SprayLists: Provably Correct Approximate Priority Queues in a Shared Memory Setting

Jerry Li

1 Introduction

An important data structure well understood in sequential settings is the priority queue, which supports the following three operations: $insert(x)$, which inserts an item with priority x, $delete(x)$, which deletes the item with priority x, and $deleteMin$, which deletes and returns the item with lowest priority.[1] Fibonacci heaps [4] support $deleteMin$ and $delete$ in amortized logarithmic time, and $insert$ in constant time, running times which are close to the information theoretic lower bound.

It is known that under the "wall clock" model of timing in the distributed setting, even queues have a linear lower bound for some process ([3]). Thus, while there has been a lot of work on provably correct priority queues in the distributed setting (see [2] for a survey), none of them have low collision rates, and indeed, it is straightforward to give executions for which the above problem must occur. However, enforcing the strict priority queue property in a shared memory setting is somewhat silly, as if there are p processes, and all of them simultaneously call $deleteMin$, then one process must obtain the element with the pth lowest priority at the time of the call anyways. This motivates us to instead consider *approximate* priority queues, in which a $deleteMin$ operation is only required to return anything amongst the $\tilde{O}(p)$ elements with lowest priority. A similar data structure was implemented by [1] in the sequential setting, where to get around a similar information theoretic lower bound, they allow a small fraction of elements to become corrupted in a Binomial-tree-like structure, which allows them to obtain $O(\log \frac{1}{\epsilon})$ running times, if an ϵ fraction of inputs are allowed to become corrupted.

We present a possible implementation of such an approximate priority queue that breaks the linear lower bound, using a data structure called *skip list* ([8], [7], [10], [9]). We alter $deleteMin$ to use a new operation called *spraying*. We demonstrate the correctness of this operation, and that in a restricted model of asynchronicity, spraying, and thus $deleteMin$, takes polylogarithmic time.

The rest of this report is laid out as follows. In section 2 we describe the model and skip lists in detail. In section 3 we motivate and describe our spraying algorithms, as well as our model of asynchronicity. In sections 4, 5, 6 we prove the correctness and efficiency of spraying, and finally in section 7 we give directions of for future work.

1.1 Probabilistic Tools

We will need the two following, important inequalities.

Theorem 1.1 (The Union Bound)**.** *If for $1 \leq i \leq k$, U_i is an event that occurs with probability p_i, then*

$$\Pr\left(\bigcup_{i=1}^{k} U_i\right) \leq \sum_{i=1}^{k} p_i.$$

Theorem 1.2 (Chernoff Tail Bounds)**.** *Let X_i be independent random variables which take value in $[0, 1]$, for $1 \leq i \leq k$, and let $X = \sum X_i$ and $\mu = \mathbb{E}(X)$. Then for $\mu' \geq \mu$, and any $\delta \in [0, 1]$,*

$$\Pr(X \geq (1 + \delta)\mu') \leq e^{-\mu'\delta^2/3},$$

and for any $\mu' \leq \mu$ and any $\delta \in [0, 1]$,

$$\Pr(X \leq (1 - \delta)\mu') \leq e^{-\mu'\delta^2/2}.$$

[1] Often $decreaseKey$ is used instead of $delete$, but since a $decreaseKey$ can be implemented by a $delete$ followed by an $insert$, the two models are equivalent

2 Background

2.1 The Asynchronous Shared Memory Model

We describe this model somewhat informally. See [6] for a more thorough treatment. We assume that p processes are operating on a shared set of registers, and in an atomic step, can perform a single *read* or *write* operation on that register, and assume each register can hold a constant number of integers and pointers to other registers. Each process may have additional local state, but a memory address in shared memory refers to the same location no matter which process accesses it. A scheduler decides at each timestep which process can perform a single operation on a register in the shared memory. Note that the scheduler may schedule a single process to take many steps in a row. For any fixed execution, we say that the time that a single processor takes during that execution is the number of steps that the process takes during the execution before terminating. We treat this scheduler as adversarial. That is, given knowledge of the current state of the system, it will try to schedule processes so that some process (or many processes) takes many steps to finish.

It is known that, assuming no process failures, we can implement locks on the shared memory using only read write registers, using algorithms such as Lamport's Bakery algorithm ([5]) which guarantee a host of nice liveness and safety conditions which we won't mention. We will simply assume that locking is an additional primitive operation.

The timing model we will use here is roughly based on the stall, or wall clock, model ([3]). Roughly, if we imagine that all processes run simultaneously and at approximately the same speed, then the time spent by a process is the time it takes waiting for locks, and the time it takes processing when it has the appropriate locks.

2.2 Skip Lists

Skip lists, introduced by Bill Pugh ([8]), are probabilistic data structures which support $insert(x)$, $delete(x)$, and $search(x)$ in expected $O(\log n)$ time for $x \in \mathbb{N}$, if there are n elements in the skip list. We develop and analyze skip lists in sequential and shared memory models below.

