CS 761: Randomized Algorithms Fall 2019

Lecture 2 — September 13, 2019

Prof. Gautam Kamath *By: Graeme Stroud, Kevin Wang, Kevin Wu*
Edited by Vedat Levi Alev

Disclaimer: These notes have not been subject to the usual scrutiny reserved for formal publications.

Recall from last lecture:

- (Markov's Inequality) For a non-negative random variable (R.V.) $X > 0$, we have that $\Pr(X > a) \leq \frac{E[X]}{a}$.
- (Coupon collector) Let $X_i \sim Uniform([n])$. How many trials T until all items of $[n]$ are seen? Using direct methods, we proved the bound $\Pr[T \geq \beta n \ln n] \leq n^{-(\beta-1)}$. We also showed that $E[T] = nH_n \approx n \ln n$, which implies, by Markov's inequality, the bound $\Pr[T \geq \beta n \ln n] \leq \frac{1}{\beta}$.
- The variance of a random variable X is $\mathrm{Var}[X] := E[(X - E[X])^2] = E[X^2] - E[X]^2$.

Remark 1. *For a random variable X, we will denote $E[X] = \mu_X$ and $\mathrm{Var}[X] = \sigma_X^2$. We will drop the subscripts when the context is clear.*

1 Chebyshev's Inequality

In the last lecture, we proved Markov's inequality, which shows that the probability that a non-negative random variable is less than a constant is bounded above by its expected value. Let's look at proving Chebyshev's inequality, which shows that a distribution with bounded variance will be concentrated around its mean.

Theorem 2 (Chebyshev's Inequality)**.** *Let X be a random variable with $\mathrm{Var}[X] < \infty$. Then*

$$\Pr(|X - \mu| \geq k) \leq \frac{\mathrm{Var}[X]}{k^2}.$$

Equivalently, denoting $\sigma = \sqrt{\mathrm{Var}[X]}$

$$\Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}.$$

Proof. We look at the non-negative random variable $Y = (X-\mu)^2$. Then $E[Y] = E[(X-\mu)^2] = \sigma^2$. By Markov's inequality, we have $\Pr\left(Y \geq k^2 E[Y]\right) \leq \frac{E[Y]}{k^2 E[Y]} = \frac{1}{k^2}$. Hence

$$\Pr\left(|X - \mu| \geq k\sigma\right) = \Pr\left((X-\mu)^2 \geq k^2\sigma^2\right) = \Pr\left(Y \geq k^2 E[Y]\right) \leq \frac{1}{k^2}.$$

□

Example 1 (Application of Chebyshev's Inequality to Coupon Collector)**.**

Recall that $T = \sum_{i=1}^{n} t_i$ *where* t_i *is the time between the* $(i-1)$*-th and* i*-th unique coupon. Recall that each* t_i *are distributed as a geometric random variable:* $t_i \sim Geometric(\frac{n-i-1}{n})$*.*

First we will show the variance of $X \sim Geometric(p)$ *is* $\mathrm{Var}[X] = (1-p)/p^2 \leq \frac{1}{p^2}$*.*

Proof. Let $q = 1-p$. We have $\mathrm{Var}[X] = E[X^2] - E[X]^2 = \sum\limits_{i=1}^{\infty} i^2 q^{i-1} p - (\frac{1}{p})^2$.

We will evaluate $\sum\limits_{i=1}^{\infty} i^2 q^{i-1} p$ by differentiation.

$$\begin{aligned}
\sum_{i=1}^{\infty} i^2 q^{i-1} p &= p \frac{\mathsf{d}}{\mathsf{d}q} \sum_{i=1}^{\infty} i q^i \\
&= p \frac{\mathsf{d}}{\mathsf{d}q} q \frac{\mathsf{d}}{\mathsf{d}q} \sum_{i=1}^{\infty} q^i \\
&= p \frac{\mathsf{d}}{\mathsf{d}q} q \frac{\mathsf{d}}{\mathsf{d}q} \frac{q}{1-q} \\
&= p \frac{\mathsf{d}}{\mathsf{d}q} \frac{q}{(1-q)^2} \\
&= p \frac{1+q}{(1-q)^3} \\
&= \frac{2-p}{p^2}
\end{aligned}$$

