CMPS 2200 Recitation 10

In this lab we'll investigate graph partitioning. You'll implement the sequential version of edge partitioning, use it to compute the number of components in a graph, then extend it to compute the size of each component in the graph.

1. In star contraction, what is the probability that a vertex v with degree d is removed?

put in answers.md

.
.
.

2. Suppose I come up with a contraction algorithm that is guaranteed to reduce $|V|$ by 5 at each iteration. In Big-Oh notation, what is the worst-case number of iterations this contraction algorithm will have?

put in answers.md

.
.
.

3. Complete the implementation of `edge_contract_seq`, which is the sequential version of edge contraction. We iterate through each edge e in the graph. If we have not already selected a different edge e' that neighbors e, then select e. When we select an edge, we can arbitrarily select one vertex to be the super vertex. In this case, let's select `e[1]`, the second node in each edge tuple. We'll have to update `selected_vertices` and `vertex_map` whenever we select an edge. Remember to also insert singleton supervertices into the `vertex_map` as needed.

Test with `test_edge_contract_seq` as well as with `test_num_components`, where we use this implementation to solve the number of components problem.

.
.
.

4. Finally, let's keep track of the sizes of each connected component, instead of just their number. Complete `edge_contract_seq_sizes`. We'll maintain a dict called `sizes` that maps each vertex to the size of its connected component. As we contract the graph, we'll update the `sizes` dictionary whenever we select an edge. We'll update the size of the super vertex based on the size of the vertex added to it.

Test with `test_edge_contract_seq_sizes` and `test_component_sizes`.

.
.
.