COL 380	March 12, 2016
Lab 1	
Instructor: Subodh Sharma	**Due: March 22, 24:00 hrs**

Problem Description

The task in this lab is to parallelize using OpenMP the multi-level graph partitioning problem by resorting to recursive bisection heuristic. Multi-level graph partitioning finds its use in problems such as telephone network design, data mining and clustering, load balancing etc. The k-way graph partitioning problem is defined as follows: Given a graph $G = (V, E)$. with $|V| = N$, partition V into k subsets, $V_1, V_2, \cdots, V_k$, such that:

1. $\forall i, jV_i \cap V_j = 0$ and $\bigcup_i V_i = V$
2. $|c|$ (Edge-cut) is minimized, where $c = \{e \in E \,|\, \forall i, j \in [1, k], e = (v, w),\ v \in V_i,\ w \in V_j\}$,
3. $\frac{max(|V_1|,|V_2|,\cdots,|V_k|)-min(|V_1|,|V_2|,\cdots,|V_k|)}{|V|} \leq 0.05$

A common way to partition a graph is by using a multilevel algorithm. Here a graph G is first *coarsened* to a graph with fewer vertices. Then, the coarsened graph is *partitioned* (using recursive bisection method). The partition is finally projected back to the original graph in the *uncoarsening* phase.

Coarsening: Given a graph $G = (V, E)$, a coarser graph can be obtained by merging adjacent vertices. The edge between two vertices is collapsed and a bigger node consisting of these two vertices is created. This edge merging idea is formally defined in terms of *matchings*. "A *matching* of a graph, is a set of edges, no two of which are incident on the same vertex". Therefore, the next level coarser graph G' is constructed from G by finding a matching of G and collapsing the vertices being matched into bigger nodes. Note that depending on how matchings are computed, the size of the maximal matching may be different. The goal of this phase is: The graph G_0 is transformed into a sequence of smaller graphs $G_1, G_2, ..., G_m$ such that $|V_0| > |V_1| > |V_2| > \ > |V_m|$.

Uncoarsening: In this phase, the partition P_m of a graph G_m is projected back to the original graph by going through the graphs $G_{m-1}...G_1$. The partition refinement heuristics such as vertex swapping are employed to make sure that the resulting partition has a smaller edge-cut.

For more on these topics, refer:

- Multilevel graph partitioning schemes
- Parallel Multilevel Graph Partitioning

Submission

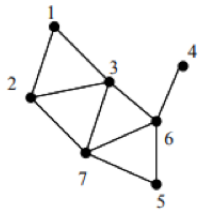
- The assignment can be done in groups of **at most** two.
- The submission of the assignment is divided into 2 phases:
  - **Phase I: Correct implementation**: Deadline: Sunday, 19-03-2018.
    Submit a **correct parallel** implementation for the first phase. A correct implementation would be considered for the correctness component of grading and **only** those teams which submitted by this deadline would be eligible for participating in the phase II. Teams that

did not meet the first deadline can submit by the second deadline, however, they will be graded only for correctness. The evaluation criterion for correctness is as follows: Computed partitions will be checked for validity (i.e. whether or not the constraints (1) and (3) are satisfied).

– **Phase II: Competition**: Deadline: Wednesday, 23-03-2018
Teams are required to improve their code for optimization and efficiency. The scoring will be based on their relative position on the leader-board. There will 2 categories of test cases: public and private. After each submission (before the final deadline) the implementation would be evaluated on public test cases and the results shall be made available on the leader board. After the final deadline, the submissions shall be executed on private test cases and the final rankings would be computed (this is to test the corner cases, if any). The test cases (private or public) will not be released. The score will be computed on the basis of:

1. Size of the edge-cut set.
2. Time of execution.

- **Input Format**: The initial line of the input file will contain the integers N (the number of nodes in the graph), and E (the number of edges). This will be followed by N lines, containing the list of neighbors for the corresponding node (with the nodes numbered from 1 to N). For example:




```
7 10
2 3
1 3 7
1 2 6 7
6
6 7
3 4 5 7
2 3 5 6
```

Figure 1: Left:Image of a graph; Right: Input file

- **Output File Format**: For the example above, if the partition is 1,2,3,7 and 4,5,6, then the output should be: 0 0 0 1 1 1 0.

Leader-board

- The url to the leader-board would be made available on Piazza soon
- We will share a document to fill in team information
- Each team has to fill in their entry numbers
- Every team would be assigned a unique id and their passwords for submission would be available on moodle
- Submit in a zip format (only zip, no tar.gz or rar or any other)
- Zip name should be teamXX.zip where XX represents team id. eg. team12.zip
- When we extract the zip there should be folder with name as teamXX .eg team12 and there should be two scripts in that file, compile.sh and run.sh.
- compile.sh compiles the code

- run.sh takes 2 parameters: *input.file* and *output.file*. All stdin/stdouts would be ignored.
- Program should read the input from *input.file* and store the output to *output.file*.
- If the input format or the output format don't match, then the submission would be considered as invalid.
- A submission queue status shall be maintained which will be publicly available.
- Timeouts for each test cases will be maintained and will be broadcasted before March 19.
- Each submission by the same team would be considered as different versions and the score for the best version shall be considered.
- Every team will have a limit of 10 submissions per day.

Grading Policy

Correct execution of the program (Phase I)	40%
Efficiency of the implementation (Phase II)	60%