio factor: 50

The node ids of the partition 0 are 0

The node ids of the partition 1 are 1, 2, 3

The total cut cost is 2

Test2:

Please enter the number of nodes: 4

Please enter each of the 4 nodes with its id and the node area:

0 1

1 4

2 2

3 1

Please enter the number of edges: 3

Please enter each of the 3 edges with the number of connected nodes and their node ids, followed by the edge cost:

3 0 1 2 5

3 0 2 3 3

3 0 1 3 4

Please enter the percentage of ratio factor: 50

The node ids of the partition 0 are 3

The node ids of the partition 1 are 0, 1, 2

The total cut cost is 7

### Implementation Notes

None.

### Algorithm Requirement

The FM algorithm used should be the one discussed in the lecture.

### Deliverables

Write a document called **README.txt** or **README.doc** summarizing your program, how to run it, and detailing any non-trivial problems that you encountered and how you solved them. Zip your source files and submit the zipped file to TA by the deadline.

### Grading

This assignment is worth 100 points. *Your program* must *compile without errors to receive* any *credit.* Full credit will be given if the program in your submission:

* Can be compiled successfully without warnings.
* Produces correct results with the instructor’s test cases.
* Satisfies the algorithm and implementation requirements.
* Is sufficiently documented in the **README** file.