Reflection

In studying network-based algorithms, I have gained deeper insight into the fundamental principles that connect graph theory with optimization problems. The exploration began with the Minimum Cut problem and Karger's Algorithm. Karger's randomized approach to finding a global minimum cut provided an interesting perspective on how probabilistic methods can achieve efficient solutions in complex graph problems. Its simplicity contrasts with deterministic algorithms, yet it highlights the power of randomization in algorithm design.

The subsequent analysis of Karger's Algorithm emphasized the trade-off between probability of success and the number of iterations. This reinforced the importance of understanding both theoretical guarantees and practical considerations when applying randomized algorithms.

Moving forward, the Min-cut Max-flow Theorem established a profound theoretical connection between two problems that, at first, seem distinct: finding the maximum flow in a network and determining the minimum cut. This duality not only provides a foundation for efficient algorithms but also illustrates the beauty of theoretical elegance in computer science.

Finally, the Ford-Fulkerson Algorithm demonstrated how these concepts can be applied to compute maximum flow. Through augmenting paths and residual networks, it offers both practical computation and an illustration of the Min-cut Max-flow Theorem in action. The algorithm's iterative nature deepened my appreciation for how theoretical results translate into concrete procedures.

Overall, this study has strengthened my understanding of how network-based algorithms serve as both practical tools and theoretical milestones.

Task 2: Weighted Minimum Cut - Extending Karger's Algorithm

Karger's algorithm is a randomized algorithm for finding the minimum cut of an unweighted graph by contracting edges. In this task, we consider the scenario where edges have weights. Your Task is to design a modified version of Karger's algorithm (or a new algorithm altogether) that accounts for edge weights.

- You are not required to write code.
- Present your proposed algorithm in one of the following formats: Pseudocode, a written paragraph description, or short video explanation (e.g., via Panopto).

Instructions:

- Think about how weights should affect the probability of edge contraction.
- Make any reasonable assumptions that help simplify your algorithm design.
- Discuss how your approach still retains the spirit of randomized contraction or improves upon it for weighted cases.

A :

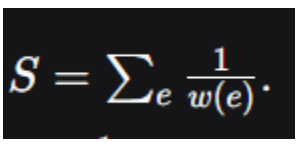
Karger's algorithm is a randomized algorithm that finds the minimum cut in an unweighted graph by repeatedly contracting random edges until only two vertices remain. The key insight is that edges with higher weights are less likely to be part of the minimum cut, so we should avoid contracting them early on.

Proposed Algorithm Description

To handle weighted graphs, we adjust the probability of selecting an edge for contraction based on its weight. Specifically, the likelihood of selecting an edge is inversely proportional to its weight. This means that edges with smaller weights have a higher chance of being contracted, while edges with larger weights are preserved longer, increasing the likelihood of finding the minimum cut.

The algorithm proceeds as follows:

1. **Initialization**: Start with the original weighted graph.
2. **Contraction Process**: While there are more than two vertices:
    a. Calculate the sum of the reciprocals of all edge weights: (S = \sum_{e} \frac{1}{w(e)}).



   b. For each edge (e), compute its selection probability: (p(e) = \frac{1}{w(e) \cdot S}).
      $$p(e) = \frac{1}{w(e) \cdot S}.$$
   c. Randomly select an edge (e) according to this probability distribution.
   d. Contract the edge (e): merge the two vertices connected by (e) into a single vertex, combining any multiple edges that arise by summing their weights.
3. **Termination**: When only two vertices remain, the cut is the set of edges between them, and the cut value is the sum of their weights.
4. **Repetition**: Since the algorithm is randomized, we run it multiple times (e.g., (O(n^2 \log n)) times) and return the minimum cut value found.

Assumptions

- The graph is connected and undirected. If the graph is disconnected, the algorithm may need to be adapted, but for simplicity, we assume connectivity.
- All edge weights are positive. Negative weights would require different handling, but in minimum cut problems, weights are typically non-negative.
- The graph has no self-loops, which can be removed during preprocessing.

Conclusion

This approach retains the core idea of Karger's algorithm—random contraction—but improves it for weighted graphs by biasing the random selection towards low-weight edges. This bias helps preserve high-weight edges, which are critical for the minimum cut. The randomized nature ensures that the algorithm remains simple and efficient, with a high probability of finding the minimum cut after multiple iterations.

Improvement for Weighted Cases

By incorporating weights into the probability distribution, the algorithm effectively reduces the chance of contracting edges that might be part of the minimum cut, leading to better performance in weighted graphs compared to a uniform random selection.

Resource ：

1. https://zhuanlan.zhihu.com/p/374959975 ; 2023-03-03 17:39;
2. https://juejin.cn/post/7117822087628554277 ; 2022-07-08 ;
3. https://hackmd.io/@ShanC/maxflow-mincut-theorem ; Aug 29 2025 ; Shan C
4. https://alrightchiu.github.io/SecondRound/flow-networksmaximum-flow-ford-fulkerson-algorithm.html ; Posted by Chiu CC on 3 20, 2016 ; Flow Networks：Maximum Flow & Ford-Fulkerson Algorithm
5. https://smilecatx3.blogspot.com/2014/06/the-ford-fulkerson-method.html ; Xavier Lin @NCKU ; The Ford-Fulkerson Method ; 2014-06-15