If the set of inputs is static, then one way to support logarithmic time *search* would be to maintain the elements in a sorted linked list, where every other element also has a pointer to the node two steps in front of it, and every fourth element, in addition to the pointers it already has, also has a pointer to the node four steps in front of it, and so on, so that every (2^i)th element has a pointer to the node (2^i) steps in front of it. We say a pointer is a height k if it skips 2^{k-1} nodes, and we define the height of an element to be the maximum height of any pointer coming out of that node. We also maintain a special element 0 which has links to the first element of each height, and a `NIL`, so that the last pointer at each height points to `NIL`. Notice that if this structure has n elements, the maximum height of any pointer is $\log n$. Call such a skip list with such a structure a *perfect* skip list. Given such a structure, it is straightforward to conduct $search(x)$ in $O(\log n)$ time, namely, starting at the top level pointers, scan until you find the largest element less than x, then descend a height and scan using those pointers and repeat until at the height 1 pointers, at which point either x is found or not, and return the appropriate value.

It is however too difficult to maintain a perfect skip list structure under inserts and deletes, however, by randomizing we can maintain a structure with guarantees which are close enough to give expected logarithmic time search. As above, each element is represented by a node which are sorted by increasing element size, whose height is chosen randomly when the node is inserted. A node of height k has k forward pointers, labelled $1, \ldots, k$, and we enforce that the ith pointer of any node is to the next largest node in the ordering with height at least i. We still maintain a special 0 node which has pointers to the first element of each height, with each pointer labeled with the height of the element it links to, and a `NIL` node at the end, and the last node of every height k has its level k pointer point to `NIL`. A sample skip list is given in Figure 1.

Searching with this structure then can be done as above, starting from the top and doing the binary search-like procedure as in the perfect skip list. To perform $insert(x)$, we first randomly choose a height h so that with probability 2^i the height is i (this can be done by repeatedly flipping a coin and setting the height to be the number of flips before we see a tails plus one). Then we perform a $search(x)$ to find where it belongs in the skip list, and for each level $k \leq h$, we find the last element u_k of height at least k before x

in the skip list, and change its height k pointer to x, and set x's level k pointer to what u_k's height k pointer pointed to.

We can then think of each height k element as belonging to k different linked lists, where the ith list comprises of all nodes of height at least i and all height i pointers. This is clearly illustrated in the figure. To perform $delete(x)$, we perform $search(x)$ and if it has height h, we simply remove it from all of the linked lists of level $k \leq h$.

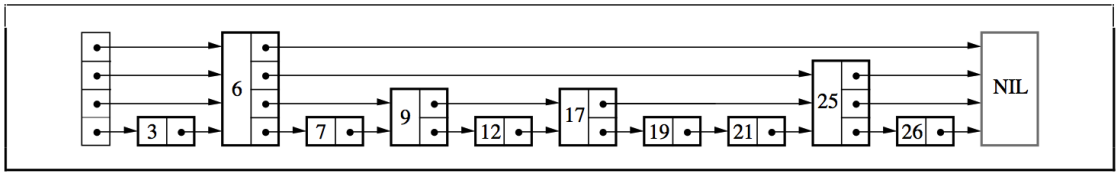

Figure 1: A sample skip list, taken from [8]

Proposition 2.1. *With probability* $1 - \frac{1}{k^2}$*, the maximum height of any element in a skip list within the first* k *elements of the skip list is* $3 \log k$*.*

Proof. The probability that any element has height at least $3 \log k$ is $2^{-(3 \log k)} = \frac{1}{k^3}$; a union bound over the first k elements then gives the desired result. □

This demonstrates that we can assume that the maximum height of any element is $O(\log n)$ in a skip list with n elements; hence if we can demonstrate that $search(x)$ takes $O(\log n)$ time as well in expectation, then $insert(x)$ and $delete(x)$ will also take $O(\log n)$ time in expectation as well, since they consist of a single search operation, plus a constant number of operations for each height.

Proposition 2.2. *In expectation,* $search(x)$ *takes* $O(\log n)$ *time on a skip list with* n *elements.*

Proof. Consider the amount of time spent by the algorithm searching at height k, for any k. For each step at height k, it must be that the element we step to has height exactly k, since otherwise we would have reached that element in the search at a higher height. Since an element which has height at least k has height exactly k with probability $1/2$, we conclude that since the number of steps we take at any height is then distributed according to the number of times a fair coin must be flipped until an experiment fails, we take a constant number of steps at each level in expectation. Since by Lemma 2.1 there are $O(\log n)$ levels, we conclude that searching takes $O(\log n)$ time in expectation. □

Given these operations on a skip list, it is trivial to implement a priority queue which supports expected logarithmic time $insert, delete$, and expected constant time $deleteMin$. The first two operations are already implemented, and $deleteMin$ can be implemented by simply removing the front element of the skip list. Since removing a single element takes constant time since in expectation any element has height 2, it takes constant time.

2.3 Skip lists in a shared memory setting

Notice that each skip list operation is highly decentralized, that is, although we may have many simultaneous reads, there is no one centralized location at which many writes occur, as compared to a structure such as a binary tree, where if a remove happened at the root, many processes would try to concurrently access the root, and thus would ultimately fail. We will limit ourselves here to mention that [7] demonstrates that skip lists can be maintained in a concurrent setting, and that as long as lock contention of processes is low (i.e., few processes attempt to simultaneously remove the same item, and few processes attempt to simultaneously insert nearby items), search, insertion, deletion operations can be done in a constant number of lock accesses. Unfortunately, in a fully asynchronous model of computation, an adaptive scheduler, which can see the state of the skip list and the coins already flipped before scheduling the next event, can force search over the list to never terminate. Thus we will instead work in a restricted model of asynchronicity (see below).

