An Intimate Analysis of Cuckoo Hashing with a Stash

Daniel Noble *

May 18, 2021

Abstract

Cuckoo Hashing is a dictionary data structure in which a data item is stored in a small constant number of possible locations. It has the appealing property that the data structure size is a small constant times larger than the combined size of all inserted data elements. However, many applications, especially cryptographic applications and Oblivious RAM, require insertions, builds and accesses to have a negligible failure probability, which standard Cuckoo Hashing cannot simultaneously achieve. An alternative proposal introduced by Kirsch et al. is to store elements which cannot be placed in the main table in a "stash", reducing the failure probability to $\mathcal{O}(n^{-s})$ where n is the table size and s any constant stash size. This failure probability is still not negligible. Goodrich and Mitzenmacher showed that the failure probability can be made negligible in some parameter N when $n = \Omega(\log^7(N))$ and $s = \Theta(\log(N))$. This paper presents a tighter analysis which shows failure probability negligible in N for all $n = \omega(\log(N))$ (which is asymptotically optimal) and presents explicit constants for the failure probability upper bound.

1 Introduction

Cuckoo hashing is a hash table implementation that improves performance by allowing objects to be stored in a number of locations [PR01]. Pagh and Rodler discovered that this small modification greatly reduced the probability of a build failure. Specifically, a Cuckoo Hash table can store n elements of size W in $\mathcal{O}(nW)$ space, with failure probability $\Theta(\frac{1}{n})$ (see [DK12] for the explicit constant).

For many applications this failure probability was sufficient. In the case that a failure did occur, the hash table could simply be rebuilt with new hash functions. While a hash table rebuild requires $\Theta(n)$ computation, the probability of this occurrance is $\Theta(\frac{1}{n})$, so the *amortized* computation cost per access is still constant.

*University of Pennsylvania, dgnoble@cis.upenn.edu

A series of works then began to use Cuckoo hashing for applications, most notably Oblivious RAM [PR10, GMOT12] (explained below), in which the failure probability above is unsufficent due to the following reasons:

- Indexes are sensitive data, so a rebuild constituted a security failure. (See Appendix D of [GM11] for more details.)
- Furthermore, the probability of a security failure needed to be *negligible* in some parameter $N \geq n$.
- Moreover, n, the number of data items, might be significantly smaller than (*e.g.,* polylogarithmic in) N.

A sequence of works arose to reduce the failure probability of Cuckoo hashing.

First, Kirsch, Mitzenmacher and Wieder explored the modification that any items which could not be stored in the Cuckoo Hash table would be stored in a "stash" of constant size s [KMW09]. They showed that the probability of a build failure was then reduced to $\mathcal{O}(n^{-s})$. This analysis allowed the failure probability to be reduced significantly, but it only applied to constant s, so did not allow the failure probability to be negligible in n.

While Cuckoo Hashing was initially designed to allow for a constant number of accesses, in certain situations it was acceptable to have a super-constant number of stash accesses if this could provide a negligible build failure probability. For instance, since the stash is small, the stash may be stored in a lower memory level, so accesses to the stash may be significantly cheaper than accesses to the hash table. Another important use-case is Oblivious RAM (ORAM). Oblivious RAM is a cryptographic primitive in which a trusted client can store data on an untrusted RAM. While encryption allows the client to hide the contents of the data, ORAM ensures that the client can also hide its access patterns from the untrusted RAM. Many Oblivious RAM designs use the hierarchical approach, where data is stored in various hash tables of exponentially increasing sizes. The client may have a small amount of memory available itself, which may be enough for instance to store the stash. Therefore it is reasonable in this model for the cost of stash accesses to be counted separately from those of accesses to the Cuckoo Hash table(s). Other Hierachical ORAMs allowed the client to only have constant memory usage, and solved the problem of super-constant stashes by reinserting stash elements into another level, or having a single shared stash. This meant that only non-stash elements would be accessed in any given Hash Table, allowing for a constant number of accesses in these Hash Tables.

Goodrich and Mitzenmacher developed such an Oblivious RAM protocol [GM11]. While the analysis of Kirsch et al. did not provide negligible failure probability, Goodrich and Mitzenmacher extended this analysis to achieve negligible failure probability in certain cases. They proved that, provided $n = \Omega(\log^7(N))$, a stash of size $s = \Theta(\log(N))$ would result in a failure probability negligible in N. (For tables of size $o(\log^7(N))$ another type of oblivious hashing data structure was needed.)

Aumüller, Dietzfelbinger and Woelfel then presented an elegant alternative analysis of Cuckoo Hashing with a Stash based on graph counting [ADW14]. They showed firstly, that for constant s the probability of a build failure was further upper-bounded by $\mathcal{O}(n^{-(s+1)})$. They then showed that for‘ sufficiently large n, the failure probability is $\mathcal{O}(n^{-\frac{s}{2}})$ when $s \leq n^{\frac{1}{3r}}$, for a suitable constant r.

These analyses leave open the question of how small n can be in terms of N and still have a Cuckoo Hash table with a stash with negligible probability of build failure. This paper extends these analyses to present a bound that is asymptotically tight. The bound on the failure probability also contains explicit constants. This is an important step towards constructing concrete hierarchical ORAM implementations, which require concrete bounds for build failures of small cuckoo hash tables.

2 Cuckoo Hashing

Cuckoo Hashing in its simplest form involves 2 hash functions, h_1 and h_2, and 2 hash tables, T_1 and T_2, each with $m = \epsilon n$ locations of capacity 1. Each hash table has a unique hash function, and the hash functions are assumed to produce outputs uniformly at random in $[m]$. The tables consist of pairs (x, y) where x is the dictionary key and y is the dictionary value. An item (x, y) is stored in the table by being inserted into $T_1[h_1(x)]$. If another item (x', y') was stored in that location, it is removed from its original location (like a baby bird being displaced from its nest by a Cuckoo chick) and is placed in $T_2[h_2(x')]$. This may replace another item, which the algorithm likewise attempts to insert. This process continues either until every item has found a location in which to be inserted, or some threshold on the recursion depth is reached.[1] In the latter case the insertion has “failed”. This triggers a “table rebuild” in which new tables are created with new hash functions and the algorithm attempts to insert every element into the new hash table.

Cuckoo Hash tables can be generalized to have a larger number of hash functions. They can also be generalized to use a single table. However, this work will analyze only the traditional 2-table version.

3 A Lower Bound

We begin by showing the following lower bound on the number of elements n in terms of the security parameter N, such that cuckoo hashing with a stash can fail with negligible probability in N. For consistency with other parts of the paper, we use the 2-table construction but this can easily be adapted to other constructions.