Thus $\mathrm{Var}[X] = \frac{2-p}{p^2} - \frac{1}{p^2} = \frac{1-p}{p^2}$. □

By this claim we have $\mathrm{Var}[t_i] \leq \frac{n^2}{(n-i+1)^2}$*. Since the* t_i *are independent,*

$$\mathrm{Var}[T] = \mathrm{Var}\left[\sum_i t_i\right] \underbrace{=}_{\perp} \sum_i \mathrm{Var}[t_i] = n^2 \sum_i \frac{1}{i^2} \leq n^2\pi^2/6.$$

By Chebyshev's inequality,

$$\Pr[|T - \mathbb{E}[T]| \geq (\beta - 1) n \ln n] \leq \frac{n^2\pi^2/6}{\beta^2 n^2 \ln^2 n} = O\left(\frac{1}{\beta^2 \ln^2 n}\right).$$

Since $\mathbb{E}[T] = nH_n \approx n \ln n$*, this shows*

$$\Pr(T \geq \beta n \ln n) \leq O\left(\frac{1}{\beta^2 \ln^2 n}\right).$$

Example 2 (Mean Estimation)**.** *Consider some distribution* D *with finite variance. We would like to estimate the mean of* D *using samples from* D*. We can use Chebyshev's inequality to show that the arithmetic mean of the samples is close to the distribution's mean with high probability.*

Let $X_1, \dots, X_n \sim D$ be independent samples and assume that $\mathrm{Var}[D] \le \sigma^2$. *Our estimate $\hat{\mu}$ will be such that $|\hat{\mu} - \mu|$ is small with high probability. Set $\hat{\mu} = \frac{1}{n}\sum_i X_i$. Then*

$$E[\hat{\mu}] = E\left[\frac{1}{n}\sum_i X_i\right] = \frac{1}{n}\sum_i E[X_i] = \mu.$$

Next, it is an easy exercise to show that $\mathrm{Var}[kX] = k^2\mathrm{Var}[X]$ *for a random variable X and $k \in \mathbb{R}$.*

Using the exercise,

$$\mathrm{Var}[\hat{\mu}] = \mathrm{Var}\left[\frac{1}{n}\sum_i X_i\right] = \frac{1}{n^2}\sum_i \mathrm{Var}[X_i] = \frac{\sigma^2}{n}.$$

By Chebyshev's inequality,

$$\Pr\left(|\hat{\mu} - \mu| \ge \frac{10\sigma}{\sqrt{n}}\right) \le \frac{\sigma^2/n}{(10\sigma/\sqrt{n})^2} = \frac{1}{100}.$$

If we set $n = O(\frac{\sigma^2}{\epsilon^2})$, then $\Pr(|\hat{\mu} - \mu| \ge \epsilon) \le \frac{1}{100}$.

2 Chernoff's Method

Chebyshev's inequality uses the second moment ($E[X^2]$). We could have proved something for the higher moments by observing $\Pr[|X-\mu| \ge a] = \Pr[|X-\mu|^k \ge a^k] \le \frac{E[|X-\mu|^k]}{a^k}$. This suggests that we may be able to get better bounds by considering more moments of the variable. The Moment Generating function (MGF) of a random variable will help determine these moments. The MGF will also gives us another way to measure how concentrated a random variable is near it's mean, by using Chernoff's method.