3 The Spraying Algorithm

3.1 Spraying

The basic idea of spraying is straightforward: when performing $deleteMin$, to avoid collision it is unavoidable that we return something amongst the $\tilde{O}(p)$ items with lowest priority. One natural attempt would be to simply choose one of the $\tilde{O}(p)$ items with lowest priority uniformly at random, however, upon further reflection it should be apparent that choosing uniformly at random from the first $\tilde{O}(p)$ items in the shared-memory setting is hard (at least, I can't do it) while avoiding collisions.

Thus, instead, we use the randomness of the skip list to emulate such a procedure, by performing a random walk on the skip list. We call our random walk procedure *spraying*. To perform a spray, a process first starts at the front of the skip list, and at some initial height h. At each level, the process randomly chooses to take some number of steps at that level, then goes down some number of levels, and repeats until it is at the bottom of the skip list, at which point it returns the element it has landed on.

It remains to decide what parameters we use for our spraying procedure, namely, at which height we start, what distribution to use to choose how far we go at each level, and how many levels to skip. We wish for sprays to have four high-level properties:

1. Sprays take polylogarithmic time
2. Sprays don't go too long.
3. Sprays don't collide too much.
4. Sprays are distributed near uniformly on a large interval.

Call a skip list on which no removes have occurred due to sprays a *clean* skip list.

Consider what we must do on a clean skip list. We must start at some level $h = \log p + K$ for some constant K, because at each level the spray can take at most polylogarithmic steps, since we want the whole procedure to take polylogarithmic time (also otherwise it'd go to far at high levels, and would have too much collision at low levels). Thus, if we started at some level $h < \alpha \log p$ for some $\alpha < 1$ then by a straightforward calculation we will spray onto at most the first $\tilde{o}(p^\alpha)$ elements, which implies by the pigeonhole principle more than a constant fraction of all processes must fail if all processes p simultaneously spray, which is bad. Alternatively, if we started at some level $h > \beta \log p$ for some $\beta > 1$ then even one step at the top level will go to an element which in expectation is the p^βth item from the front of the list, which is too far, since we only want to spray amongst the first $\tilde{O}(p)$ items. Thus our choice of $h = \log p$ as a starting height is justified.

We next seek to determine the distribution. Intuitively, at the lowest level we need to choose uniformly between $[1, c]$ for some $c \geq \log p$ since in expectation, amongst the first $\tilde{O}(p)$ elements in the skip list, there will be at least a constant number of structures which consist of a group of $\log p + O(1)$ consecutive items, each with height 1. We can only reach items near the end of these structures by spraying approximately $\log p$ steps at height 1, and we should prefer to choose all of these items uniformly, since they are all within the first $\tilde{O}(p)$ elements. Given this, and since we don't want sprays to take more than polylogarithmic time, we know that c should be polylogarithmic in p. While this logic doesn't totally apply at high levels, it turns out that choosing from $[1, c]$ uniformly at random for some c which is polylogarithmic in p suffices.

The choice of how many levels to skip in one go is the most artificial of three. The natural choice of going down one level at a time, combined with a choice of $c = \log p$, yields `PerfectSpray`. Unfortunately, this doesn't give us good enough concentration bounds for the techniques we wish to apply, and we will be primarily using `PerfectSpray` and the analysis of its performance on perfect skip lists as motivation for the general proof techniques we apply on general skip lists. Instead we choose to skip $\log \log p$ levels at a time, and $c = f(p) \log^3 p$ for any function f which increases to infinity.[2] These parameters give us `BasicSpray`. We give the pseudocode for `BasicSpray` below, the code for `PerfectSpray` is very similar and omitted.

However, there is one more problem, namely, we will not hit elements at the front of the list, which is bad, given that we want a priority queue. Even if we allow sprays to stay where they by choosing uniformly from $[0, c]$ instead of $[1, c]$, such a modified `PerfectSpray` would have probability $(\log p + 1)^{-\log p} \leq p^{-\log \log p}$

[2]We need this f because we want $p^{-c/\log^3 p}$ to grow slower than any polynomial in p; we can think of f basically as a large constant

Algorithm 3.1 `BasicSpray`

$x := 0$.
$k := \log p / \log\log p$
while $k > 0$ **do**
  Choose $s \leftarrow Unif[1, f(p)\log^3 p]$
  Walk x by s steps at level k
  $k \leftarrow k - \log\log p$
end while
Choose $s \leftarrow Unif[1, f(p)\log^3 p]$
Walk x by s steps at level 1
Return x

of hitting the first element, and `BasicSpray` would have a similarly small probability of hitting the first element. To get around this, we simply "pad" the skip list, namely, add $K(p)$ dummy entries in the front of the skip list, where $K(p)$ is some number we determine later. If a spray hits the first $K(p)$ entries, it is said to fail. We will need to choose $K(p)$ so that with at most constant probability sprays will land in the first $K(p)$ entries of the skip list, and that in some reasonably large interval after the first $K(p)$ entries, every element is hit with probability $\tilde{O}(p^{-1})$. By adding this behavior and adding retries for failed sprays to `BasicSpray` we obtain our full spraying algorithm, `FullSpray`, given below.