[1] Many works (e.g. [KMW09]) set this recursion depth to $\alpha \log(N)$ for a sufficiently large constant α. This makes the probability that an item that can be inserted is not inserted small, but does not make this probabality negligible. Therefore we instead in our analysis assume that the maximum recursion depth is $2n$, which ensures an optimal allocation.

Theorem 1. *If $n = \mathcal{O}(\log(N))$ and $n - s = \Omega(n)$ then it is impossible for a 2-table Cuckoo Hash table to have a negligible build failure probability in N.*

Proof. Since $n - s = \Omega(n)$, it follows that $n - s \geq c_0 n$ for sufficiently large n where the constant c_0 satisfies $0 < c_0 \leq 1$. Therefore:

$$\begin{aligned}
\frac{n-s}{n} &\geq c_0 \\
\frac{n-s-2}{n} &\geq c_0 - \frac{2}{n} \\
\frac{n-s-2}{n} &\geq \frac{c_0}{2} \text{ when } n \geq \frac{4}{c_0} \\
\frac{n-s-2}{n} &\geq c_1 \text{ for constant } c_1 \text{ satisfying } 0 < c_1 \leq \frac{1}{2}
\end{aligned}$$

Since $n = \mathcal{O}(\log(N))$, there is some constant c_2 such that $n \leq c_2 \log(N)$ (for sufficiently large n).

Let $m = \epsilon n$ be the size of each table.

If all n items are hashed to the first $\lceil \frac{n-s-2}{2} \rceil$ locations in both tables, then $2\lceil \frac{n-s-2}{2} \rceil \leq n-s-1$ items can be stored in the table, and s items can be stored in the stash, but 1 item will not be able to be stored at all, so the build fails.

The probability that all n items are stored in the first $\lceil \frac{n-s-2}{2} \rceil$ locations in both tables is at least:

$$\begin{aligned}
\left(\frac{n-s-2}{2\epsilon n}\right)^{2n} &\geq \left(\frac{c_1}{2\epsilon}\right)^{2c_2 \log(N)} \\
&\geq N^{2c_2 \log\left(\frac{c_1}{2\epsilon}\right)}
\end{aligned}$$

This is non-negligible in N. Therefore the probability of a build failure is non-negligible.

□

This immediately implies the contrapositive:

Corollary 1. *Cuckoo Hashing with a stash requires $n - s = o(n)$ or $n = \omega(\log(N))$ in order to succeed with failure negligible in N.*

The case that $n - s = o(n)$ is very unnatural–it implies that a sub-constant number of elements are stored in the table, at which point the Cuckoo table is not providing much use. Thus, in any realistic setting where Cuckoo tables are used, it is necessary that $n = \omega(\log(N))$. This provides the lower bound for n in terms of N such that Cuckoo Hashing with a stash has a negligible probability of failure. We will later provide analysis that shows that this is also the *upper bound* for Cuckoo Hashing to succeed, so this bound is tight.

4 Graph Representation

Analyses of Cuckoo Hash table failure often represent the problem as a graph problem as follows. For each location in the Cuckoo hash table, create a vertex. Since the Cuckoo hash table has two tables each of size m, there will be $2m$ vertices. For each element stored in the Cuckoo hash table, draw an edge between the two locations in which it may be stored. Let G be the resulting graph. Since there will be one location from each table, G will be bipartite, with m vertices in each part. There may also be multiple edges between a pair of vertices, so G is a multigraph. Observe also that the graph is not connected: since $n < m$ some nodes will not be connected to any edges and there may also be multiple connected components that contain edges.

Let $G(m, m, n)$ be a function that generates a graph representation of a random Cuckoo hash table, *i.e.,* it is a bipartite graph with parts A and B each of m vertices and n edges chosen uniformly at random from $A \times B$. $G \leftarrow G(m, m, n)$ represents a sampling of a random Cuckoo graph G. Let $\gamma(G)$ denote the cyclotomic number of G, that is the minimum number of edges that must be removed in order for G to be acyclic. Let $\mathbf{ex}(G)$ denote the *excess* of G, that is the minimum number of edges that must be removed from G to ensure that every connected component is acyclic or unicyclic.

Analysis is based on the following critical observation (which is proven, for instance, as Lemma 5 of [ADW14]).

Theorem 2. *Let G be the graph representation of a Cuckoo hash table with a stash of size s. Then the build succeeds if and only if $\boldsymbol{ex}(G) \leq s$.*

5 A Tight Analysis

Our primary result is the following theorem:

Theorem 3. *The probability of build failure for a 2-table cuckoo hash table with $n = \omega(\log(N))$ elements and a stash of size $s = \Theta(\log(N))$, is negligible in N.*

This follows from the following concrete bounds:

Theorem 4. *The probability of build failure for a 2-table cuckoo hash table with n elements, tables of size $m = \epsilon n$ and a stash of size s fails is:*

$$\boldsymbol{Pr}(\boldsymbol{ex}(G(m, m, n) \geq s + 1) \leq \epsilon_4(s + 2)\left(\frac{\epsilon_5(s + 1)}{n}\right)^{s+1}$$

where ϵ_4 and ϵ_5 are the following constants:

$$\epsilon_4 = \frac{16e^3\epsilon(\epsilon + 1)^2}{(\epsilon - 1)^5} e^{\frac{20\epsilon(\epsilon+1)^2}{(\epsilon-1)^5}}$$

$$\epsilon_5 = \frac{(\epsilon + 1)^2}{e\epsilon(\epsilon - 1)^5}$$

The remainder of this section proves Theorem 4, (though some more tedius lemmas are deferred to section 6).

The analysis will proceed by observing the randomly generated graph, carefully choosing which information about the graph is revealed. This will give bounds on both the size and the cyclotomic number of the component containing a randomly chosen edge, which will give bounds on the excess of the entire graph. We will make extensive use of the following algorithm, which should be viewed as occuring on a randomly generated graph. Observed variables are therefore drawn from probability distributions and the distribution of the remainder of the graph at any point is conditioned on the variables that have already been observed.

Edge Component Search Algorithm

1. While $\exists$ an undiscovered edge in G
   (a) Select one such edge at random. Call it e. Call the vertices it connects v_1 and v_2.
   (b) Let Q_e be a queue initialized to $\{v_1, v_2\}$. Set $V_e = \{\}$, $Y_e = \{e\}$.
   (c) While Q_e is not empty
      i. Set $v \leftarrow \text{dequeue}(Q_e)$.
      ii. Add v to V_e.
      iii. Set N_v to be the number of undiscovered neighbors of v, (*i.e.,* neighbors of v which have never been placed in Q_e for any edge e). This should be thought of as first observing $|N_v|$, and then observing the vertices themselves.
      iv. Enqueue all vertices in N_v to Q_e.
      v. For each w in N_v, add one of the edges connecting v to w to Y_e. (If there is more that one such edge, pick one at random, without observing the total number of such edges.)
   (d) Set $T_e = (V_e, Y_e)$.
   (e) For every pair of vertices in V_e which are in different parts of G, observe the number of unobserved edges between the vertices. Set Z_e to be the set of these edges.
   (f) Set $C_e = (V_e, Y_e \cup Z_e)$.

