Multiple Secret Leaders

Authors here

August 23, 2023

1 Data Independent Priority Queue

Here we introduce a data independent priority queue. We will assume that there are no items in the queue with equal priority. In practice, we can break ties by adding a unique identifier to each item.

Algorithm 1 `MergeFill`

1: **Input:** $\texttt{DIPQ.Head}, \mathcal{P}_{\texttt{count}}$
2: $e_1, ..., e_\gamma \leftarrow \texttt{sort}(\texttt{DIPQ.Head}, \mathcal{P}_{\texttt{count}})$ where $\gamma = |\texttt{DIPQ.Head}| + |\mathcal{P}_{\texttt{count}}|$
3: $\texttt{DIPQ.Head}' \leftarrow \{e_1, ..., e_{\min(\sqrt{n},\gamma)}\}$
4: $\mathcal{P}'_{\texttt{count}} \leftarrow \{e_{\sqrt{n}+1}, ..., e_\gamma\}$
5: **return** $\texttt{DIPQ.Head}', \mathcal{P}'_{\texttt{count}}$

Algorithm 2 `Fill`

1: **Input:** $\mathcal{P}_{\texttt{count}}, \texttt{DIPQ.Stash}$
2: $e_1, ..., e_\gamma \leftarrow \mathcal{P}_{\texttt{count}} \bigcup \texttt{DIPQ.Stash}$ where $\gamma = |\mathcal{P}_{\texttt{count}}| + |\texttt{DIPQ.Stash}|$
3: $\mathcal{P}'_{\texttt{count}} \leftarrow \{e_1, ..., e_{\min(\sqrt{n},\gamma)}\}$
4: $\texttt{DIPQ.Stash}' \leftarrow \{e_{\min(\gamma,\sqrt{n}+1)}, ..., e_\gamma\}$
5: **return** $\mathcal{P}'_{\texttt{count}}, \texttt{DIPQ.Stash}'$

1.1 Invariants

Let $i, j \in \mathbb{N}$ be the total number of insertions and extractions respectively. Note that if $i - j \leq \sqrt{n}$ and in the prior $i+j$ operations, $i-j$ never exceeded $\sqrt{n}$, then the priority queue is trivially correct because all elements remain in the head which is itself a priority queue. Thus, we will assume that at some point in the sequence of insertions and extractions, we had $i - j > \sqrt{n}$.

Before specifying our invariants, we will add some notation. Let $\mathcal{G}$ ($\mathcal{G}$ for "good") be the largest set of smallest elements in the head. More formally, $\mathcal{G}$ is the largest subset of $\texttt{DIPQ.Head}$ such that $\forall e \in \texttt{DIPQ.Head}, e \notin \mathcal{G}$, $a < e, \forall a \in \mathcal{G}$. Note that $|\mathcal{G}| \leq \sqrt{n}$. Further, let g (the "good" element) be the smallest element in $\texttt{DIPQ}$ such that $\forall a \in \mathcal{G}, g > a$ and $\forall e \in \texttt{DIPQ} \setminus \mathcal{G}, g \leq e$. Notice that g is not in $\texttt{DIPQ.Head}$ because if it were, $\mathcal{G}$ would contain g.

We will show that the following invariants hold:

- The "good" element is not in the prior $\sqrt{n} - |\mathcal{G}|$ iterated over partitions. Formally, $g \notin \bigcup_{q \in [\texttt{count} - \sqrt{n} + |\mathcal{G}|, \texttt{count})} \mathcal{P}_q$

Proof. We proceed by joint induction on i, j where $i - j \leq n$. We will first prove the base case where $i - j = \sqrt{n} + 1$ and $i - j \leq \sqrt{n}$ for all prior operations. Note that the last operation had to be an insertion in-order for $i - j$ to increase. Thus, the head contains $\sqrt{n}$ elements, all of which are smaller than the element evicted by `PQ.ExtractLargest` (line 11) in the insertion operation. Note that all partitions, $\mathcal{P}_\alpha$ for $\alpha \in [\sqrt{n}]$, were empty before this operation and thus the call to `DIPQ.order` adds the element p' evicted from the head to $\mathcal{P}_{\texttt{count}}$. Thus the good element, g, ends up in the first partition. As $|\mathcal{G}|$ after the insert is $\sqrt{n}$, we have that the invariant hold trivially.

Now for the inductive case, assume that the invariant hold for insertion count i and extraction count j. We will show that they hold for $i + 1, j$ and $i, j + 1$.