Algorithm 3.2 `FullSpray`

$K(p) := pf(p)\log^2 p$
while true **do**
  Run `BasicSpray`
  if `BasicSpray` finishes **then**
    Let x be the value returned
    if x is not in the first $K(p)$ elements **then**
      Remove x
    end if
  end if
end while

Notation We introduce here a few terms that will be useful in the analysis of `BasicSpray`.

- We call the element with the pth lowest priority in the skip list the pth element in the skip list.
- We call the part of the spray consisting of the walk at level $k\log\log p$ the *kth level* of the spray, for $k \geq 1$, and the part of the spray consisting of the walk at level 1 the *0th level* of the spray.
- We let E_k denote the expected length the kth level travels if it travels $\log^3 p$ steps, and let $\ell_p = \log p / \log\log p$, for $k \geq 1$ and $E_0 = \log^3 p$ denote the length a spray at height 1 travels if it travels the full distance. We know that $E_k = 2^{k\log\log p - 1} f(p)\log^3 p$ on a clean skip list.
- We will say that an iteration of `BasicSpray` *succeeds* if it finishes and returns a non-dummy element. Under the simplifying assumptions (see next section), if we can show that any given iteration of `BasicSpray` succeeds with at least a constant probability, then with high probability `FullSpray` will succeed in a constant number of calls to `BasicSpray`, thus condition (1) will be satisfied.
- Finally, we say a probability is *negligible* if it grows slower than any inverse polynomial. We will often argue that the structure of the skip list follows some property, or some part of the algorithm will work, except with negligible probability. Since we envision that applications of this data-structure will be in algorithms where each process makes at most polynomially many steps, it follows that the number of elements in the skip list is some polynomial in p, and thus the probability that the skip list doesn't follow this property, or some part of the algorithm will fail, is still negligible in n.

3.2 Simplifying Assumptions

Unfortunately I could not prove the results I wanted to in full generality in a fully asynchronous shared memory model, so instead, we make a few assumptions, which I will often assume and forget to mention in the rest of the paper.

A Simpler Model of Asynchronicity We consider a weaker, less asynchronous model of computation, in which sprays can be scheduled to occur in rounds. We assume that the skip list is already populated. Then, in the ith round of execution, we can specify any subset of processes on which to run `BasicSpray` algorithms with, and after all these sprays run (in any asynchronous order), we can perform *delete* operations at any element a spray landed. We penalize every process a $f(p)\log^3 p$ factor for each round of execution for spraying, since either that process is spraying during the execution or forced to wait for the next round of execution. We also penalize each process a further polylogarithmic factor for each *delete* operation it does. This requires a bit of justification, however, since spraying simulates uniform choice, and choosing p elements uniformly at random out of $O(p\log^3 p)$ buckets results in relatively few elements nearby, we can demonstrate that if processes perform *delete* operations somewhat synchronously, they will finish in expected constant time, and worst case polylogarithmic time. We purposefully ignore *insert* operations, because unfortunately, if many inserts of nearby items occur, then an adversarial scheduler can very easily cause a long chain of processes waiting for locks. However, if most real-world applications, where *insert* operations look somewhat random, then by the same omitted analysis *insert* operations take polylogarithmic time. Notice that if indeed at most a constant fraction of `BasicSprays` fail during any round (i.e. a constant number collide), then in this model of computation, in logarithmically many rounds, p `FullSpray` operations which started simultaneously will finish, and thus `FullSpray` still takes poly-logarithmic time per process.

Dirty Skip Lists are Clean We will assume that *even after any number of sprays, the structure of the skip list is distributed as the structure of a clean skip list.* This is not clear, and not true for general remove procedures. For instance, consider the naive remove procedure in which we remove the first element of height $\log p$ that we find. After k such removes, the structure of the skip list has dramatically altered: in particular, the first element of height $\log p$ has now distance roughly $kn\log p$ from 0 as opposed to distance $n\log p$ as before.

However, this does not seem to happen with sprays. By what we prove below it is clear that a `BasicSpray` on a clean skip list emulates (up to some small, inverse polynomial error) uniformly sampling from an interval in the skip list. Since uniform sampling would not change the distribution of the structure of the skip list, we expect that the distribution after one `BasicSpray` of the list will not differ too much from a clean skip list; from here an inductive argument should suffice to argue that if we do at most polynomially (for some small polynomial) many operations on the skip list, the error does not accumulate enough to change the analysis. If we can somehow then clean the skip list by calculating new heights for every element in the skip list every polynomially many steps (possibly deciding to do so in a randomized fashion), then intuitively between cleans we should have the desired distributional properties of the skip list structure, and the cost of the cleaning can be amortized away. However, I didn't have time to figure out the details of this, and it felt like it detracted from the point of the report, which is to demonstrate the nice combinatorial and decentralized properties of spray operations.

The idea is that now, in this semi-realistic model of priority queues, the question of the effectiveness and efficiency of spraying as an approximate $deleteMin$ operation becomes an entirely combinatorial question; namely, up to some padding of the skip list, do sprays choose near uniformly from the first $\tilde{O}(p)$ elements in a skip list, on a random skip list? Since this is not only an interesting question in its own right, as this produces a decentralized way to near-uniformly sample from a set of points from the front of a priority queue without knowing exactly what elements are in that set, but only the size of the set, but also because the model described above seems to model reality fairly well,[3] we seek to answer the question posed in the rest of this report.