Theorem 5. *C_e calculated in step 1f will be the connected component in G containing the edge e chosen in step 1a.*

Proof. Observe that steps 1b and 1c are identical to a Breadth First Search (BFS), except that the queue begins containing two vertices instead of 1. However, v_2 is a neighbor of v_1, so the initial state of the system can be viewed as the state of a BFS starting at v_1 where the first neighbor (v_2) has already been found, and added to the queue. Therefore, the resulting BFS will find exactly

the nodes reachable from v_1 in G, which is exactly the nodes in the connected component in G containing e. Observe also that any edges that may exist in C_e are found, either in steps 1b and 1(c)v, in which case they are added to Y_e, or in step 1e in which case they are added to Z_e. Either way, these edges exist in C_e. Lastly, only edges in the connected component containing e exist in C_e, since only edges in G connecting vertices in C_e are added. □

Observing the correspondence to a BFS also indicates the following Lemma.

Lemma 1. *T_e calculated in step 1d is a spanning tree of C_e.*

Furthermore, Y_e and Z_e are disjoint. Therefore, Z_e contains a set of edges in C_e which, if removed, produces a tree. This implies the following fact.

Fact 1. $|Z_e| = \gamma(C_e)$

Lastly, since the random edge selected in 1a will always be one that has not yet been discovered, and all edges in a component are discovered when that component is explored, each new component found will be separate to all previous components found. Furthermore, since the algorithm continues until all edges are found, it will find all components of G.

Now we upper-bound the number of neighbors found in step 1(c)iii.

Theorem 6. *The number of neighbors found in step 1(c)iii is stochastically dominated by Bin$(n, \frac{1}{m})$*

Proof. First, the number of undiscovered neighbors of v found in step 1(c)iii is at most the number of undiscovered edges that connect to v. Let u be the number of discovered edges at a certain point of time, and $n-u$ be the number of undiscovered edges. Let A be the part of the bipartite graph containing v and B the other part. Each of the u edges has one end-point in A. One of these u edges is known to have its end-point in A at v (for v_1, v_2 this is e, and for other vertices, it is the edge that was used to find v). Therefore, there are $u-1$ edges that have other end-points in A, and so at most $u-1$ vertices in A that are end-points of previously discovered edges. Only vertices that are end-points of a previously-discovered edge may have their number of neighbors examined (in step 1(c)iii). Therefore, there are at least $m-u+1$ vertices in A (including v) which prior to to step 1(c)iii have not had their number of neighbors examined.

Some of the remaining $n-u$ edges may be later discovered to exist between previously found vertex-pairs (in step 1e). The number of undiscovered edges that are not in this category is still at most $n-u$.

Therefore, there are at most $n-u$ edges that could contribute towards $|N_v|$, and for each, the only thing that is known about the edge's end-point in A is that it is not one of the at most $u-1$ vertices in A which have had their number of neighbors counted. Hence, each such edge will have v as its end-point in A with probability at most $\frac{1}{m-u+1}$. Since there are at most $n-u$ such edges, $|N_v|$ is stochastically dominated by Bin$(n-u, \frac{1}{m-u+1})$, which by Lemma 4 is stochastically dominated by Bin$(n, \frac{1}{m})$

□

We now need to show bounds on the number of edges found in step 1e. It will help to first define three types of vertex pairs. The first are opened vertex pairs, for which the number of edges between the pair of vertices is fully known (including when it is known to be zero). Step 1e cannot find any edges between opened vertex pairs, since it only finds previously undiscovered edges. The second type is partially opened vertex pairs, for which it is known that at least one edge exists between them but it is not known how many more exist. The third type is unopened vertex pairs, for which it is not yet known whether the vertices are neighbors.

We begin by showing bounds on the number of edges between unopened vertex pairs.

Theorem 7. *In the Edge Component Search Algorithm, if at a point in time u edges have been discovered, then the number of edges between an unopened vertex pair, v and w, is stochastically dominated by Bin$(n, \frac{1}{m(m-u)})$.*

Proof. First we show that the number of edges between v and w is stochastically dominated by $\text{Bin}(n-u, \frac{1}{(m-u)^2})$.

Let q be the number of edges that exist between unopened vertex pairs. Every such edge must not yet have been discovered, but there may be some undiscovered edges between partially opened vertex pairs. Therefore $q \leq n-u$

For the q edges that exist between unopened vertex pairs, we do not know any information about which vertices they exist between beyond the fact that they exist between unopened vertex pairs. Futhermore, it is equally likely to exist between any such pair.

Since only u edges have been discovered, there must be at least $m-u$ vertices in each part that touch no discovered edges. Hence, each pair of such vertices is an unopened vertex pair. Therefore, there are at least $(m-u)^2$ unopened vertex pairs. Thus, for any given unopened pair, and an edge that exists between a unopened pair, the probability that the edge exists between that unopened pair is at most $\left(\frac{1}{m-u}\right)^2$. Hence, the number of edges between any given unopened pair will be stochastically dominated by $\text{Bin}(n-u, \frac{1}{(m-u)^2})$, which by Lemma 5, is stochastically dominated by $\text{Bin}(n, \frac{1}{m(m-u)})$. □

Next we show bounds on the number of additional edges found in step 1e between partially opened pairs.

Theorem 8. *In the Edge Component Search Algorithm, the number of additional edges between a partially opened vertex pair is stochastically dominated by the number of edges between an unopened vertex pair.*

(Proof deferred to section 6).

Combining this with Theorem 7 and observing that $u \leq n$, we get the following result.

Theorem 9. *In the Edge Component Search Algorithm, any vertex pair that is partially opened or unopened, has a number of undiscovered edges that is stochastically dominated by Bin$(n, \frac{1}{m(m-n)})$.*

Now define a function $H(m,n)$, which samples a graph $H \leftarrow H(m,n)$ chosen the same as C_e in step 1 of the Edge Component Search Algorithm except that:

- Edges and vertices are given new unique identifiers when discovered that may not be the same as the names "found" by the Edge Component Search Algorithm.
- $|N_v|$ in step 1(c)iii is chosen from $\text{Bin}(n, \frac{1}{m})$
- In step 1e, the additional edges between any pair of vertices in different parts is chosen from $\text{Bin}(n, \frac{1}{m(m-n)})$. (Recall the graph is a tree at this point, so is bipartite.)
- We refer to V_e as V, Q_e as Q, Y_e as Y and Z_e as Z.