2.1 Moment Generating Functions (MGF)

Definition 3. *The MGF of a random variable X is $M_X(t) = E[e^{tX}]$. Expanding out the Taylor series for e^x, we have*

$$E[e^{tX}] = E[\sum_{i\ge 0}\frac{t^i}{i!}X^i] = \sum_{i \ge 0}\frac{t^i}{i!}E[X^i]..$$

Exercise 1. *Show that*

- $M_X^{(k)}(0) = E[X^k]$, *where $f^{(k)}$ is the kth derivative of f.*
- *If $X = X_1 + X_2$, then $M_X(t) = M_{X_1}(t) \cdot M_{X_2}(t)$.*

2.2 Chernoff bound for Bernoulli Variables

The general form of the Chernoff method for a random variable X is

$$\Pr(X \ge a) = \Pr(e^{tX} \ge e^{ta}) \le \frac{E[e^{tX}]}{e^{ta}} = \frac{M_X(t)}{e^{ta}} \quad \forall t \ge 0$$

We will consider a simple case of our random variable being a sum of independent and identically distributed Bernoulli random variables. Let $X_i \sim Bernoulli(p_i)$ be independent, and $X = \sum_i X_i$. Then $\mu_X = \sum_i \mu_i = \sum_i p_i$.

The MGF of X can be computed by:

$$\begin{aligned} M_X(t) &= E[e^{tX}] \\ &= \prod_i E[e^{tX_i}] \\ &= \prod_i (p_i e^{t\cdot 1} + (1-p_i)e^{t\cdot 0}) \\ &= \prod_i (1 + p_i(e^t - 1)) \\ &= \prod_i e^{p_i(e^t-1)} && \text{Since } 1 + x \leq e^x \\ &= e^{\sum_i p_i(e^t-1)} \\ &= e^{\mu(e^t-1)} \end{aligned}$$

If we take $a = (1+\delta)\mu$ and $t = \ln(1+\delta)$, then we have, using the general form of the Chernoff method

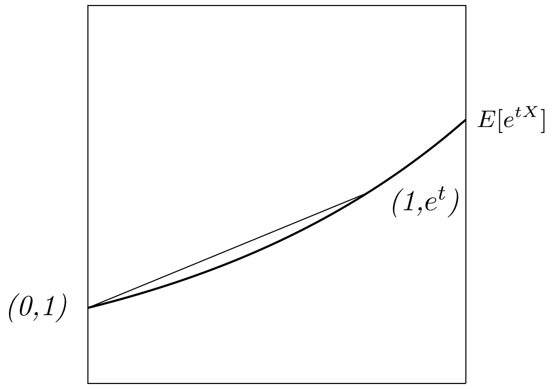
$$\Pr(X \geq (1+\delta)\mu) \leq \frac{e^{\mu\delta}}{(1+\delta)^{(1+\delta)\mu}}$$

Exercise 2. *Show that for* $0 \leq \delta \leq 1$, $\frac{e^\delta}{(1+\delta)^{(1+\delta)}} \leq e^{-\delta^2/3}$.

Using this exercise, we can deduce that $\Pr(X \geq (1+\delta)\mu) \leq e^{-\mu\delta^2/3}$ for $0 \leq \delta \leq 1$. Similarly, we may show that $\Pr(X \geq (1-\delta)\mu) \leq e^{-\mu\delta^2/2}$.

Theorem 4 (A common Chernoff bound)**.** $\Pr(|X - \mu| \geq \delta\mu) \leq 2e^{-\mu\delta^2/3}$, *for* $0 \leq \delta \leq 1$

Remark 5. *This can be slightly generalized to continuous random variables* X_i *supported on* $[0,1]$ *with* $E[X_i] = p_i \in [0,1]$.



This graph illustrates that $e^{tx} \leq (e^t - 1)x + 1$ *when* $x \in [0,1]$

Thus $E[e^{tX}] \leq E[(e^t-1)X+1] = (e^t-1)E[X]+1$. Therefore, the entire derivation from above works for this random variable.