[3]Also because anything more general just doesn't seem to work theoretically...

4 Sprays don't go too long

Verifying property (2) is fairly straightforward (under our simplifying assumption). It follows almost trivially from the following lemma.

Lemma 4.1. *For $k \leq \log p$, the kth level of spray goes $f(p)\log^3 p$ steps starting at any point will with probability at most $p^{O(\alpha^2 \log^2 p)}$ go more than $(1+\alpha)2^{k-1}f(p)\log^3 p$ distance, for any bounded $\alpha > 0$.*

Proof. Fix some element x. Let X_T be the number of elements with height at least k that we see in a random walk of T steps starting at the xth element. We know that $\mathbb{E}(X_T) = 2^{-k+1}T$. Choose $T = (1+\alpha)2^{k-1}f(p)\log^3 p$. Then by a Chernoff bound,

$$\Pr(X_T \leq f(p)\log^3 p) \leq e^{-O((\alpha/(1+\alpha))^2 f(p)\log^3 p)} = p^{-O(\alpha^2 f(p)\log^2 p)}$$

so if we take T steps at the bottom level we will with high probability hit enough elements of the necessary height, which implies that a spray at that height will not go more than that distance. □

This immediately gives us the following.

Theorem 4.2. *Let $\sigma(p) = \log p/(\log p - 1)$. For any fixed α, if two sprays at level k are separated by distance $\alpha\sigma(p)E_k$, then except with probability $p^{-O(\alpha^2 f(p)\log^2 p)}$, they will end up at different positions in the spray list.*

Proof. Consider the spray with at each level sprays the maximal number of steps. Clearly this spray goes the farthest of any spray. Let x_k denote the element at which spray ends up after it finishes its kth level for $0 \leq k \leq \ell_p$ and $x_{\ell_p+1} = 0$, and let d_k be the distance that the spray travels at level k. For any $k > 0$, by Lemma 4.1, we know that

$$\begin{aligned}\Pr(d_k > (1+\alpha)E_k) &= \Pr(\text{The } k\text{th level of the spray starting at } x_{k+1} \text{ goes further than } (1+\alpha)E_k \text{ steps}) \\ &\leq p^{-O(\alpha^2 f(p)\log^2 p)},\end{aligned}$$

so by the union bound, $\Pr(\exists k : d_k > (1+\alpha)E_k) \leq p^{-O(\alpha^2 f(p)\log^2 p)}$ as well.

Thus, this worst case spray will, except with negligible probability, will not go more than

$$\sum_{i=0}^{k/\log\log p} d_k \leq (1+\alpha) \sum_{i=0}^{k/\log\log p} E_i = \alpha\frac{1}{2}\frac{2^k \log p - 1}{\log p - 1} f(p)\log^3 p \leq \alpha\frac{\log p}{\log p - 1}2^{k-1}f(p)\log^3 p = \alpha\sigma(p)E_k$$

steps, as claimed. □

Corollary 4.3. *For any fixed α, sprays can go farther than $\alpha\sigma(p)pf(p)\log^3 p$ with probability at most $p^{O(\alpha^2 f(p)\log^2 p)}$.*

5 An upper bound on the probability sprays hit an element

5.1 Analysis of `PerfectSpray` on a perfect skip list

We first give the following, motivating result. While the proof technique cannot apply to general skip lists, the technique itself is interesting and provides nice intuition which motivated the proof for `BasicSpray` on random skip lists.

Theorem 5.1. *For any x in a perfect skip list, at most a $\frac{1}{p}$ fraction of* `PerfectSpray` *sprays will hit x.*

Proof. Notice that any spray on a perfect spray list which sprays a_i at level i lands at element $\sum_{i=1}^{k} a_i 2^{i-1}$. Thus, for any x, it suffices to count the number of $\log p$ tuples $(a_1, \ldots, a_{\log p})$ so that $0 \leq a_i \leq \log p$ for all i and $\sum_{i=0}^{k} a_i 2^i = x$. Note this corresponds to the sum where the ith a_i is bit shifted $i-1$ times to the left. For each i, let a_{ij} denote the jth order bit of a_i in the binary expansion of a_i, and let x_j denote the jth order bit of x in its binary expansion. For any k, call a k-tuple $(a_1, \ldots, a_k)$ *viable* if there exists some $\log p$

tuple $(a_1, \ldots, a_k, a_{k+1}, \ldots, a_{\log p})$ so that $\sum_{i=1}^{k} a_i 2^{i-1} = x$. Notice that for (a_1) to be viable, it must be that $a_{10} = x_0$. Inductively, for all $k \leq \log p$, conditioned on choice of $a_1, \ldots, a_{k-1}$, the only viable a_k must have $a_{k0} + \sum_{i=1}^{k-1} a_{i(k-i)} + c_k = x_k$, where c_k is a carry bit which depends only on a_i for $i < k$, which implies that at most half of the a_k are viable, conditioned on choice of the previous a_i. Thus,

$$\begin{aligned}\Pr(\text{spray hits } x) &= \Pr((a_1) \text{ viable}) \prod_{k=2}^{\log p} \Pr((a_1, \ldots, a_k) \text{ viable} | (a_1, \ldots, a_{k-1}) \text{ viable}) \\ &\leq \left(\frac{1}{2}\right)^{\log p} = \frac{1}{p}\end{aligned}$$

as claimed. □

Of course, this argument completely falls apart when applied to random skip lists, as it is incredibly fragile. However, the ideas of "filtering" paths based on levels, and the backwards nature of the analysis, where we proceed from the last step to the first step, can be seen in the analysis of `BasicSpray` below.

