EECS 527 - Homework 2

Topic: Empirical comparison
of the flat Fiduccia-Mattheyses partitioning algorithm
and the Multi-level Fiduccia-Mattheyses framework

The partitioning formulation:
- vertex-weighted hypergraphs
  (weights correspond to gate areas -- given in benchmarks)
- no hyperedge weights
- balance constraints: each partition must contain 42%-58% of total weight

Goal: estimate the gap between the performance of flat FM and
multi-level FM; compare those to dart-throwing and
a simple variant of your algorithms from mini-project 1.

Main action items:
- Download and install UCLA Physical Design tools
- Find the FMPart and MLPart packages
- Identify the infrastructure (source code and executables)
  for empirical comparison of FM and MLFM algos
- Perform 20 runs of FM, MLFM, dart-throwing and your simple algorithm
- Collect best cuts and average cuts
- Analyze empirical data and make conclusions

Hand in: a report, including empirical data for FM and MLFM,
dart-throwing and your simple algorithm on the 18 ibm benchmarks

Download site for UCLA Physical Design tools (UCLApack):
http://vlsicad.eecs.umich.edu/software/PDtools/

Note: You may not have enough space on your CAEN systems,
besides CAEN systems have an unusual configuration
which may prevent the installation of this package.
Try using your EECS accounts.

What to do:
- When you install UCLApack, find the packages FMPart and MLPart
- See the "regression" script in each --- it launches sample runs
  of relevant binaries (with .exe extensions). Sample benchmarks
  are also there, and you should be able to solve the same ibm benchmarks
  in .net/.are format
  (you will need to create one-line .aux files and simple .blk files)
- High-level source codes for those binaries are available
  in the same directories, and you can figure out precisely
  what the binaries do (for example, MLPartTest0.exe is produced
  from MLPartTest0.cxx)
- Perform 20 runs of FM and MLFM on each of ibm18 benchmarks
  with prescribed tolerances and record best/average cuts,
  average runtimes per run and memory consumption
  (those numbers are printed by the executables you will use)
- Compare that to dart-throwing
  (you need to implement a dart-throwing algorithm
  for balanced min-cut partitioning; note that your
  random partitions must satisfy balance constraints,

```
     so you might want to use the randomized engineer's method
     described in class)
  - Compare that to the following simple algorithm:
     produce a linear ordering (by BFS), then
     scan "the middle" of the ordering and find
     the smallest cut among those cuts whose balances
     satisfy the required balance constraints
     (recall that we are now dealing with weighted hypergraphs)

What not to do:
  - In this assignment you DO NOT need to write
    any code when working with UCLApack
```