Example 3 (Comparison of Chernoff's bound with Chebyshev's Inequality.)**.**
Let $X \sim Bin(n, 1/2) = \sum_i Bernoulli(1/2)$. We have $\mathrm{Var}[X] = \sum \mathrm{Var}[Bernoulli(1/2)] = n \cdot \frac{1}{4} = \frac{n}{4}$ and $E[X] = \sum E[Bernoulli(1/2)] = \frac{n}{2}$. By Chebyshev's inequality

$$\Pr(|X - n/2| \geq k\sqrt{n}) \leq \frac{n/4}{(k\sqrt{n})^2} = \frac{1}{4k^2}.$$

By Chernoff (Theorem 4), we have

$$\Pr\left(|X - n/2| \geq \frac{2k}{\sqrt{n}}\frac{n}{2}\right) \leq 2\exp\left(\frac{-4k^2}{n} \cdot n/2 \cdot 1/3\right) = 2\exp(-2k^2/3).$$

To obtain a bound of $O(\delta)$, we need to set $k = \sqrt{1/\delta}$ for Chebyshev's, while we only need $k = \sqrt{\ln(1/\delta)}$ for Chernoff.

2.3 Bounds for Coupon Collector

In lecture 1, we gave two bounds for the Coupon Collector problem using a direct bound and Markov's inequality. Chebyshev's inequality (Example 1) and Chernoff's method can be used to get a bound too. The quantity we are bounding for coupon collector is $\Pr(T \geq \beta n \ln n)$.

- Direct bound gives $\leq n^{-(\beta-1)}$.
- Markov's inequality gives $\leq O(\frac{1}{\beta})$.
- Chebyshev's inequality gives $O(\frac{1}{\beta^2 \ln^2 n})$.
- Chernoff's method [1] gives $\leq n^{-(\beta-1-\ln\beta)}$.

3 Graph Algorithms

We will explore algorithms for two well known graph problems: the Min-Cut problem and the Minimum Spanning Tree problem.

3.1 Karger's Min-Cut Algorithm

Given a graph, consider splitting the vertices into two non-empty disjoint sets. Some edges may have its endpoints be in opposite sets. If we "cut" all of these edges, we end up separating the vertices of the two sets from each other. The main goal will be to find a partitioning that minimizes the number of cut edges.

Definition 6. *Let $G = (V, E)$ be a graph. We will denote $|V| = n$ and $|E| = m$. A cut of G is a partition $(S, T = V \setminus S)$ of V. A cut (S, T) is a min-cut if the number of cut edges $\{(u, v) \in E; u \in S, v \in T\}$ is minimum over all cuts of G. For two vertices $s, t \in V$, an st-cut is a cut (S, T) such that $s \in S$ and $t \in T$.*

Remark 7. *We can compute the minimum cut for a graph (V, E), by computing the minimum st-cut for all distinct pairs of vertices $s, t \in V$, and taking the st-cut that has the least number of edges.*

As a baseline, we can compare with the complexity of algorithms based on max flow.

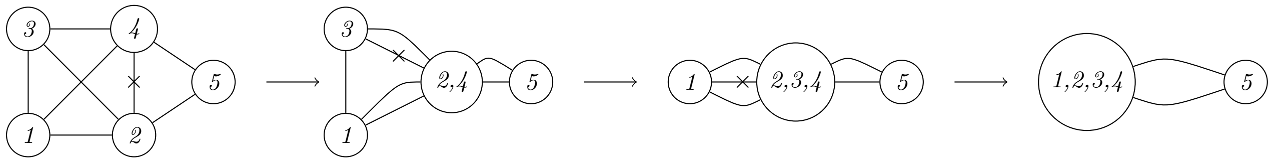
Remark 8. *The value of the min st-cut is equal to the maximum amount of flow that can pass from source s to sink t in G, where each edge has a capacity of 1. If a st-flow that is maximum has been computed, a minimum st-cut can be computed from it in linear time. The best known algorithm for computing the max st-flow is by Orlin [2] which has run time of $O(mn)$. Applying this naively over all pair of vertices st gives a run time of $O(mn^3)$. This can be optimized so that s is fixed and we iterate over all t, giving a run time of $O(mn^2)$.*