5.2 Analysis of `BasicSpray` on a random clean skip list

Fix any interval $I = [a, b]$ for $a, b \in \mathbb{N}$ and $a \leq b$. In expectation, there are $(b-a)2^{k-1}$ elements in I with height at least k in the skip list; the following lemma simply states that with high probability, we do not deviate far from the expectation.

Lemma 5.2. *For any fixed b and any height i, there are at most $(1+\beta)(k+1)2^{i-1}$ items between the $(b-k)$th item and the bth item with height at least i on a clean skip list, with probability at least $1 - e^{-\beta^2(k+1)/3}$, and there are at least $(1-\beta)(k+1)2^{i-1}$ items, with probability at least $1 - e^{-\beta^2(k+1)/2}$.*

Proof. Let X_i be the random variable which is 1 if the $(b-k+i)$th item has a bucket of height at least i, and 0 otherwise, and let $X = \sum_{i=0}^{k} X_i$. Since $\mathbb{E}(X_i) = 2^{-i+1}$ we have $\mathbb{E}(X) = (k+1)2^{-i+1}$ and thus by a Chernoff bound

$$\Pr(X \geq \alpha(k+1)2^{i-1} \log^3 p) \leq \exp\left(\frac{-\beta^2 2^{-i+1}(k+1)}{3}\right) \leq e^{-\beta^2(k+1)/3}.$$

The other direction follows from a Chernoff bound applied in the other direction. □

Theorem 5.3. *There is a constant C so that for any fixed x, except with negligible probability, the fraction of sprays with land at x is at most $C\frac{1}{pf(p)\log^3 p}$.*

Proof. For each $k \geq 1$, define $I_k = [x - \alpha\sigma(p)E_{k-1} + 1, x]$, and let t_k denote the number of elements in I_k with height $k \log\log p$. By Theorem 4.2, after a spray has finished walking on the kth level, if it is outside I_k, it will hit x with probability at most $p^{-O(\alpha^2 f(p)\log^2 p)}$. Thus, by a union bound, we conclude that if for any k, the spray is outside I_k after spraying at level k, then with at most negligible probability it will hit x, so we may assume that a spray will only hit x if for each k it is in I_k after spraying at level k. But the probability that the kth level of any spray lands in I_k is at most $t_k/(f(p)\log^3 p)$ as we choose how far to spray uniformly at random. By Lemma 5.2 we know that except with probability $e^{-\beta^2(E_k+1)/3} = p^{-O(\beta^2 f(p)\log^2 p)}$, I_k contains at most $(1+\alpha)(1+\beta)\sigma(p)E_{k-1}2^{k\log\log p - 1} = (1+\alpha)(1+\beta)\frac{1}{\log p}\sigma(p)f(p)\log^3 p$ elements with height at least $k\log\log p$, hence $t_k/(f(p)\log^3 p) \leq (1+\alpha)(1+\beta)\sigma(p)\frac{1}{\log p}$ except with negligible probability, for any fixed k. By a union bound over all $\log p/\log\log p$ levels, this holds except with negligible probability for all levels. If we choose $\alpha = \beta$, by a further union bound, the fraction of sprays that land in I_1 after the first k level sprays is at most

$$\left((1+\alpha)^2\sigma(p)\frac{1}{\log p}\right)^{\log p/\log\log p}$$

except with probability $p^{-O(\alpha^2 f(p)\log^2 p)}$. If we choose $(1+\alpha)^2 = (1+\frac{1}{\log p})$ so that $\alpha = \sigma(p)^{1/2} - 1$, since $(\log p)^{-\frac{\log p}{\log\log p}} = \frac{1}{p}$, and

$$\lim_{p\to\infty} \sigma(p)^{\log p/\log\log p} = \lim_{p\to\infty} \left(1 - \frac{1}{\log p}\right)^{\log p/\log\log p} = 1$$

by ugly calculus, and

$$\lim_{p\to\infty} \alpha \log^2 p = \lim_{p\to\infty} \left(\sqrt{\frac{\log p}{\log p - 1}} - 1\right)^2 \log^2 p = \frac{1}{4},$$

by (less) ugly calculus as well, we conclude that except with probability $p^{-O(f(p))}$, the fraction of sprays that land in I_1 is at most Cp^{-1} for some constant C which can be made arbitrarily close to one as p grows large. Since conditioned on landing in I_1, a spray must further land at x exactly, which happens with at most a $\frac{1}{f(p)\log^3 p}$ fraction of these sprays, we conclude that a total of $C\frac{1}{pf(p)\log^3 p}$ sprays land at x, as claimed.

□

Importantly, this demonstrates (under our simplifying assumption) that two `BasicSpray` sprays are unlikely to collide.