Theorem 10. *For any component C_e discovered in the Edge Component Search Algorithm, $\gamma(C_e)$ is stochastically dominated by $\gamma(H)$ for an independent sample $H \leftarrow H(m,n)$.*

Proof. We can view the two graph-sampling algorithms as running in parallel using the same source of randomness. We can choose an interpretation of the randomness generated such that if an event in the sampling of H stochastically dominates an event in the sampling of C_e, the event always happens in H if it happens in C_e. Since the probability of finding an edge in C_e is always stochastically dominated by that of finding the edge in H (from Theorems 6 and 9), C_e will be a subset of H for any choice of randomness. Therefore $\gamma(C_e) \leq \gamma(H)$ for any choice of randomness, which implies that $\gamma(C_e)$ is stochastically dominated by $\gamma(H)$. □

We can now upper bound $|H|$ and $\gamma(H)$ in order to upper bound $\gamma(C_e)$.

Theorem 11. *For $H \leftarrow H(m,n)$, where $m = \epsilon n$ for $\epsilon > 1$, and $\epsilon_1 = \frac{(\epsilon-1)^2}{\epsilon+1}$, for $k \geq 2$,*

$$\boldsymbol{Pr}(|H| \geq k) \leq \frac{2\epsilon^2}{k-1} e^{-\epsilon_1 k}$$

Proof. Now, the vertices of H are found by each vertex having a number of children chosen from the distribution $\text{Bin}(n, \frac{1}{m})$. Therefore (V, Y) can be viewed as the result of a Galton-Watson Branching process, with 2 roots, and children chosen from independent samples of $\text{Bin}(n, \frac{1}{m})$. The Otter-Dwass formula [Pit98, Dwa69] states that the probability that a Galton-Watson process that initially has α nodes, will be of size k is exactly

$$\frac{\alpha}{k}\mathbf{Pr}(S_k = k - \alpha)$$

where S_k is the distribution of k samples of the progeny distribution. In this case $S_k = \text{Bin}(nk, \frac{1}{m})$. Therefore, for $k \geq 2$

$$\mathbf{Pr}(|H| = k) = \frac{2}{k}\mathbf{Pr}(\text{Bin}(nk, \frac{1}{m}) = k - 2)$$

$$
\begin{aligned}
&= \frac{2(k-1)}{(kn-k+2)(kn-k+1)} m^2 \left(1-\frac{1}{m}\right)^2 \mathbf{Pr}(\mathrm{Bin}(nk, \frac{1}{m}) = k) \\
&\leq \frac{2\epsilon^2(k-1)}{(k-\frac{k}{n})(k-\frac{k}{n})} \mathbf{Pr}(\mathrm{Bin}(nk, \frac{1}{m}) = k) \\
&\leq \frac{2\epsilon^2}{k-1} \mathbf{Pr}(\mathrm{Bin}(nk, \frac{1}{m}) = k) \\
&\leq \frac{2\epsilon^2}{k-1} \mathbf{Pr}(\mathrm{Bin}(nk, \frac{1}{m}) \geq k) \\
&\leq \frac{2\epsilon^2}{k-1} e^{-\epsilon_1 k}
\end{aligned}
$$

where $\epsilon_1 = \frac{(\epsilon-1)^2}{\epsilon+1}$ and the last step comes from a Chernoff Bound.

□

Now we can bound $\gamma(H)$ for a given $|H|$.

Theorem 12. *For $H \leftarrow H(m,n)$,*

$$\boldsymbol{Pr}(\gamma(H) \geq t | |H| = k) \leq \left(\frac{enk^2}{4m(m-n)t}\right)^t$$

Proof. H is bipartite. If one part has size a, the other has size $k-a$. The cylotomic number of H is the number of additional edges added in the last step. The number of pairs of vertices that may have edges added between them is $a(k-a)$ which has maximum value $\lfloor \frac{k}{2} \rfloor \lceil \frac{k}{2} \rceil = \lfloor \frac{k^2}{4} \rfloor$.

Each such vertex pair has a number of edges drawn from the distribution $\mathrm{Bin}(n, \frac{1}{(m-n)m})$. Therefore the total number of edges is stochastically dominated by $\mathrm{Bin}(n\lfloor \frac{k^2}{4} \rfloor, \frac{1}{m(m-n)})$.

Applying the Chernoff bound from Lemma 7 completes the proof. □

Theorem 13. *For any component C_e found by the Edge Component Search Algorithm,*

$$\boldsymbol{Pr}(\gamma(C_e) \geq t) \leq \epsilon_2 \left(\frac{\epsilon_3 t}{m}\right)^t$$

where $\epsilon_2 = 8e\epsilon^2$ and $\epsilon_3 = \frac{1}{e(\epsilon-1)\epsilon_1^2} = \frac{(\epsilon+1)^2}{e(\epsilon-1)^5}$.

Proof. First we show bounds on $\gamma(H)$, where $H \leftarrow H(m,n)$. Combining Theorem 11 and Theorem 12 we can obtain bounds for $\gamma(H)$:

$$\mathbf{Pr}(\gamma(H) \geq t) \leq \sum_{k=2}^{\infty} \mathbf{Pr}(|H| = k) \mathbf{Pr}(\gamma(H) \geq t | |H| = k)$$

$$\leq \sum_{k=2}^{\infty} \frac{2\epsilon^2}{k-1} e^{-\epsilon_1 k} \left(\frac{enk^2}{4m(m-n)t}\right)^t$$
$$\leq 2\epsilon^2 \left(\frac{en}{4m(m-n)t}\right)^t \sum_{k=2}^{\infty} \frac{1}{k-1} e^{-\epsilon_1 k} k^{2t}$$
$$\leq 4\epsilon^2 \left(\frac{en}{4m(m-n)t}\right)^t \sum_{k=2}^{\infty} e^{-\epsilon_1 k} k^{2t-1}$$

Applying Lemma 8 yields:

$$\leq 4\epsilon^2 \left(\frac{en}{4m(m-n)t}\right)^t 2e \left(\frac{2t}{\epsilon_1 e}\right)^{2t}$$
$$\leq 8e\epsilon^2 \left(\frac{t}{em(\epsilon-1)\epsilon_1^2}\right)^t$$

Since $\gamma(C_e)$ is stochastically dominated by $\gamma(H)$,

$$\mathbf{Pr}(\gamma(C_e) \geq t) \leq 8e\epsilon^2 \left(\frac{t}{e(\epsilon-1)\epsilon_1^2 m}\right)^t$$

□

This immediately implies the following corollary:

Corollary 2. *For any component C_e found by the Edge Component Search Algorithm,*

$$\boldsymbol{Pr}(\boldsymbol{ex}(C_e) \geq s) \leq \epsilon_2 \left(\frac{\epsilon_3(s+1)}{m}\right)^{s+1}$$

where $\epsilon_2 = 8e\epsilon^2$ and $\epsilon_3 = \frac{1}{e(\epsilon-1)\epsilon_1^2} = \frac{(\epsilon+1)^2}{e(\epsilon-1)^5}$.

