Range Queries

Lawrence L. Larmore UNLV

1 The Range Query Problem

The input to an instance of the range query problem is a sequence of data $x_1, x_2, \ldots x_n$ of some data type, and an associative operation, which we call *addition* and denote as "+," on that data type. Thus, $x + (y + z) = (x + y) + z$, which we write simply as $x + y + z$, for any three items x, y, and z. We do not assume commutativity, nor do we assume the existence of inverses.

For each $0 \leq p < q \leq n$, let the *range query* $Q(p, q) = \sum_{i=p+1}^{q} x_i$. Our problem is to design a data structure that enables us to evaluate an arbitrary query quickly, while keeping the space complexity and initialization time of the data structure small. We will assume that it takes $O(1)$ time to execute one addition, and $O(1)$ space to store one item.

1.1 Examples.

We give some examples of data types that could arise in practice for this problem.

1. Numbers (real numbers, integers, or whatever) and ordinary addition.
2. Numbers and ordinary multiplication.
3. Numbers and maximum (or minimum).
4. Matrices and matrix multiplication (or tropical matrix multiplication).
5. Bit strings and masking.
6. Lots of others.

1.2 Sets of Intervals

It is helpful to visualize the range query problem as follows. Given an integer n, let $I_{p,q}$ be the half-open interval $(p, q]$, and let $\mathcal{I}_n$ be the set of $I_{p,q}$ for all integers $0 \leq p < q \leq n$.

A solution to the range query problem of size n is to find a set of intervals $\mathcal{D} \subseteq \mathcal{I}_n$ such that every member of $\mathcal{I}_n$ is the exact union of members of $\mathcal{D}$. We define the *size* of our solution to be the cardinality of $\mathcal{D}$, and the *query cost* of our solution to be the maximum number of elements of $\mathcal{D}$ needed to evaluation a query.

1.3 The Function F_d

For any $1 \le d \le n$, we define $F_d(n)$ to be the minimum size of any solution to the range query problem of size n which has query cost d. We will generally write $\mathcal{D}_d(n)$ for a set of intervals that constitutes a solution to that problem. For example, $F_1(n) = \binom{n+1}{2} = \Theta(n^2)$, since the only possible solution of query cost 1 is $\mathcal{D} = \mathcal{I}_n$, that is, $\mathcal{D}$ contains every interval.

At the other extreme, $F_n(n) = n$, by letting $\mathcal{D} = \{I_{i-1,i}\}$. We illustrate these two solutions in Figures 1 and 2.

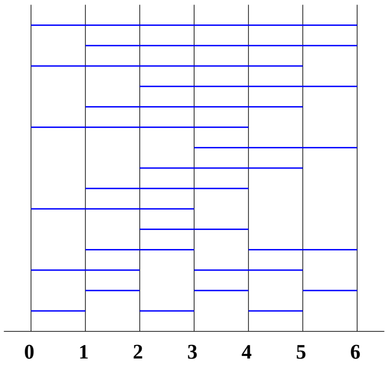

Figure 1: $F_1(n) = \frac{n(n+1)}{2}$

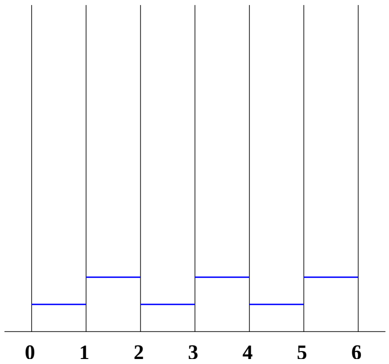

Figure 2: $F_n(n) = n$

1.4 Asymptotic Solutions

Recall that $\alpha_1 = \lceil n/2 \rceil$, $\alpha_2 = \lceil \log_2 n \rceil$, and that, for $k \ge 2$, $\alpha_k(n)$ is defined by the recurrence

$$\alpha_k(n) = \begin{cases} 0 & \text{if } n = 1 \\ 1 + \alpha_k(\alpha_{k-1}(n) & \text{if } n \ge 2 \end{cases}$$

Thus, $\alpha_3(n) = \log^*(n)$.

The following results have been proved.

- $F_1(n) = \Theta(n^2) = \Theta(n\alpha_1(n))$.
- $F_2(n) = \Theta(n \log n) = \Theta(n\alpha_2(n))$.
- $F_3(n) = \Theta(n \log \log n)$.
- For $k \ge 3$, $F_{2k-2}(n) = \Theta(n\alpha_k(n))$
- For $k \ge 3$, $F_{2k-1}(n) = \Theta(n\alpha_k(\log n)) = \Theta(n\alpha_k(n))$.