6 A lower bound on the probability sprays hit an element

We wish to lower bound the probability of hitting the xth smallest item, for x in some reasonable interval which we will define later. The technique is ultimately to make the estimates made above tight. Ultimately, the technique we use to prove Theorem 5.3 is to define some critical region for each level of some fixed length, and argue that (1) sprays hitting x must hit the critical region at each level, and (2) sprays in a critical region at some level have a bounded probability of entering the next critical region. If we can redefine this notion of critical region (for x in some appropriate interval) so that a tighter version of (2) holds, namely, that every spray in a critical region at level k has the can enter every element of the critical region at level $k-1$, then intuitively, all the inequalities presented above are tight. Below, we formalize that intuition.

Fix some x. Let $I_0 = [x - \log^3 p, x - 1]$. Then if $x - \log^3 p \geq 0$ we know that if a spray lands in a bucket in I_0 at height 1 it has a $\log^3 p$ probability of spraying to x. Let t_0 be the number of buckets in I_0 of height $\log\log p$. Inductively, for all $k \leq \ell_p - 1$, given an interval $I_{k-1} = [a_{k-1}, b_{k-1}]$ so that $a_{k-1} \geq 0$ and t_{k-1} which is the number of elements in I_{k-1} of height $k\log\log p$, then notice that there are, except with probability $p^{-O(\beta^2\log^2 p)}$, at most $f(p)\log^3 p$ elements in $[b_{k-1} - \frac{1}{1+\beta}E_k, b_{k-1}]$ with height $k\log\log p$, by Lemma 5.2. Thus let $a_k = b_{k-1} - \frac{1}{1+\beta}E_k$ and $b_k = a_{k-1} - 1$, and let t_k be the number of elements in $I_k = [a_k, b_k]$ of height $(k+1)\log\log p$. Assume for now that $a_k \geq 0$. But then, any spray which lands in any bucket in I_k after spraying at level $k+1$ can reach every element of I_{k-1} of height $k\log\log p$ by a spraying at level k, since there are at most $f(p)\log^3 p$ elements of height $k\log\log p$ in the interval $[a_k, b_{k-1}]$ and $b_k < a_{k-1}$. Thus, of the sprays that land in I_k after spraying at level $k+1$, a $t_k/(f(p)\log^3 p)$ fraction will land in I_{k-1}. Thus it suffices to compute what each t_k is.

Lemma 6.1. *Let $s_k = b_k - a_k + 1$ be the number of elements in I_k. For all $k \geq 2$, we have*

$$\left(\gamma_0 - \gamma_1\frac{1}{\log p}\right)E_k \leq s_k \leq \left(\gamma_0 + \gamma_1\frac{1}{\log^2 p}\right)E_k$$

with

$$\begin{aligned}\gamma_0 &= \frac{\log p}{(\beta+1)(\log p+1)}\\ \gamma_1 &= \frac{\beta\log p + \beta + 1}{(\beta+1)(\log p+1)}.\end{aligned}$$

Proof. Define ξ_k to be the quantity so that $s_k = \xi_k E_k$. Clearly $\xi_0 = 1$, and inductively,

$$\begin{aligned} s_k &= \frac{1}{1+\beta} E_k - s_{k-1} \\ &= \left(\frac{1}{1+\beta} - \frac{\xi_{k-1}}{\log p} \right) E_k \end{aligned}$$

so

$$\xi_k = \frac{1}{1+\beta} - \frac{1}{\log p} \xi_{k-1}.$$

Homogenizing gives us a second order homogenous recurrence relationship

$$\xi_k = \left(1 - \frac{1}{\log p} \right) \xi_{k-1} + \frac{1}{\log p} \xi_{k-2}$$

with initial conditions $\xi_0 = 1$ and $\xi_1 = \frac{1}{1+\beta} - \frac{1}{\log p}$. Solving gives us that

$$\xi_k = \gamma_0 + \gamma_1 \left(-\frac{1}{\log p} \right)^k.$$

Notice that $\xi_{2k+2} \leq \xi_{2k}$ and $\xi_{2k+3} \geq \xi_{2k+1}$ and moreover, $\xi_{2k+1} \leq \xi_{2k'}$ for any k, k'. Thus for $k \geq 2$ the maximum of ξ_k occurs at $k = 2$, and the minimum occurs at $k = 1$. Substituting these quantities in gives us the desired results. □

Theorem 6.2. *There are constants $K, C' > 1$ which for p sufficiently large can be chosen arbitrarily close to 1 so that for all $x \in [Kp \log^2 p, \frac{1}{1+\beta} pf(p) \log^3 p]$, except with negligible probability, at least a $C' \frac{1}{p \log^3 p}$ fraction of* `BasicSpray` *sprays land at x.*

Proof. The arguments above are precise, as long as (1) every element of I_{k-1} can be reached by a spray from 0 at level k and each $a_k \geq 0$. By Lemma 5.2, condition (2) holds except with probability $p^{-O(\beta^2 f(p) \log^2 p)}$. Moreover, each $a_k \geq 0$ is equivalent to the condition that $x \geq \log^3 p + \sum_{k=1}^{\ell_p - 1} s_k$, but by Lemma 6.1, we have that (except with probability $p^{-O(\beta^2 f(p) \log^2 p)}$) that

$$\sum_{k=1}^{\ell_p - 1} s_k \leq \left(\gamma_0 + \gamma_1 \frac{1}{\log^2 p} \right) \left(\sum_{k=1}^{\ell_p - 1} E_k \right).$$

For the choice of $\beta = \frac{1}{\log p}$, the first term in this product can be made arbitrarily close to one for p sufficiently large, and thus for some constant D, we have that except with negligible probability,

$$\sum_{k=1}^{\ell_p - 1} s_k \leq D f(p) \log^3 p \left(\sum_{k=1}^{\ell_p - 1} 2^{k \log \log p - 1} \right) = O(pf(p) \log^2 p),$$

in fact, if we were to do the calculations carefully, the constant term the big-Oh hides can be made arbitrarily close to one for p sufficiently large.