Note that this bound not only applies to the first component found, but to *every* component found.

Let C'_e be the component containing e if e is the first edge found in the Edge Component Search Algorithm and let C'_e be an empty component otherwise. $\mathbf{ex}(C'_e) = \mathbf{ex}(C_e)$ if e is the first edge found in C_e, and $\mathbf{ex}(C'_e) = 0$ otherwise. In either case the bound of Corollary 2 applies to $\mathbf{ex}(C'_e)$.

We will need the following Lemma (proven in section 6).

Lemma 2. *Let $U(s,q)$ be the set of sequences of positive integers, where $T \in U$ if and only if $|T| = q$ and $\sum_{1\leq i\leq q} T_i = s$, where $s \geq q \geq 1$. Then $\sum_{T\in U(s,q)} \prod_{1\leq i\leq q} (T_i+1)^{T_i+1} \leq 0.89^{q-1}(s+1)^{s+1}$*

We can now bound the excess of the entire graph,

$$
\begin{aligned}
\mathbf{Pr}(\mathbf{ex}(G) \geq s) &= \mathbf{Pr}(\sum_e \mathbf{ex}(C'_e) \geq s) \\
&\leq \sum_{\substack{j_1,\ldots,j_n \\ \sum_i j_i = s}} \mathbf{Pr}(\wedge_i \mathbf{ex}(C'_{e_i}) \geq j_i) \\
&\leq \sum_{q=1}^{s} \sum_{\substack{j_1,\ldots,j_n \\ \sum_i j_i = s \\ |\{j_i : j_i \geq 1\}| = q}} \prod_{\{j_i : j_i \geq 1\}} \epsilon_2 \left(\frac{\epsilon_3(j_i+1)}{m}\right)^{j_i+1} \\
&\leq \sum_{q=1}^{s} \sum_{\substack{R \subseteq \{1,\ldots,n\} \\ |R| = q}} \sum_{\substack{j_1,\ldots,j_q \\ \sum_i j_i = s \\ j_i \geq 1}} \prod_{i=1}^{q} \epsilon_2 \left(\frac{\epsilon_3(j_i+1)}{m}\right)^{j_i+1} \\
&\leq \sum_{q=1}^{s} \binom{n}{q} \sum_{T \in U(s,q)} \prod_{i=1}^{q} \epsilon_2 \left(\frac{\epsilon_3(T_i+1)}{m}\right)^{T_i+1} \\
&\leq \sum_{q=1}^{s} \binom{n}{q} \epsilon_2^q \left(\frac{\epsilon_3}{m}\right)^{s+q} \sum_{T \in U(s,q)} \prod_{i=1}^{q} (T_i+1)^{T_i+1}
\end{aligned}
$$

Apply Lemma 10:

$$
\begin{aligned}
&\leq \sum_{q=1}^{s} \binom{n}{q} \epsilon_2^q \left(\frac{\epsilon_3}{m}\right)^{s+q} 0.89^{q-1}(s+1)^{s+1} \\
&\leq \frac{1}{0.89}(s+1)^{s+1} \left(\frac{\epsilon_3}{m}\right)^{s} \sum_{q=1}^{s} \left(\frac{0.89 e \epsilon_2 \epsilon_3}{\epsilon q}\right)^{q} \\
&\leq \frac{(s+1)e}{0.89} \left(\frac{\epsilon_3 s}{m}\right)^{s} \sum_{q=1}^{s} \left(\frac{0.89 e \epsilon_2 \epsilon_3}{\epsilon q}\right)^{q}
\end{aligned}
$$

Applying Lemma 11:

$$
\begin{aligned}
&\leq \frac{(s+1)e}{0.89} \left(\frac{\epsilon_3 s}{m}\right)^{s} \frac{2e 0.89 \epsilon_2 \epsilon_3}{\epsilon} e^{\frac{0.89\epsilon_2\epsilon_3}{\epsilon}} \\
&\leq \frac{(s+1)16e^3\epsilon(\epsilon+1)^2}{(\epsilon-1)^5} e^{\frac{20\epsilon(\epsilon+1)^2}{(\epsilon-1)^5}} \left(\frac{\epsilon_3 s}{m}\right)^{s}
\end{aligned}
$$

This completes the proof of Theorem 4.

6 Additional Lemmas

This section contains proofs of (more tedius) lemmas.

We begin by showing a useful lemma for inequalities with exponentials.

Lemma 3.

$$a_2 b_1 \leq a_1 b_2 \Rightarrow \left(1 - \frac{1}{a_1 + 1}\right)^{b_1} \geq \left(1 - \frac{1}{a_2}\right)^{b_2}$$

Proof.

$$\begin{aligned}
\left(1 - \frac{1}{a_1 + 1}\right)^{b_1} &\geq \left(1 - \frac{1}{a_2}\right)^{b_2} \Leftrightarrow \\
b_1 \ln\left(1 - \frac{1}{a_1 + 1}\right) &\geq b_2 \ln\left(1 - \frac{1}{a_2}\right) \Leftarrow \\
b_1 \left(1 - \frac{1}{1 - \frac{1}{a_1 + 1}}\right) &\geq b_2 \left(-\frac{1}{a_2}\right) \Leftrightarrow \\
b_1 \left(1 - \frac{a_1 + 1}{a_1}\right) &\geq -\frac{b_2}{a_2} \Leftrightarrow \\
-\frac{b_1}{a_1} &\geq -\frac{b_2}{a_2} \Leftrightarrow \\
a_2 b_1 &\leq a_1 b_2
\end{aligned}$$

□

Lemma 4. *Bin*$(n - q, \frac{1}{m-q+1})$ *is stochastically dominated by Bin*$(n, \frac{1}{m})$ *when* $m \geq n$.

Proof. Now $\mathrm{Bin}(n_1, p_1)$ is stochastically dominated by $\mathrm{Bin}(n_2, p_2)$ if and only if $n_1 \leq n_2$ and $(1 - p_1)^{n_1} \geq (1 - p_2)^{n_2}$ [KM10]. Set $n_1 = n - q$, $n_2 = n$, $p_1 = \frac{1}{m-q+1}$, $p_2 = \frac{1}{m}$. Clearly $n_1 \leq n_2$. Observe that

$$\left(1 - \frac{1}{m - q + 1}\right)^{n-q} \geq \left(1 - \frac{1}{m}\right)^{n} \Leftarrow$$

Applying Lemma 3

$$\begin{aligned}
(n - q)m &\leq n(m - q) \Leftrightarrow \\
-qm &\leq -qn \Leftrightarrow \\
m &\geq n
\end{aligned}$$

The last statement is true, so the condition $(1-p_1)^{n_1} \geq (1-p_2)^{n_2}$ holds, □

Lemma 5. *If integers* m, n, u *satisfy* $m > n \geq u \geq 1$ *then Bin*$(n - u, \frac{1}{(m-u)^2})$ *is stochastically dominated by Bin*$(n, \frac{1}{m(m-u)})$.

