Multiple Secret Leaders

Authors here

August 25, 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`

Input: $\texttt{Head}, \mathcal{P}_{\texttt{count}}$
$e_1, ..., e_\gamma = \texttt{sort}(\texttt{Head}, \mathcal{P}_{\texttt{count}})$ where $\gamma = |\texttt{Head}| + |\mathcal{P}_{\texttt{count}}|$ ▷ Via a data-independent sort
$\texttt{Head} = \{e_1, ..., e_{\min\left(\sqrt{n}, \gamma\right)}\}$
$\mathcal{P}_{\texttt{count}} = \{e_{\sqrt{n}+1}, ..., e_\gamma\}$
return $\texttt{Head}', \mathcal{P}_{\texttt{count}}$

Algorithm 2 `Fill`

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

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, the size of `DIPQ` 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 that $|\texttt{DIPQ}| > \sqrt{n}$.

Before specifying our invariants, we will add some notation. Let $\mathcal{G}$ ($\mathcal{G}$ for "good") be the largest set of smallest priorities in the head. More formally, $\mathcal{G}$ is the largest subset of `Head` such that $\forall e \in \texttt{DIPQ}, e \notin \mathcal{G}$, $a < e, \forall a \in \mathcal{G}$. Note that $|\mathcal{G}| \leq \sqrt{n}$. Further, let g be the smallest priority in `DIPQ` which is not in the head, `Head`. Note that $g \notin \mathcal{G}$.

We will show that the following invariants hold:

- The g 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 operation has 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 10) in the insertion operation. We thus have that $|\mathcal{G}| = \sqrt{n}$ after the insertion. We can then see that the invariant holds trivially as $\texttt{count} - \sqrt{n} + |\mathcal{G}| = \texttt{count} - \sqrt{n} + \sqrt{n} = \texttt{count}$ and thus $\bigcup_{q \in [\texttt{count} - \sqrt{n} + -\sqrt{n}, \texttt{count})} \mathcal{P}_q = \emptyset$.

Now, to prove the inductive case we need to show that after an operation, the new good set, $\mathcal{G}'$ has size at least $|\mathcal{G}| - 1$ and that $g \notin \mathcal{P}_{\texttt{count}+1}$. Then, as we have the inductive hypothesis that $g \notin \bigcup_{q \in [\texttt{count} - \sqrt{n} + |\mathcal{G}|, \texttt{count})} \mathcal{P}_q$, we can show that $g \notin \bigcup_{q \in [\texttt{count}+1-\sqrt{n}+|\mathcal{G}|-1, \texttt{count}+1)} \mathcal{P}_q$.

We will proceed by assuming 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 (`DIPQ`)

function INIT
    $\texttt{count} \leftarrow 0$
    $\texttt{Head} \leftarrow []$
    $\mathcal{P}_0, \cdots, \mathcal{P}_{\sqrt{n}-1} \leftarrow \emptyset$
    $\texttt{Stash} \leftarrow \emptyset$
function INSERT(p, v)
    if $|\texttt{Head}| < \sqrt{n}$ **then**
        Append (p, v) to `Head`
    else
        $i, (p', v') = \texttt{GetLargest}(\texttt{Head})$ where i is the index of (p', v') in `Head`
        $I = 1$ if $p' < p$ else 0
        $\texttt{Stash}[i] = I \cdot (p, v) + (1 - I) \cdot (p', v')$
        $\texttt{Head}[i] = I \cdot (p', v') + (1 - I) \cdot (p, v)$
    Call `Order`()
function EXTRACTFRONT
    $j, (p, v) = \texttt{GetSmallest}(\texttt{Stash})$ where j is the index of (p', v') in `Stash`
    $k, (p', v') = \texttt{GetLargest}(\texttt{Head})$ where k is the index of (a, b) in `Head`
    $I = 1$ if $p' < p$ else 0
    $\texttt{Stash}[j] = (1 - I) \cdot (p', v') + I \cdot (p, v)$ ▷ Conditionally swap (p, v) and (p', v')
    $\texttt{Head}[k] = I \cdot (p', v') + (1 - I) \cdot (p, v)$ ▷ Conditionally swap (p', v') and (p, v)
    $i, (p_r, v_r) = \texttt{GetSmallest}(\texttt{Head})$
    Call `Order`()
    return (p_r, v_r)
function ORDER
    $\mathcal{P}_{\texttt{count}}, \texttt{Stash} = \texttt{Fill}(\mathcal{P}_{\texttt{count}}, \texttt{Stash})$
    $\texttt{Head}, \mathcal{P}_{\texttt{count}} = \texttt{MergeFill}(\texttt{Head}, \mathcal{P}_{\texttt{count}})$
    $\texttt{count} \leftarrow \texttt{count} + 1 \mod \sqrt{n}$