By Lemmas 6.1 and 5.2 and by a union bound, we have that except with probability $p^{-O(\beta^2 f(p) \log^2 p)}$,

$$t_k \geq \frac{1}{\log p} \left(\gamma_0 - \gamma_1 \frac{1}{\log p} \right) f(p) \log^3 p,$$

for all k. Thus by the logic above, if we let E_k denote the event that the spray is in I_k right before spraying

at that level, we have

$$\begin{aligned}\Pr(\text{spray hits } x) &\geq \Pr(\text{spray hits } x|I_0)\left(\prod_{k=1}^{\ell_p-1}\Pr(I_{k-1}|I_k)\right)\Pr(I_{\ell_p-1}) \\ &\geq \frac{1}{\log^3 p}\prod_{k=0}^{\ell_p-1}\frac{t_k}{f(p)\log^3 p} \\ &\geq \frac{1}{\log^3 p}\left(\left(\gamma_0-\gamma_1\frac{1}{\log p}\right)\frac{1}{\log p}\right)^{\ell_p}.\end{aligned}$$

If we choose $\beta = \frac{1}{\log n}$, then by some maximally ugly calculus, one can show that

$$\lim_{p\to\infty}\left(\gamma_0-\gamma_1\frac{1}{\log p}\right)^{\ell_p} \to 1,$$

we conclude that a $C'\frac{1}{p\log^3 p}$ fraction of sprays hit x, for some constant C' which can be chosen arbitrarily close to 1 for sufficiently large p, except with negligible probability. □

This theorem demonstrates that except with probability roughly $O(\frac{\log^2 p}{p})$, we will land in this interval $I = [Kp\log^2 p, \frac{1}{1+\beta}pf(p)\log^3 p]$, and thus (under our simplifying assumption) `FullSpray` will not fail because of failing to travel sufficiently far except with very small probability. Moreover, along with Theorem 5.3, we get matching lower and upper bounds in this interval, which demonstrates our claim that in I, spraying approximates uniform selection up to a relatively small error, which intuitively justifies our simplifying assumption.

7 Conclusions

We have described a new data structure for approximate priority queues in a shared memory model of distributed computing, and proved its correctness and efficiency in a restricted model of computation. Our new algorithm, in this limited model, runs in poly-logarithmic time, and is guaranteed, except with negligible probability, to return elements from within the $\tilde{O}(p)$ elements with smallest priority, moreover, simulates up to a small error, uniform sampling from the skip list from the first roughly $O(pf(p)\log^3 p)$ elements of the priority queue, while being relatively oblivious to the state of the priority queue.

7.1 Future Work

Analyzing Dirty Skip Lists It should be possible remove the first simplifying assumption, that skip lists that have been sprayed upon still are structured like clean skip lists, through the amortized "cleaning" operation suggested in Section 3.2 should suffice. However, the analysis of the dirty skip lists seems nontrivial, and at the very least, very ugly, although intuitive and well-motivated.

Analysis of `PerfectSpray` We conjecture that `PerfectSpray`, or some variant which still runs in $O(\log^2 p)$ time also has the same distributional properties as `BasicSpray` on random clean skip lists. Simulations of `PerfectSpray` sprays on random skip lists in MATLAB seem to suggest so, at least. As a first step, it should be possible to, with more care, demonstrate that in some interval near the front of a perfect skip list of size roughly $O(p\log^2 p)$, every element is hit with uniform probability.

7.2 Acknowledgments

The idea of spraying came from Dan Alistarh and Justin Kopinsky, and the author is grateful for their helpful feedback in the analysis of these sprays. The author would also like to thanks Nir Shavit for his supervision and insight throughout this research.

References

[1] Bernard Chazelle. The soft heap: an approximate priority queue with optimal error rate. *J. ACM*, 47(6):1012–1027, 2000.

[2] Kristijan Dragicevic and Daniel Bauer. A survey of concurrent priority queue algorithms. In *IPDPS*, pages 1–6, 2008.

[3] Faith Ellen, Danny Hendler, and Nir Shavit. On the inherent sequentiality of concurrent objects. *SIAM J. Comput.*, 41(3):519–536, 2012.

[4] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. In *FOCS*, pages 338–346, 1984.

[5] Leslie Lamport. A new solution of dijkstra's concurrent programming problem. *Commun. ACM*, 17(8):453–455, 1974.

[6] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[7] William Pugh. Concurrent maintenance of skip lists. *UMIACS Technical Report*.

[8] William Pugh. Skip lists: A probabilistic alternative to balanced trees. *Commun. ACM*, 33(6):668–676, 1990.

[9] Nir Shavit and Itay Lotan. Skiplist-based concurrent priority queues. In *IPDPS*, pages 263–268, 2000.

[10] Nir Shavit and Asaph Zemach. Scalable concurrent priority queue algorithms. In *PODC*, pages 113–122, 1999.