Background

December 2023

1 Weight of an MSF

Consider a graph G with n nodes where each edge has a weight between 1 and W. Let G_i, where $i \in \{1, .., W\}$, be the graph formed by removing all edges of weight more than i from G. Let c_i be the number of components of G_i. Note that c_0 is equal to n.

Now consider the difference between G_i and G_{i+1}. We have that G_{i+1} is equal to G_i plus all the edges of weight $i+1$ in G. Hence, if G_{i+1} has fewer components than G_i, there must be edges of weight $i+1$ connecting components of G_i. More precisely, for each less component, there must be a unique combination of two distinct components in G_i that are connected by an edge of weight $i+1$ in G_{i+1}, and thereby also in G. Note that in G, there can multiple edges of weight $i+1$ or more that connect two components of G_i. But in an MSF of G, only one of these edges would be included, more precisely the one with the lowest weight. Hence, we know that for each less component G_{i+1} has compared to G_i, there is precisely one edge of weight $i+1$ that will be included in the MSF. Therefore, in summary, if we let e_i be the number of edges of weight i in an MSF of G, we have $e_i = c_{i-1} - c_i$ for every $i \in \{1, .., W\}$.

Based on this, the total weight of an MSF in G must be equal to $\sum_{i=1}^{W} i \cdot (c_{i-1} - c_i)$. This sum $1 \cdot (c_0 - c_1) + 2 \cdot (c_1 - c_2) + ... + W \cdot (c_{W-1} - c_W)$ can be simplified to $c_0 + (\sum_{i=1}^{W-1} c_i) - W c_W$ or equivalently $c_0 - (W+1) \cdot c_W + \sum_{i=1}^{W} c_i$. As $c_0 = n$, we have that the final formula for the weight of an MSF is $n - (W+1) \cdot c_W + \sum_{i=1}^{W} c_i$.

This formula can also be derived from the formula of an MST by Czumaj and Sohler [1]: $n - W + \sum_{i=1}^{W-1} c_i$. To show this, consider a disconnected graph G with maximum edge weight W. We must add $c_W - 1$ edges of weight $W + 1$ to G to get a connected graph G'. Using the formula of Czumaj and Sohler [1], G' would have an MST of total weight $n - (W+1) + \sum_{i=1}^{W} c_i$. An MSF of the original graph G would then be equal to this MST's weight minus the total weight of the $c_W - 1$ number of added edges of weight $W + 1$. Hence, an MSF of G would have weight $n - (W+1) + (\sum_{i=1}^{W} c_i) - (c_W - 1) \cdot (W+1) = n - (W+1) \cdot c_W + \sum_{i=1}^{W} c_i$.

2 Estimation of MSF Weight

Given the formula $n - (W+1) \cdot c_W + \sum_{i=1}^{W} c_i$, we can estimate the weight of an MSF by estimating c_i for each $i \in \{1, ..., W\}$. This can be done by using an algorithm proposed by Czumaj and Sohler [1]. In this algorithm, we calculate each estimate c_i^* as follows:

- Sample s nodes uniformly from the n total nodes.
- For each sampled node $u_j \in \{u_1, ..., u_s\}$, do:
  - Randomly choose X such that $P(X \geq k) = \frac{1}{k}$.
  - Do a BFS in G_i starting at u and break if 1) the entire component of u has been visited, or 2) more than X vertices have been visited.
  - Set b_j to 1 if the first case stopped the BFS, otherwise 0.
- Set $c_i^* = \frac{n}{s} \sum_{j=i}^{s} b_j$

Czumaj and Sohler [1] show that this algorithm will produce an estimate c_i^* such that $\mathbf{E}(c_i^*) = c_i$ and also such that c_i^* is concentrated around c_i. The only problem with this algorithm is that it requires us to pick a specific s. In the MSF project, Kattis only gives the algorithm the number of total nodes n, the maximum edge-weight W, and the maximum number of requests we can do without getting a score-penalty (where each request gives us the neighbors and corresponding edge-weights of the node the request concerns). None of this easily translates to what the maximum s could be.

Therefore, we made a modification to the above algorithm such that instead of sampling s nodes for each c_i, we pick just one node and go through all the c_i^*, and then repeat with another random node and so on until we don't have requests left to make. In other words, one could say that we first calculate each c_i^* with $s = 1$, then with $s = 2$, and so on. If we run out of requests in the middle of an iteration for a random node, we use the last s we were able to calculate all c_i^* for. In summary, by doing all of this, we maximize s while never making more requests than allowed.

The actual algorithm we implemented does not update c_i^* for each random node. Instead, it keeps track of the sum of all the b_j's over all c_i^*, $i \in \{1, .., W\}$, as well as the sum of all b_j's from the c_W^* iterations specifically. When we run out of requests, we simply calculate $\sum_{i=1}^{W} c_i^*$ and c_W^* by multiplying the sums with $\frac{n}{s}$ respectively. This works as n and s will be the same for all c_i^*.

Lastly, the distribution of the X in the algorithm from Czumaj and Sohler [1] was chosen as $\frac{1}{Y}$ where Y is a random variable from the continuous uniform distribution from 0 to 1. This fulfils the condition $P(X \geq k) = \frac{1}{k}$ stipulated by Czumaj and Sohler [1] which can be shown as follows: $P(X \geq k) = 1 - P(X \leq k) = 1 - P(\frac{1}{Y} \leq k) = 1 - P(\frac{1}{k} \leq Y) = 1 - (1 - P(\frac{1}{k} \geq Y)) = P(Y \leq \frac{1}{k}) = [Definition\ of\ CDF\ of\ continous\ uniform\ distribution\ from\ 0\ to\ 1] = \frac{1}{k}$.

References

[1] A. Czumaj and C. Sohler, "Sublinear-time algorithms," in *Property Testing: Current Research and Surveys*. Springer, 2010, p. 50. DOI: 10.1007/978-3-642-16367-8_5.