Karger's Min-Cut algorithm provides a much more efficient algorithm than using an st-flow algorithm. The algorithm has a low probability of success, but can be called repeatedly in order to boost the probability of success. The pseudocode is presented in Algorithm 1. The main idea: repeatedly pick a random edge from the graph and "contract" the edge, by merging the two endpoints. The subsequent graph may be a multi-edge graph, and if we ever end up with a self loop, we remove it. Once only two vertices are left, we return the cut given by the edges that remain.

```
Input: $G = (V, E)$, undirected
Output: a partition $(S, T)$ of $V$ such that the number of cut edges is minimized
while $|V| > 2$ do
    choose $e \in E$ uniformly
    contract $e$
return the remaining cut
```

Algorithm 1: Karger's Min-Cut Algorithm

Example 4. *An example execution of the algorithm:*



Observation 9. *At any intermediate step, a cut in the intermediate graph is a cut in the original graph.*

Lemma 10. *Karger's Min Cut algorithm outputs a min-cut with probability $\geq \frac{2}{n(n-1)}$.*

Proof. Let $G_1 := G, G_2, \ldots, G_{n-1}$ be the graphs we obtain as we run Karger's algorithm. At step i, we contract an edge in G_i to produce the graph G_{i+1}. Fix a min-cut F of size k. We will show that

$$\Pr(\textit{some } e \in F \text{ is contracted at step i}) = \frac{\textit{\# edges in } F}{\textit{\# edges in } G_i} \leq \frac{2}{n - i + 1}$$

We condition this probability on never contracting an edge in F so far. Observe that for every i and vertex $v \in G_i$, $\deg v \geq k$. Otherwise, a cut containing only v will have smaller size than F.

Furthermore, since we contract one edge at each step, the size of G_i is $n - i + 1$. Thus the number of edges in G_i is at least $(n - i + 1) \cdot k/2$.
Thus $\Pr(\textit{some } e \in F \text{ is contracted at step i}) \leq \frac{2}{n-i+1}$ and $\Pr(\text{contract an edge not in } F) \geq \frac{n-i-1}{n-i+1}$.
Taking this over all contractions we have

$$\begin{aligned}
\Pr(\text{never contract an edge in } F) &= \prod_{i=1}^{n-2} \Pr(\text{contract an edge not in } F \text{ at step } i) && (1)\\
&\geq \prod_{i=1}^{n-2} \frac{n-i-1}{n-i+1} = \prod_{i=1}^{n-2} i \prod_{i=3}^{n} \frac{1}{i} && (2)\\
&= \frac{2}{n(n-1)} && (3)
\end{aligned}$$

□

Theorem 11. *If we run the algorithm $O(n^2)$ times and take the minimum cut produced, we will find the global min-cut with probability $> 99\%$.*

Proof. We upper bound the probability that, after T independent runs of the algorithm, none of them produce a min-cut. $\Pr(T \text{ runs fail}) \leq (1 - \frac{2}{n(n-1)})^T \leq \exp(-2T/(n(n-1)))$. Taking $T = cn^2$, the failure probability is $\leq e^{-2c}$. □

Remark 12. *Each run of Karger's algorithm can be done in $O(m)$ time, so the total running time is $O(mn^2)$. This can be improved to $\tilde{O}(mn)$ [3]. The key idea is that in the first few steps, the algorithm is unlikely to fail, so these steps can be reused.*

Interestingly, while this statement is algorithmic, it also implies structural properties about the number of min-cuts in a graph.

