The problem of graph sparsification is to construct a graph $G'$ from a graph $G$ with fewer edges, such that certain properties of $G$ are maintained in $G'$.

Consider $L$, the $n$ by $n$ laplacian matrix of a graph $G$ (with $n$ vertices). We would like to construct a sparse graph $G'$ from $G$ with Laplacian $L'$, such that

$$
(1 - \varepsilon)x^T L'x \leq x^T L x \leq (1 + \varepsilon)x^T L'x \quad \forall x \in \mathbb{R}^n
$$

Let $L$ have eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. According to the Courant Fischer theorem,

$$
\lambda_l = \max_{S \text{ of dimension } k} \min_{x \in S} \frac{x^T L x}{x^T x}
$$

Therefore, if we can sparsify $G$ as described above to construct $G'$, $L'$ will have approximately the same eigenvalues as $L$. This property makes such a sparsified graph useful in solving diagonally-dominant linear systems quickly [2].

There are algorithms that exist which sparsify graphs as above. Spielman and Teng wrote an algorithm to sparsify graphs to $\frac{n \log^{O(1)} n}{\varepsilon^2}$ edges in time $O(m \log^{O(1)} m)$. Nikhil Srivastava wrote an algorithm to sparsify graphs to $n \log(n) / \varepsilon^2$ edges in time $O(m \log^{O(1)} n)$ using effective
resistances to sample edges of the original graph. [2, 3] However, the constants in the running times of these algorithms make it difficult to solve linear systems quickly.

In my project, I will consider the Benczúr-Karger algorithm, which uses non-uniform random sampling to sparsify graphs. The Benczúr-Karger algorithm sparsifies graphs to \(\frac{n \log(n)}{\varepsilon^2}\) edges that maintain approximately the same cut values in time \(O(m \log^3 n)\). Note that Benczúr-Karger sparsifications are guaranteed only to maintain cut values, which is not the notion of sparsification we are looking for. In particular, they are not useful in solving linear systems quickly. However, the values of cuts are closely related to the quantity \(x^T L x\) (the quantity we would like to preserve). Consider a cut \(c(S, \overline{S})\), and the characteristic vector \(x\):

\[
x(i) = \begin{cases} 
1 & \text{if } v_i \in S \\
0 & \text{if } v_i \in \overline{S}
\end{cases}
\]

We have:

\[
x^T L x = \sum_{e=(v_i, v_j) \in E} \omega(e)(x(i) - x(j))^2 = \sum_{e=(v_i, v_j) \in E, v_i \in S, v_j \in \overline{S}} \omega(e) = c(S, \overline{S}).
\]

Since each characteristic vector \(x\) represents a subset \(S \subseteq V\) (and therefore a cut in \(G\)), the Benczúr-Karger algorithm approximately preserves the quantity \(x^T L x\) for all \(x \in \{0, 1\}^n\). However, there are cases where it fails to preserve \(x^T L x\) for vectors \(x\) whose components are not in \(\{0, 1\}^n\).

The first part of my project will entail implementing the Benczúr-Karger algorithm. I will then run the algorithm on several graphs, including known examples where the algorithm does not sparsify the graph using the sparsification notion we are looking for. I will use the resulting data to attempt to understand how and why the algorithm fails. I will attempt to patch the algorithm so it approximately maintains \(x^T L x\) for all \(x \in \mathbb{R}^n\) (empirically or possibly theoretically). If such an improvement is possible (without increasing the running time too much), it will result in an algorithm that sparsifies graphs to \(\frac{n \log(n)}{\varepsilon^2}\) edges in time \(O(m \log^3 n)\). This would be an improvement over [2] and [3].