Proof. $\mathrm{Bin}(n_1, p_1)$ is stochastically dominated by $\mathrm{Bin}(n_2, p_2)$ if and only if $n_1 \leq n_2$ and $(1-p_1)^{n_1} \geq (1-p_2)_2^n$ [KM10]. Set $n_1 = n-u$, $n_2 = n$, $p_1 = \frac{1}{(m-u)^2}$ and $p_2 = \frac{1}{m(m-u)}$. Clearly $n_1 \leq n_2$. Now

$$\left(1 - \frac{1}{(m-u)^2}\right)^{n-u} \geq \left(1 - \frac{1}{m(m-u)}\right)^n \Leftarrow$$

Applying Lemma 3

$$\begin{aligned}
(n-u)m(m-u) &\leq n\left((m-u)^2 - 1\right) \\
(n-u)m &\leq n(m-u-\frac{1}{m-u}) \Leftrightarrow \\
um &\geq n(u+\frac{1}{m-u}) \Leftrightarrow \\
\frac{m}{n} &\geq \frac{u+\frac{1}{m-u}}{u} \Leftrightarrow \\
\frac{m-n}{n} &\geq \frac{1}{u(m-u)} \Leftrightarrow \\
\frac{n}{m-n} &\leq u(m-u)
\end{aligned}$$

The last statement holds since $u(m-u) \geq m-1 \geq n \geq \frac{n}{m-n}$. Hence $(1-p_1)^{n_1} \geq (1-p_2)^{n_2}$ as required, so $\mathrm{Bin}(n-u, \frac{1}{(m-u)^2})$ is stochastically dominated by $\mathrm{Bin}(n, \frac{1}{m(m-u)})$.

□

Theorem 8. *In the Edge Component Search Algorithm, the number of additional edges between a partially opened vertex pair is stochastically dominated by the number of edges between an unopened vertex pair.*

Proof. The case of the pair v_1, v_2 is special because the initial edge e that was found between these was discovered by selecting a random undiscovered edge, rather that requesting information about the pair v_1, v_2. Therefore, the occurrance of e between v_1 and v_2 does not affect the distribution of other edges. Hence the remaining edges between v_1 and v_2 will actually be distributed exactly the same as between any unopened vertex pair.

For the remaining partially opened vertex pairs, we will prove a slightly different statement, which implies the one above. Let A be the number of additional edges between the partially opened pair and B be the number between the unopened pair. We show that if $i < j$, the probability that $A = i$ and $B = j$ is less than the probability that $A = j$ and $B = i$.

Let there be some state, S, observed on the remainder of the system. By Bayes:

$$\frac{\mathbf{Pr}(B = i \wedge A = j+1|S)}{\mathbf{Pr}(B = j \wedge A = i+1|S)} = \frac{\mathbf{Pr}(B = i \wedge A = j+1)\mathbf{Pr}(S|B = i \wedge A = j+1)}{\mathbf{Pr}(B = j \wedge A = i+1)\mathbf{Pr}(S|B = j \wedge A = i+1)}$$

Now $\mathbf{Pr}(S|B = i \wedge A = j+1) = \mathbf{Pr}(S|B = j \wedge A = i+1)$, since in both cases $i+j+1$ edges will have been used between the two vertex-pairs. Therefore, the probability above is simply

$$\frac{\mathbf{Pr}(B = i \wedge A = j+1)}{\mathbf{Pr}(B = j \wedge A = i+1)}$$

Note that this statement is true regardless of what the state S is (as long as it is possible), so we can consider S to be all information learned about the assignment of edges from the beginning of the Edge Component Search Algorithm.

So, if there are initially b vertex pairs and n edges:

$$\begin{aligned}
\frac{\mathbf{Pr}(B = i \wedge A = j+1|S)}{\mathbf{Pr}(B = j \wedge A = i+1|S)} &= \frac{\mathbf{Pr}(B = i \wedge A = j+1)}{\mathbf{Pr}(B = j \wedge A = i+1)} \\
&= \frac{\mathbf{Pr}(B = i|A = j+1)\mathbf{Pr}(A = j+1)}{\mathbf{Pr}(B = j|A = i+1)\mathbf{Pr}(A = i+1)} \\
&= \frac{\binom{n-j-1}{i}\left(\frac{1}{b-1}\right)^i\left(1-\frac{1}{b-1}\right)^{n-j-1-i}\binom{n}{j+1}\left(\frac{1}{b}\right)^{j+1}\left(1-\frac{1}{b}\right)^{n-j-1}}{\binom{n-i-1}{j}\left(\frac{1}{b-1}\right)^j\left(1-\frac{1}{b-1}\right)^{n-i-1-j}\binom{n}{i+1}\left(\frac{1}{b}\right)^{i+1}\left(1-\frac{1}{b}\right)^{n-i-1}} \\
&= \frac{\binom{n-j-1}{i}\binom{n}{j+1}}{\binom{n-i-1}{j}\binom{n}{i+1}} \\
&= \frac{i+1}{j+1} < 1
\end{aligned}$$

Therefore, $\mathbf{Pr}(B = i \wedge A = j+1|S) < \mathbf{Pr}(B = j \wedge A = i+1|S)$, when $i < j$, which implies that after observation of the sysem S, $A-1$ is stochastically dominated by B. Therefore, the number of additional edges between a partially opened pair is stochastically dominated by the number of edges between an unopened pair. □

Lemma 6. *Let X be the sum of indepenent Bernoulli variables, with mean μ. A basic form of the Chernoff Bound for any $\delta > 0$ is as follows:*

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

This implies the following looser bound:

Lemma 7. *For any non-negative integer t,*

$$\boldsymbol{Pr}(X \geq t) \leq \left(\frac{e\mu}{t}\right)^t$$

Proof. For $t \leq \mu$, $\frac{e\mu}{t} \geq e$, so $\left(\frac{e\mu}{t}\right)^t \geq 1$, so the statement holds as the probability cannot be more than 1. For $t > \mu$ we can view $t = (1+\delta)\mu$ for some $\delta > 0$.

$$\begin{aligned}
\mathbf{Pr}(X \geq (1+\delta)\mu) &\leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu \\
&\leq \left(\frac{e}{(1+\delta)}\right)^{(1+\delta)\mu} e^{-\mu} \\
&\leq \left(\frac{e\mu}{(1+\delta)\mu}\right)^{(1+\delta)\mu} e^{-\mu} \\
&\leq \left(\frac{e\mu}{t}\right)^t
\end{aligned}$$

□

Lemma 8. *Let $t \geq 1$, $\epsilon_1 \in (0,1)$. Then:*

$$\sum_{k=1}^{\infty} k^{2t-1} e^{-\epsilon_1 k} \leq 2e \left(\frac{2t}{\epsilon_1 e}\right)^{2t}$$

Proof. It is possible to approximate a summation with an integral, using the same methods as Reinman sums but in reverse. Let $f(x)$ be a continuous function that is monotonically increasing until a maximum point x_{max}, after which x is monotonically decreasing. Let $x' = \lfloor x_{max} \rfloor$. Let $h(x) = \min(f(\lfloor x \rfloor), f(\lfloor x \rfloor + 1))$. Let us observe how $\sum_{x=a}^{b} h(x)$ approximates $\int_a^{b+1} f(x)dx$.