Corollary 13. *For any G, there are $O(n^2)$ min cuts of G.*

Proof. For any min cut F, it is output with probability $\geq \frac{2}{n(n-1)}$. These events are disjoint. Thus

$$1 \geq \sum_{\text{min cut } F} \Pr(F \text{ survives}) \geq \# \text{ min cuts} \cdot \Omega(n^2).$$

□

3.2 Minimum Spanning Tree

A minimum spanning tree (MST) of a graph with edge weights asks for a subgraph with the same vertices and whose edges form a tree of minimum total weight. The Karger-Klein-Tarjan MST algorithm is an efficient randomized algorithm for this problem. Unlike Karger's Min-Cut algorithm, this MST algorithm always returns the correct answer, but the runtime is random. In expectation, the runtime is linear in the number of edges of the graph.

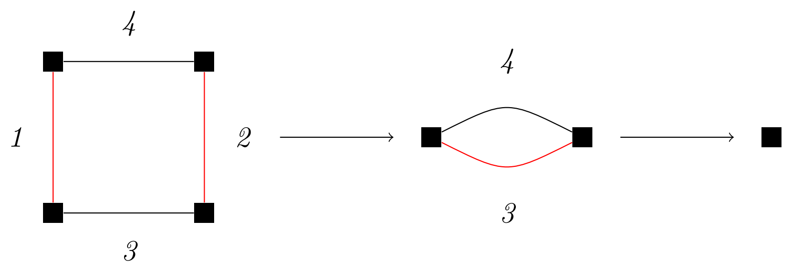
Definition 14. *Let G be an undirected graph. Suppose that we have weights w_{ij} for every edge $(i, j) \in E$ and assume that these weights are unique. A spanning forest is a subset $S \subseteq E$ such that $G' = (V, S)$ contains no cycles and S is the maximal set with this property. Equivalently, if u, v is connected in G, it is connected in G'. If $|S| = n - 1$, then we say that G' is a spanning tree. A minimum spanning (forest,tree) (MSF, MST) is a spanning (forest,tree) of minimum weight $(\sum_{e \in S} w_e)$.*

Fact 15. *The best deterministic algorithm currently is $O(m\alpha(m, n))$ where α is the inverse Ackermann function. [4]*

Recall Kruskal's algorithm for MST: First sort edges by increasing weight. Initialize a set that collects the edges of the minimum spanning tree. At the beginning, this set is empty. Going through the edges in sorted order, check if adding it to the currently chosen edges would create a cycle with the edges added so far, and add it if it does not. Return the tree gotten after adding $n - 1$ edges.

Another minimum spanning tree algorithm is Boruvka's Algorithm: for every node, pick an incident edge of minimum weight, and add it to the MST edge set. Contract all these edges in the graph. Then recursively repeat on the new graph, accumulating the edges until we get a one node graph. This means we did $n - 1$ contractions, and each edge we contracted are the edges of the minimum spanning tree.

Example 5 (Execution of Boruvka's algorithm)**.**



The red edges get contracted, and will be the edges of the minimum spanning tree.

Remark 16. *We contract at least $|V|/2$ edges at each step of Boruvka's algorithm. Therefore, it takes $O(\log n)$ steps in order to contract $n - 1$ edges. Each step takes $O(m)$ time, so the total run time is $O(m \log n)$.*

Boruvka's algorithm is as efficient asymptotically as Kruskal's. We will see how Boruvka's algorithm is used in the Karger-Klein-Tarjan algorithm. The key observation that motivates this algorithm is that there are certain edges of G that are not part of any MST of G. So we can remove such edges from the graph, and this subgraph will have the same MSF's as G. We will be able to compute a subset of these edges efficiently.