Thus $F_4(n) = \Theta(n \log^* n) = \Theta(n\alpha_3(n))$, and also $F_5(n) = \Theta(n \log^* n) = \Theta(n\alpha_3(n))$.

1.5 Solution for $d = 2$

We will show that $F_2(n) = \Theta(n \log n)$, using the construction given by Hirschberg and Volper.

The data structure $\mathcal{D}_2(n)$ is defined recursively. At the top level, it contains all intervals which have $\lfloor n/2 \rfloor$ as an end point; we call these *long* intervals. There are n long intervals, shown in blue in Figure 3, in the example where $n = 13$. Any query $Q(p,q)$ for $p \le \lfloor n/2 \rfloor \le q$ can be evaluated by combining the stored values for at most two long intervals, namely $I_{p,\lfloor n/2 \rfloor}$ and $I_{\lfloor n/2 \rfloor,q}$. To cover queries in the range $0 \le p < q < \lfloor n/2 \rfloor$ or $\lfloor n/2 \rfloor < p < q \le n$, we add a copy of $D_2(\lfloor n/2 \rfloor - 1)$ on the left, and a copy of $D_2(n - \lfloor n/2 \rfloor - 1)$ on the right. If $n \le 2$, there is no recursive step. Each of the layers of the construction contains at most n intervals, and there are at most $\log_2 n$ layers, giving us the upper bound.

We can also set up a recurrence. Ignoring small discrepencies caused by rounding and adding or subtracting 1, we have:

$$F_2(n) \le n + 2F(n/2)$$

giving us $F_2(n) = O(n \log n)$ by the Master Theorem.

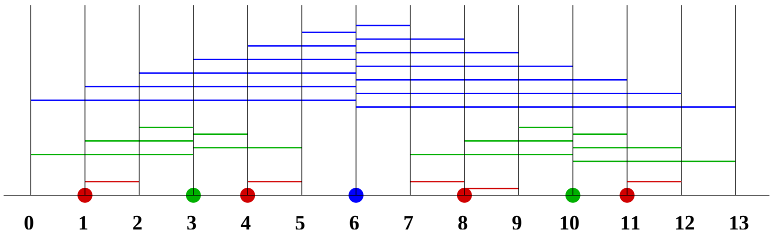

Figure 3: $F_2(n) = \Theta(n \log n)$.

The lower bound is a bit harder. We need to prove the following theorem.

Theorem 1.1 *Any solution to the query problem whose query cost is 2 has size* $\Omega(n \log n)$.

Proof: Define a query $Q(p,q)$ to be *long* if $p \le \lfloor n/2 \rfloor \le q$. We also define an interval $I_{p,q}$ to be *long* if $p \le \lfloor n/2 \rfloor \le q$, namely that $I_{p,q}$ contains the point $\lfloor n/2 \rfloor$.

All other queries and intervals are called *short*. By definition, the number of intervals in $\mathcal{D}_n$ needed to evaluate short queries is $F_2(\lfloor n/2 \rfloor - 1) + F_2(n - \lfloor n/2 \rfloor - 1)$, and these intervals are all short.

However, no long query can be evaluated using only short intervals. Let $\mathcal{D}_n^{long}$ be the set of long intervals in $\mathcal{D}_n$.

Let L be the set of all $0 \le p < \lfloor n/2 \rfloor$ such that p is the left end point of some long interval, and let R be the set of all $\lfloor n/2 \rfloor < q \le n$ such that q is the right end point of some long interval. Then $|R| \ge |L| = \lfloor n/2 \rfloor$.

Suppose there are fewer than $\lfloor n/2 \rfloor$ long intervals $\mathcal{D}_n$. Then there must be some $0 \le p < \lfloor n/2 \rfloor$ such that $p \notin L$, as well as some $\lfloor n/2 \rfloor < q \le n$ such that $q \notin R$. The query $Q(p,q)$ cannot be evaluated using only two intervals in $\mathcal{D}_n$, since at least one of those intervals must be a long interval with an end point at either p or q, constradiction. Thus, the cardinality of $\mathcal{D}_n^{long}$ is at least $\lfloor n/2 \rfloor$

Ignoring rounding and addition or subtraction of 1, we have the recurrence

$$F_2(n) \geq 2F_2(n/2) + n/2$$

By the Master Theorem, $F_2(n) = \Omega(n \log n)$. □

1.6 Estimating $F_3(n)$

Construction of $\mathcal{D}_3(n)$ is more complex. We first pick a set of numbers, spaced approximately $\sqrt{n}$ apart. Let us call those *top* numbers. We now choose *long top* intervals and *short top* intervals, as follows.

1. Between every pair of top numbers, there is a long top interval.
2. If a number i is not top, then there is an interval between i and the closest left top number, and also between i and the closest right top number. We will call those *short top* intervals.