Algorithm 3 Data Independent Priority Queue

function DIPQ.INIT
    count $\leftarrow 0$
    DIPQ.Head $\leftarrow$ PQ.Init$\sqrt{n}$
    $\mathcal{P}_1, \cdots, \mathcal{P}_{\sqrt{n}} \leftarrow \emptyset$
    DIPQ.Stash $\leftarrow \emptyset$
function DIPQ.INSERT(p)
    if $|$DIPQ.Head$| < \sqrt{n}$ **then**
        DIPQ.Head $\leftarrow$ DIPQ.Head.Insert(p)
    else
        DIPQ.Head $\leftarrow$ DIPQ.Insert(DIPQ.Head, p)
        DIPQ.Head, $p' \leftarrow$ DIPQ.Head.ExtractLargest
        DIPQ.Stash $\leftarrow$ DIPQ.Stash $\bigcup \{p'\}$
    Call Order()
function DIPQ.EXTRACTFRONT
    DIPQ.Head, $p \leftarrow$ DIPQ.Head.Front
    $p' \leftarrow$ GetLargest(DIPQ.Stash)
    if $p > p'$ **then** Remove the front most element, p, from DIPQ.Head and set $r = p$.
    else Remove p' from the stash and set $r = p'$
    Call Order()
    return r
function DIPQ.ORDER
    $\mathcal{P}_{\text{count}}$, DIPQ.Stash $\leftarrow$ Fill($\mathcal{P}_{\text{count}}$, DIPQ.Stash)
    DIPQ.Head, $\mathcal{P}_{\text{count}} \leftarrow$ MergeFill(DIPQ.Head, $\mathcal{P}_{\text{count}}$)
    count $\leftarrow$ count $+ 1 \mod \sqrt{n}$

Case 1: $i+1, j$ Let p be the inserted element. Note that if $p \leq a$ for some $a \in \mathcal{G}$, then the new "good" set, $\mathcal{G}'$ has size $|\mathcal{G}'| \leftarrow \min(\sqrt{n}, |\mathcal{G}|+1)$. If $p > a$ for all $a \in \mathcal{G}$, then the new "good" set, $\mathcal{G}'$ has size $|\mathcal{G}|$ or $|\mathcal{G}|+1$ depending on whether $\mathcal{P}_{\texttt{count}}$ contains g or not. In either case, we have that $|\mathcal{G}'| \geq |\mathcal{G}|$. If $|\mathcal{G}'| = \sqrt{n}$, then the invariants hold trivially.

Otherwise, we have that either $|\mathcal{P}_{\texttt{count}}|$ is 0 or non-empty. If $|\mathcal{P}_{\texttt{count}}| = 0$, then $g \notin \mathcal{P}'_{\texttt{count}}$ where $\mathcal{P}'_{\texttt{count}}$ is the updated $\mathcal{P}_{\texttt{count}}$ after running the insertion. In the case that $|\mathcal{P}_{\texttt{count}}| > 0$, note that because we call `Order`() at the end of insertion (line 13) and $|\mathcal{G}'| < \sqrt{n}$, `DIPQ.Head` either has an empty slot or an element β such that $\beta > g$. Thus after calling `DIPQ.Order`, we have that g is moved into `DIPQ.Head`$'$ if $g \in \mathcal{P}_{\texttt{count}}$ by the functionality of `MergeFill`. We can then see that g is not in $\mathcal{P}'_{\texttt{count}}$ and thus the invariants hold that $g \notin \bigcup_{q \in [\texttt{count}+1-\sqrt{n}+|\mathcal{G}|, \texttt{count}+1)} \mathcal{P}_q$.

Case 2: $i, j+1$ Note that on an extraction from `DIPQ.Head`, $|\mathcal{G}|$ may decrease by 1. Again, for the case where the new good set, $\mathcal{G}'$, has size $\sqrt{n}$, the invariants hold trivially. So, we assume that $|\mathcal{G}'| < \sqrt{n}$. We must now show that g is not in the updated current partition, $\mathcal{P}'_{\texttt{count}}$, after `DIPQ.Order` is called. We can show this because we call `MergeFill` on the head and the current partition, $\mathcal{P}_{\texttt{count}}$. I.e. if `MergeFill` introduced element g into the head, $\mathcal{G} \notin \mathcal{P}'_{\texttt{count}}$ and thus the invariants hold. Otherwise, if `MergeFill` did not introduce g into the head, then $g \notin \mathcal{P}_{\texttt{count}}$ and so $g \notin \mathcal{P}'_{\texttt{count}}$ as elements in $\mathcal{P}_{\texttt{count}}$ can only increase after `MergeFill`. By the inductive hypothesis and the above, we can see that $g \notin \bigcup_{q \in [\texttt{count}+1-\sqrt{n}+|\mathcal{G}|-1, \texttt{count}+1)} \mathcal{P}_q$. Thus, we then have that invariant holds.