Case 1: $i+1, j$ Let (p, v) be the inserted element. We will now show that the new good set, $\mathcal{G}'$ has size at least $|\mathcal{G}|$. If $p < a$ for some $a \in \mathcal{G}$, then $p \in \mathcal{G}'$. As at most one element is evicted from $\mathcal{G}$, we have that $|\mathcal{G}'| \geq |\mathcal{G}| - 1 + 1$. If $p > a$ for all $a \in \mathcal{G}$, then the $\mathcal{G}'$ does not lose any elements from $\mathcal{G}$ and so $|\mathcal{G}'| \geq |\mathcal{G}|$.

Now, note that if $|\mathcal{G}'| = \sqrt{n}$, then the invariants hold trivially as $\bigcup_{j \in [\texttt{count}+1-\sqrt{n}+\sqrt{n}, \texttt{count}+1)} \mathcal{P}_j = \emptyset$. So, we assume that $|\mathcal{G}'| < \sqrt{n}$. Note that because we call `Order` at the end of insertion (line 14) and $\sqrt{n} > |\mathcal{G}'| \geq |\mathcal{G}|$, `Head` either had an empty slot or an element β such that $\beta > g$ prior to insertion. If $p < g$, then $g' \leftarrow p$ (where g' is the updated g) and (p, v) are moved into the head. Otherwise, after calling `Order`, we have that g is moved into $\texttt{Head}'$ if $g \in \mathcal{P}_{\texttt{count}}$ by the functionality of `MergeFill`. So, we can then see that $g' \notin \mathcal{P}'_{\texttt{count}}$. Finally, as $g' \leq g$ and no element smaller than or equal to g is in $\bigcup_{q \in [\texttt{count}-\sqrt{n}+|\mathcal{G}|, \texttt{count})} \mathcal{P}_q$ by the inductive hypothesis, $g' \notin \bigcup_{q \in [\texttt{count}+1-\sqrt{n}+|\mathcal{G}|, \texttt{count}+1)} \mathcal{P}_q$.

Case 2: $i, j+1$ Note that on an extraction from `Head`, $|\mathcal{G}|$ may decrease by at most 1. We can then see that the new good set $\mathcal{G}'$ has size at least $|\mathcal{G}| - 1$. 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 `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 $g \notin \bigcup_{q \in [\texttt{count}+1-\sqrt{n}+|\mathcal{G}|-1, \texttt{count}+1)} \mathcal{P}_q$ via an identical line of reasoning as in **case 1**. 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

`ExtractFront` Correctness

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 `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 or g will be in the stash. Thus, calling `ExtractFront` will return the smallest element as we always have that $|\mathcal{G}| > 0$ or, if $|\mathcal{G}| = 0$, g, now the smallest element, is in the stash. Otherwise, if e, such that $e < g$, 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 `ExtractFront`.

Size of Stash

We will show that $|\texttt{Stash}| \leq \sqrt{n}$. As, $|\texttt{Stash}|$ and $\mathcal{P}_\alpha$, for all $\alpha \in [\sqrt{n}]$, does not grow if $|\texttt{Head}| < \sqrt{n}$, we will assume that $|\texttt{Head}| = \sqrt{n}$. 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. We can also note that the stash can only grow by at most 1 element per call to Insert. Thus, assume towards contradiction that $|\texttt{Stash}| > \sqrt{n}$. Then, we have that an element remained in the stash for more than $\sqrt{n}$ calls to DIPQ.Order() which is a contradiction. We can then see that $|\texttt{Stash}| \leq \sqrt{n}$.

Data Independence

We will now show that the priority queue is data independent. We can do this by simply noting that every operation is data independent.

1.3 Time Complexities

1.4 Complexity

Assume that we have a data independent sorting algorithm which takes $O(f(n))$ time where $f(n)$ is some function of n. Now, we can show that the time complexity of Insert and ExtractFront is $O(\sqrt{n} + f(n))$. Note that Order makes a call to MergeFill which does a sort and that the rest of the operations take at most $O(\sqrt{n})$ time. So, Order takes $O(\sqrt{n} + f(n))$ time. We can also see that in Insert and ExtractFront all non-Order operations take at most $O(\sqrt{n})$ time. So, Insert and ExtractFront take $O(\sqrt{n} + f(n))$ time.

References