Definition 17. *Given a forest F, an edge $e = (u, v)$ is F-*heavy *if*

- *u, v is connected in F*

- *$w_{uv} \geq$ the weight all of edges on the u, v path in F.*

*If e is not F-heavy, we say that it is F-*light*.*

Exercise 3. *Suppose that $e = (u, v)$ is F-heavy for some F. Equivalently, suppose that e is the heaviest edge of a cycle C. Then no MSF (and therefore no MST) of G will contain (u, v).*

Hint/sketch: Suppose that e is in a MST. Replace it by the other edges in the cycle. Resolve any cycles this creates and argue that the resulting tree has smaller weight. □

Theorem 18. *There is an algorithm taking G and a forest F, and computes all F-heavy edges in $O(m + n)$ time. [5]*

The idea behind the algorithm is that we will sample a small subgraph of G, and compute the MSF F of this subgraph. Then, remove all F-heavy edges and hope that the resulting graph is sparse, and directly compute the MST of this.

```
Algorithm: Naive MST
Generate $G(p)$ by sampling each edge with probability $p$
Find MSF of $G(p)$, call it $F_p$
Remove all $F_p$-heavy edges from $G$ to form $G'$.
Find MST of $G'$.
```

Lemma 19. *$E[\#$ of $F_p -$ light edges in $G] \leq n/p$.*

Proof. We will follow Kruskal algorithm with a twist. We will sample the edges in order of the weight. If it is selected, add it to F if it does not form a cycle among the selected edges so far. Note that since each sample is independent, this is equivalent to constructing $G(p)$ normally. Furthermore, since the order of the edges is the same as in Kruskal's algorithm, the forest we obtain is a MSF of $G(p)$. Let E_i be the state of F after we sample the i-th edge, $e_i = (u, v)$. Then e_i is F-light $\Leftrightarrow$ e_i connects two connected components in E_{i-1}. This is because e_i is F-light when u, v is not connected in F or some edge of F is heavier on the u, v path. In the first case, the two components were never connected. In the second, it was connected by an edge after e_i in the order, and thus must have higher weight. Since each F-light edge is selected with probability p, the number of F-light edges considered between the j-th and $j + 1$th edge added to F is $1/p$ (this is a geometric random variable). Since we add $n - 1$ F-light edges to F, so the expected number of F-light edges is $\sum_{i=1}^{n-1} \frac{1}{p} = \frac{n-1}{p} \leq \frac{n}{p}$. □

The cost of the second line of Naive MST is $mp = |G(p)|$. The cost of the fourth line is $\frac{n}{p}$. Equating these we get that $p = \sqrt{\frac{n}{m}}$. Using this, both of the calls to find the MST of the smaller graphs would take $O(\sqrt{nm} \log n)$ time. The total time would be $O(m + \sqrt{nm} \log n)$. This is linear if $m \geq n \log^2 n$. We would like to have a linear time algorithm in general. To do this, we will compute the MST of these constructed graphs recursively. Thus we obtain:

```
Algorithm: Linear MST (Karger-Klein-Tarjan's algorithm)
Run 3 iterations of Boruvka's algorithm
Set $p = 1/2$
Generate $G(p)$ by sampling each edge with probability $p$
Use Linear MST on $G(p)$ to obtain MSF $F_p$
Remove all $F_p$-heavy edges from $G$ to form $G'$.
Use Linear MST to find MST of $G'$.
```

The run time of this algorithm has the recurrence $T(m, n) = c(m+n)+T(m/2, n/8)+T(n/4, n/8)$. It can be verified that $T(m, n) \leq 2c(m + n) = O(m + n)$ is a solution to this recurrence.

References

[1] S. Janson, "Tail bounds for sums of geometric and exponential variables," *Statistics & Probability Letters*, vol. 135, pp. 1 – 6, 2018.

[2] J. B. Orlin, "Max flows in o(nm) time, or better," in *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pp. 765–774, 2013.

[3] D. R. Karger and C. Stein, "A new approach to the minimum cut problem," *J. ACM*, vol. 43, no. 4, pp. 601–640, 1996.

[4] B. Chazelle, "A minimum spanning tree algorithm with inverse-ackermann type complexity," *J. ACM*, vol. 47, no. 6, pp. 1028–1047, 2000.

[5] B. Dixon, M. Rauch, and R. E. Tarjan, "Verification and sensitivity analysis of minimum spanning trees in linear time," *SIAM Journal on Computing*, vol. 21, no. 6, pp. 1184–1192, 1992.