By induction, we have that the invariant holds for all i, j when the number of elements in the priority queue passed $\sqrt{n}$ at some point in the sequence of operations. □

1.2 Correctness Proof

Assuming that the invariants hold in section 1.1, we will show that the algorithm is correct. Note that if the total number of elements in the queue never exceeded $\sqrt{n}$ elements, correctness holds as elements are never moved out of the head which is itself a priority queue. Otherwise, we will show that $\mathcal{P}_{\texttt{count}}$ is always "close enough" to the next "good" element. Whenever `DIPQ.ExtractFront` is called, we have that $|\mathcal{G}|$ may decrease by 1. If $|\mathcal{G}|$ does decrease by 1, we still have that the partition containing the next good element, g, will be reached in at most $|\mathcal{G}|-1$ `DIPQ` operations. So, by the call of `MergeFill` in `DIPQ.Order`(), we will have that g is in the head or the stash after $|\mathcal{G}|-1$ calls. We can then guarantee that the smallest element in the head and stash are at least as small as g. If no insertions were called with element e where $e < g$, then after $|\mathcal{G}-1|$ operations, our new $\mathcal{G}$ set, $\mathcal{G}'$, will have size of at least $|\mathcal{G}|-1-(|\mathcal{G}|-1)+1 = 1$ as g will be moved into the head. Thus, calling `DIPQ.ExtractFront` will return the smallest element as we always have that $|\mathcal{G}| > 0$. Otherwise, if e is inserted within the $|\mathcal{G}|-1$ operations, then e will be in the head and be a part of the new $\mathcal{G}$ set, $\mathcal{G}'$. Similarly, after $|\mathcal{G}|-1$ operations, we will have that $|\mathcal{G}'| \geq 1$ and thus the smallest element will be returned by `DIPQ.ExtractFront`.

1.3 Time Complexities

1.3.1 Stash Size

Fist, we will show that $|\texttt{DIPQ.Stash}| \leq \sqrt{n}$. As, $|\texttt{DIPQ.Stash}|$ and $\mathcal{P}_\alpha$, for all $\alpha \in [\sqrt{n}]$, does not grow if $|\texttt{DIPQ.Head}| < \sqrt{n}$, we will assume that $|\texttt{DIPQ.Head}| = \sqrt{n}$. Note that we are guaranteed that the total number of elements in `DIPQ` is at most n and that the capacity of each partition is

$\mathcal{P}_\alpha = \sqrt{n}$. So, the total capacity of all partitions is n and, we have that there is always at least $\sqrt{n}$ empty slots in the partitions. Thus, we have that no element which is added to the stash remains in the stash for more than $\sqrt{n}$ calls to `DIPQ.Order`() as each call attempts to place an element from the stash into the current partition. Moreover, after performing an initial $\sqrt{n}$ insertions, we have for each subsequent operation, we are guaranteed to be able to cumulatively remove $i + j - \sqrt{n}$ elements from the stash where $i + j$ is the total number of operations. So, because we can add at most i elements to the stash and remove at least $i + j - \sqrt{n}$ elements from the stash, we have that $|\texttt{DIPQ.Stash}| \leq \sqrt{n}$.

1.4 Complexity

Now, we can show that the time complexity of `DIPQ.Insert` and `DIPQ.ExtractFront` is $O(\sqrt{n} \log n)$. We assume that we can sort a list of size k in $O(klogk)$ time. Upon insertion into the head, calling `DIPQ.Head.Insert` and `DIPQ.Head.ExtractLargest` takes $O(\sqrt{n} \log \sqrt{n}) = O(\sqrt{n} \log n)$ time. We can also see that `DIPQ.Order` takes $O(\sqrt{n} \log n)$ time as `Fill`, `MergeFill` take $O(\sqrt{n} \log n)$ time. Thus, both `DIPQ.Insert` and `DIPQ.ExtractFront` take $O(\sqrt{n} \log n)$ time.

References