Observe that for any integer a, $h(x)$ is the same for all $x \in [a, a+1)$. Since $f(x)$ has no local minima and is continuous, the minimum value of $f(x)$ over the range $[a, a+1)$ is either at $f(a)$ or $f(a+1)$. Therefore $f(x) \geq \min(f(a), f(a+1)) = h(x)$ for $x \in [a, a+1)$. Since this applies to the interval $[a, a+1)$ for any integer a, $h(x) \leq f(x)$ for all x. Hence, for any integers a and b, $\sum_a^b h(x) = \int_a^{b+1} h(x) \leq \int_a^{b+1} f(x)dx$.

$$\begin{aligned}
\int_a^{b+1} f(x)dx &\geq \sum_a^b h(x) \\
&\geq \sum_a^{x'-1} f(x) + min(f(x'), f(x'+1)) + \sum_{x'+1}^{b} f(x+1) \\
&\geq \left(\sum_a^{b+1} f(x)\right) - max(f(x'), f(x'+1) \\
&\geq \left(\sum_a^{b+1} f(x)\right) - f(x_{max})
\end{aligned}$$

Hence:

$$\sum_{a}^{b} f(x) \leq \int_{a}^{b} f(x)dx + f(x_{max})$$

Let $f(x) = x^{2t-1}e^{-\epsilon_1 x}$ where $t \geq 1$ and $0 < \epsilon_1 < 1$. First we need to show that it is a function that is monitonically increasing, then monitonically decreasing.

$$f'(x) = (2t-1)x^{2t-2}e^{-\epsilon_1 x} - \epsilon_1 x^{2t-1}e^{-\epsilon_1} \qquad = x^{2t-2}e^{-\epsilon_1}(2t-1-\epsilon_1 x)$$

Observe that x^{2t-2} and $e^{-\epsilon_1}$ are both positive. Therefore $f'(x)$ will be positive when $x < \frac{2t-1}{\epsilon_1}$, $f'(x) = 0$ at $x = \frac{2t-1}{\epsilon_1}$ and will be negative when $x > \frac{2t-1}{\epsilon_1}$. Therefore this function is monotonically increasing, then monitonically decreasing, as required, with $x_{max} = \frac{2t-1}{\epsilon_1}$. We can easily calculate:

$$\begin{aligned} f(x_{max}) &= \left(\frac{2t-1}{\epsilon_1}\right)^{2t-1} e^{-(2t-1)} \\ &= \left(\frac{2t-1}{\epsilon_1 e}\right)^{2t-1} \end{aligned}$$

Hence the inequality applies to the sum and

$$\begin{aligned} \sum_{k=1}^{\infty} k^{2t-1}e^{-\epsilon_1 k} &\leq \int_{1}^{\infty} x^{2t-1}e^{-\epsilon_1 x}dx + \left(\frac{2t-1}{\epsilon_1 e}\right)^{2t-1} \\ &\leq \int_{0}^{\infty} x^{2t-1}e^{-\epsilon_1 x}dx + \left(\frac{2t-1}{\epsilon_1 e}\right)^{2t-1} \end{aligned}$$

By a standard integral identity, $\int_0^\infty x^{2t-1}e^{-\epsilon_1 x}dx = \frac{(2t-1)!}{\epsilon_1^{2t}}$. Furthermore, a factorial approximation shows that $(2t-1)! \leq (2t)^{2t}e^{-(2t-1)}$. Hence

$$\begin{aligned} \sum_{k=1}^{\infty} k^{2t-1}e^{-\epsilon_1 k} &\leq \left(\frac{2t}{\epsilon_1}\right)^{2t} e^{-(2t-1)} + \left(\frac{2t-1}{\epsilon_1 e}\right)^{2t-1} \\ &\leq \left(\frac{2t}{\epsilon_1}\right)^{2t} e^{-(2t-1)} + \left(\frac{2t}{\epsilon_1}\right)^{2t-1} e^{-(2t-1)} \end{aligned}$$

Recalling that $0 < \epsilon_1 < 1$, so $\frac{2t}{\epsilon_1} > 1$

$$\begin{aligned} &\leq 2\left(\frac{2t}{\epsilon_1}\right)^{2t} e^{-(2t-1)} \\ &\leq 2e\left(\frac{2t}{\epsilon_1 e}\right)^{2t} \end{aligned}$$

□

Lemma 9. *For all integers $s \geq 2$, $\sum_{a=1}^{s-1}(a+1)^{a+1}(s+1-a)^{s+1-a} \leq 0.89(s+1)^{s+1}$*

Proof. By calculation, this is true for $s \in \{2,3,4,5,6,7,8\}$, for which the left-hand side values are, respectively $\{16, 216, 2777, 38824, 607534, 10707768, 212342547\}$ and the right-hand side values are respectively $\{24.03, 227.84, 2781.25, 41523.84, 732953.27, 14931722.24, 344804235.2\}$. For $s > 8$ we prove by induction.

Given that it holds true for $s \leq 8$, let us show it holds true for $s+1$.

$$\sum_{a=1}^{s}(a+1)^{a+1}(s+2-a)^{s+2-a}$$
$$\leq \sum_{a=1}^{t}(a+1)^{a+1}(s+2-a)^{s+2-a} + \sum_{a=t+1}^{s-1}(a+1)^{a+1}(s+2-a)^{s+2-a} + (s+1)^{s+1}2^2$$
$$\leq \sum_{a=1}^{t}(a+1)^{a+1}(s+2-a)^{s+2-a} + (s+1-t)e\sum_{a=t+1}^{s-1}(a+1)^{a+1}(s+1-a)^{s+1-a} + (s+1)^{s+1}2^2$$
$$\leq \sum_{a=1}^{t}(a+1)^{a+1}(s+2-a)^{s+2-a} + (s+1-t)e$$
$$\left(\sum_{a=1}^{s-1}(a+1)^{a+1}(s+1-a)^{s+1-a} - \sum_{a=1}^{t}(a+1)^{a+1}(s-a+1)^{s-a+1}\right) + (s+1)^{s+1}2^2$$

