Homework #2
EEL 4932: Computer-aided Design of VLSI:
Rickard Ewetz

Programming exercise #1
The due date is listed on webcourses.
(Turn in a hardcopy of the report during the first class after the due date.)

0) Overview

The objective of HW#2 is to implement a 2-way min-cut partitioner. Your program will read an input file (01.in, ..,05.in) and produce an output file (01.out,..,05.out). Each input file contains nodes and edges. The nodes all have size one but the edges may have different weights. The output file will contain the achieved cut size and an assignment of each node to one of the two partitions. The size of each partition is required to be between 40% and 60% of the total size of all the nodes.

The remainder of this document is organized, as follows:
1) The input format
2) The output format
3) Base assignment
4) Bonus assignment
5) What to submit

You are allowed to use any programming language. However, I would encourage you to use C/C++.

1) The input file format

An example of the input file format is shown below. The first row provides the number of nodes. The second row provides the number of edges E. The subsequent E rows gives the details of the edges. “12 4” indicates an edge from node 1 to node 2 with a weight of 4.

```
numNodes 4 //The number of nodes
numEdges 4 //The
1 2  4 // An edge between Node 1 and Node 2 with a weight of 4
1 3 3  // An edge between Node 1 and Node 3 with a weight of 3
2 3 1
3 4 1
```

(The input file may have multiple edges between the same pair of nodes. This can be handled by replacing the two edges with a single edge with a weight equal to the sum of the two edges.)

2) The output file format

An example of an output file is shown below. The first line should provide the cut of the partitioning listed on the subsequent N lines. Each line indicates if node “i” is assigned to partition 0 or partition 1.

```
cut 4
0 // Node 1 is assigned to partition 0
0 // Node 2 is assigned to partition 0
```

1 // Node 3 is assigned to partition 1
1

3) Base assignment

Your program should take the input and output file name as an input from the terminal. For exmaple, if your program is called “hw2”. You should be run your program using the command “./hw2 01.in 01.out”, where “01.in” is the input file and “01.out” is the output file.

a) Read the input file and store it using a matrix graph representation. (5pt)
b) Assign the first N/2 nodes to partition 0 and the remaining N/2 nodes to partition 1. Compute the cut and save an output file with the solution. (5pt)
c) Compute the D values for each node.
d) Compute the gain for each pair of nodes.
e) Swap the pair with the highest gain.
f) Perform an incremental update of the D values. Update the gain values.
g) Implement a partitioner based on repeating the steps e) and f). This is a simplified version of the KL algorithm. Save the obtained partition. (20 pt)

You are required to apply the algorithm to all input files. However, it is acceptable that the algorithm crashes for large input files that require too much memory.

4) Bonus assignment

a) Use an edge representation for the graph. Can you process larger input files? (5pt)
b) Implement the largest partial sum feature and locking of nodes. (5pt)
c) Move a single node instead of swapping pairs of nodes. Is the algorithm faster? (5pt)
d) Implement a multi-level partitioner to handle large input files. (5pt)

5) What to submit

1) Please submit your source code on webcourses.
2) Please submit a 1 page document where you list the part of the assignment that you have completed. You are required to turn-in a hardcopy of the document and upload it to webcourses. Please provide a table where the cut and run-time is reported for the initial partition (after step 3.b) and after the optimization (step 3.g). To claim any bonus points, please report the improvements with respect to (step 3.g). Please provide a description for how to run your code.