There are approximatelty $\sqrt{n}/2$ long top intervals, and approximately $2n$ short top intervals. Thus, the number of top intervals is $O(n)$.

$\mathcal{D}_3(n)$ is then defined to consist of the following intervals.

1. All top intervals.
2. If b and b' are neighboring top numbers, let $m = b' - b - 1$, the number of integers strictly between b and b'. We then recursively consruct $\mathcal{D}_3(m)$, and place a copy of that structure, shifted to the right by b, into $\mathcal{D}_3(n)$.

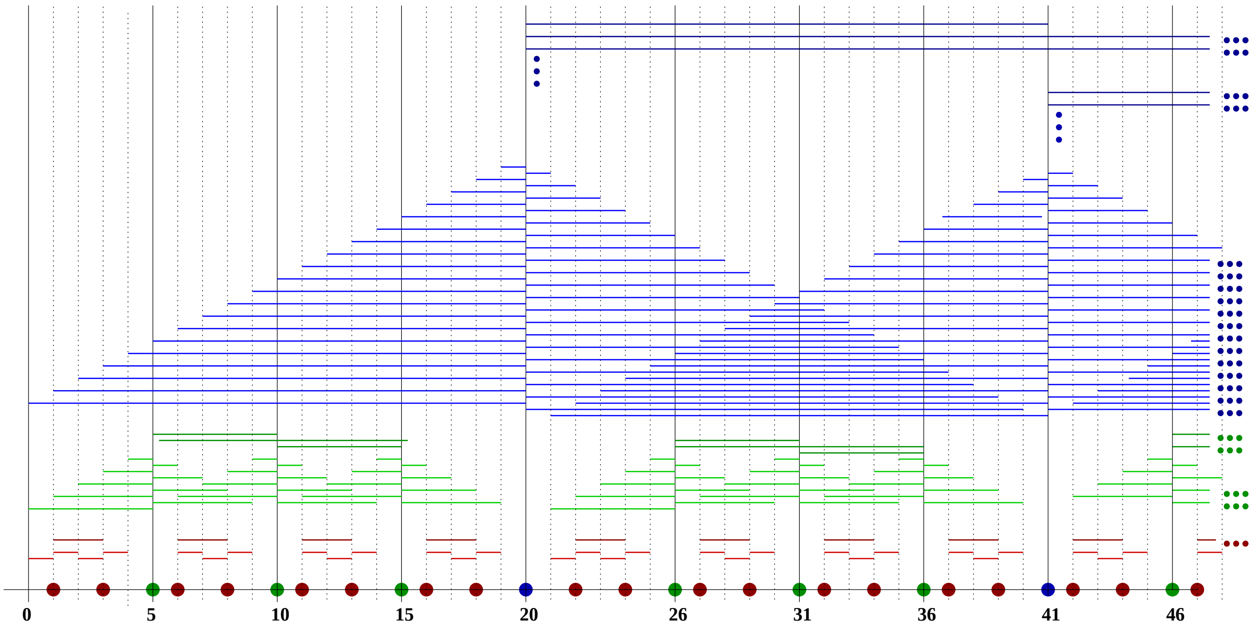

Figure 4: $\mathcal{D}_3(n)$ where $n = 400$ Top intervals are shown in blue. The top intervals of structures at the second first recursive level, such as $\mathcal{D}_3(19)$, are shown in green. The top intervals of the structures at the second recursive level, such as $\mathcal{D}_3(5)$, are shown in red. Top numbers are shown in blue. Top numbers at the first level of recursion are shown in green, and top numbers at the second level of recursion are shown in red.

We analyze F_3 by setting up a recurrence. Using the recursive constructino of $\mathcal{D}_3(n)$ given above, and ignoring rounding and addition or subtraction of 1, we have the recurrence

$$F_3(n) \leq \sqrt{n} F_3(\sqrt{n}) + O(n)$$

from which we conclude that $F_3(n) = O(\log \log n)$.

1.7 $F_4(n)$

We construct $\mathcal{D}_4(n)$ as follows.

1. Pick a set of *top numbers*, spaced approximately $\log_2 n$ apart.
2. Let m be the number of top numbers, approximately $n/\log_2 n$. Attach a copy of $\mathcal{D}_2(m)$ to the top numbers. The intervals of this structure will be called *long top intervals*.
3. There will be a short top interval between each number that is not top with the nearest top numbers on both its right and its left.
4. Inside any gap between two top numbers, say of size ℓ, attach a copy of $\mathcal{D}_4(\ell)$.

We obtain a recurrence allowing us to solve for the size of $\mathcal{D}_4(n)$.