Applying the inductive hypothesis yields:

$$\leq e\sum_{a=1}^{t}(a+1)^{a+1}(s+2-a)(s+1-a)^{s+1-a} + (s+1-t)e0.89(s+1)^{s+1}$$
$$- e(s+1-t)\sum_{a=1}^{t}(a+1)^{a+1}(s-a+1)^{s-a+1} + (s+1)^{s+1}2^2$$
$$\leq e\sum_{a=1}^{t}(a+1)^{a+1}(s-a+1)^{s-a+1}((s+2-a)-(s+1-t)) + 0.89(s+2)^{s+2}$$
$$- te0.89(s+1)^{s+1} + (s+1)^{s+1}2^2$$
$$\leq 0.89(s+2)^{s+2} + 2^2(s+1)^{s+1} + e\sum_{a=1}^{t}(a+1)^{a+1}(s-a+1)^{s-a+1}(t+1-a) - te0.89(s+1)^{s+1}$$

Setting $t = 3$ yields:

$$\leq 0.89(s+2)^{s+2} + 2^2(s+1)^{s+1} + e2^2s^s3 + e3^3(s-1)^{s-1}2 + e4^4(s-2)^{s-2} - 3e0.89(s+1)^{s+1}$$

$$\leq 0.89(s+2)^{s+2} + (s+1)^{s+1}\left(2^2 + \frac{12}{s} + \frac{54}{es(s-1)} + \frac{256}{e^2 s(s-1)(s-2)} - 3e0.89\right)$$

For $s \geq 8$, the term $2^2 + \frac{12}{s} + \frac{54}{es(s-1)} + \frac{256}{e^2 s(s-1)(s-2)} \leq 5.5$. Since $3e0.89 > 5.5$ the inequality simplifies to:

$$\sum_{a=1}^{s} (a+1)^{a+1}(s+2-a)^{s+2-a} \leq 0.89(s+2)^{s+2}$$

Since it holds true up to $s = 2, \ldots, 8$ by inspection, and holds true for $s \geq 8$ by induction, the statement is true for all $s \geq 2$. □

Lemma 10. *Let $U(s,q)$ be the set of sequences of positive integers, where $T \in U$ if and only if $|T| = q$ and $\sum_{1\leq i\leq q} T_i = s$, where $s \geq q \geq 1$. Then $\sum_{T\in U(s,q)} \prod_{1\leq i\leq q} (T_i+1)^{T_i+1} \leq 0.89^{q-1}(s+1)^{s+1}$*

Proof. We proceed by induction on the length of the sequences. For $q = 1$, U contains a single sequence T with $T_1 = s$. Then $\sum_{T\in U(s,q)} \prod_{1\leq i\leq q} (T_i+1)^{T_i+1} = (s+1)^{s+1} = 0.89^0(s+1)^{s+1}$.

Assume that the theorem holds for all sequences of length $q \geq 1$. We will show that it also holds for all sequences of length $q+1$.

$$\sum_{T\in U(s,(q+1))} \prod_{1\leq i\leq q+1} (T_i+1)^{T_i+1} \leq \sum_{T_1=1}^{s-q} (T_1+1)^{T_1+1} \sum_{T'\in U((s-T_1),q)} \prod_{1\leq i\leq q} (T'_i+1)^{T'_i+1}$$

Applying our inductive hypothesis gives:

$$\leq \sum_{a=1}^{s-1} (a+1)^{a+1} 0.89^{q-1}(s-a+1)^{s-a+1}$$
$$\leq 0.89^{q-1} \sum_{a=1}^{s-1} (a+1)^{a+1}(s-a+1)^{s-a+1}$$

Using Lemma 9

$$\leq 0.89^q (s+2)^{s+2}$$

□

Lemma 11. *For $a > 0$,*

$$\sum_{x=1}^{\infty} \left(\frac{a}{x}\right)^x \leq 2ae^{\frac{a}{e}}$$

Proof. Over the positive reals the function $f(x) = \left(\frac{a}{x}\right)^x$ is maximized at $x = \frac{a}{e}$, for which it has value $e^{\frac{a}{e}}$. Therefore:

$$\begin{aligned}
\sum_{x=1}^{\infty} \left(\frac{a}{x}\right)^x &\leq \sum_{x=1}^{2a-1} \left(\frac{a}{x}\right)^x + \sum_{x=2a}^{\infty} \left(\frac{a}{x}\right)^x \\
&\leq \sum_{x=1}^{2a-1} (e^{\frac{a}{e}}) + \sum_{x=2a}^{\infty} \left(\frac{1}{2}\right)^x \\
&\leq e^{\frac{a}{e}}(2a-1) + 1 \\
&\leq 2ae^{\frac{a}{e}}
\end{aligned}$$

□

7 Acknowledgements

This research was sponsored in part by ONR grant (N00014-15-1-2750) "SynCrypt: Automated Synthesis of Cryptographic Constructions".

References

[ADW14] Martin Aumüller, Martin Dietzfelbinger, and Philipp Woelfel. Explicit and efficient hash families suffice for cuckoo hashing with a stash. *Algorithmica*, 70(3):428–456, 2014.

[DK12] Michael Drmota and Reinhard Kutzelnigg. A precise analysis of cuckoo hashing. *ACM Transactions on Algorithms (TALG)*, 8(2):1–36, 2012.

[Dwa69] Meyer Dwass. The total progeny in a branching process and a related random walk. *Journal of Applied Probability*, 6(3):682–686, 1969.

[GM11] Michael T Goodrich and Michael Mitzenmacher. Privacy-preserving access of outsourced data via oblivious RAM simulation. In *ICALP*, pages 576–587. Springer, 2011.

[GMOT12] Michael T Goodrich, Michael Mitzenmacher, Olga Ohrimenko, and Roberto Tamassia. Privacy-preserving group data access via stateless oblivious RAM simulation. In *SODA*, pages 157–167. SIAM, 2012.

[KM10] Achim Klenke and Lutz Mattner. Stochastic ordering of classical discrete distributions. *Advances in Applied probability*, 42(2):392–410, 2010.

[KMW09] Adam Kirsch, Michael Mitzenmacher, and Udi Wieder. More robust hashing: Cuckoo hashing with a stash. *SIAM Journal on Computing*, 39(4):1543–1561, 2009.

[Pit98] Jim Pitman. Enumerations of trees and forests related to branching processes and random walks. *Microsurveys in discrete probability*, 41:163–180, 1998.

[PR01] Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. In *ESA*, pages 121–133. Springer, 2001.

[PR10] Benny Pinkas and Tzachy Reinman. Oblivious RAM revisited. In *CRYPTO*, pages 502–519. Springer, 2010.