Linear Range Queries

Suppose that a number of values are stored in a line, a plane, or some other space. A *range query* consists of giving a *range*, such as in interval in a line, a square or other region in a plane; in general, some subset of the space. The answer to the query is the sum of the numbers in that range. Alternatively, the query could be for the minimum or maximum of the numbers in the range, or any other appropriate function of the numbers. In our discussion, we will refer to the "sum," with the understanding that the query could be for some other function.

We will restrict our attention to queries in a line, where there are only finitely many numbers on the line, each at some point. Without loss of generality, each number x_i can be thought of as being the value of the interval $[i-1,i]$, the query $Q(p,q) = Q([i,j])$, for $0 \le p < q \le n$, returns $\sum_{i=p+1}^{q} x_i$. We can replace addition by any associative operation, such as min or max. Thus we have $Q(i,j) + Q(j,k) = Q(i,k)$ for integers $i < j < k$. Throughout our discussion, we assume we are given interval values $x_1, \ldots x_n$ for some n.

Range Query Data Structures

Our solution to the range query problem consists of building a data structure which hold values of $Q(I)$ for I in a selected set of intervals. Query(J) for any interval is then the sum $\sum_{i=1}^{k} Q(I_i)$ where each $Q(I_i)$ is stored in the data structure, and J is the disjoint union $\coprod_{i=1}^{k} I_i$

If Subtraction is Permitted

In the special case where operation is actually addition we can build a structure of size $\Theta(n)$ where each query can be answered with only two fetches to the structure. Simply let S_i be the sum of the first i entries. The query QI, for $I = (i,j)$, returns simply $S_j - S_{i-1}$. However, subtraction is not always available, since the operation could be min, or max, or some other associative operation.

Time Space Tradeoff

There is a tradeoff between the size of the data structure, *i.e.* the number of query value stored, and the time it takes to execute a query, measured by the number of query values needed to obtain an answer. For example, if we store the values of all intervals, structure has size $\Theta(n^2)$, and a query needs to look at only one stored value. At the other extreme, if we are allowed to fetch n intervals during a query, the data structure contains all intervals of length 1. Such a structure is clearly pointless.

Union of Two Intervals

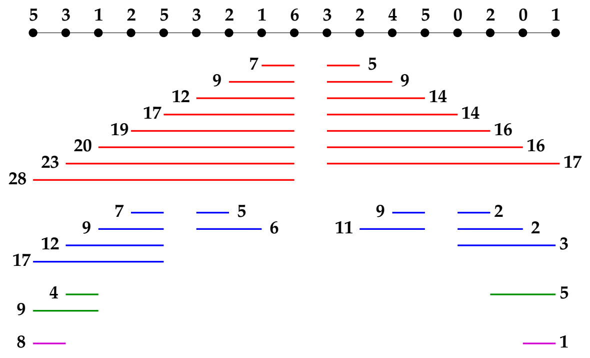
The simplest structer that is an improvement over the trivial $O(n^2)$ size case is a data structure that stores $\Theta(n \log n)$ intervals and their query values, and where each interval J is the union of at most two of the stored intervals. We now describe the structure.

Ranks

Each integer in $1 \ldots n$ has a *rank*. If $i = p2^k$, where p is odd and k is an integer, we define $\mathit{rank}(i) = k$. Thus $0 \leq \mathit{rank}(i) \leq \lfloor \log_2 n \rfloor$ for all i in the range. We assign $\mathit{rank}(0) = \infty$.

List of Stored Intervals

For every stored interval $[p, q]$, $\mathit{rank}(p) \neq \mathit{rank}(q)$. We say $[p, q]$ is a *left* interval of rank k if $\mathit{rank}(q) = k$ and $\mathit{rank}(i) < k$ for all $p \leq i < q$. We say $[p, q]$ is a *right* interval of rank k if $\mathit{rank}(p) = k$ and $\mathit{rank}(i) < k$ for all $p < i \leq q$. We also say that $[0, q]$ is a right interval of rank ∞, for any $1 \leq q \leq n$. The data structure contains all left and right intervals and their values. For each $0 \leq k \leq \lfloor \log_2 n \rfloor$, there are fewer than n left intervals and fewer than n right intervals of rank k. There are also $n - 1$ right intervals of rank ∞. The size of the data structure is $\Theta(n \log n)$.



Queries

We now compute $Q(p, q)$ for $0 \leq p < q \leq n$. If $[p, q]$ is a left or right interval, the value of the query is in the data structure. Otherwise, there is some r such that $p < r < q$ and r has the maximum rank in that interval. Then $Q(p, q) = Q(p, r) + Q(r, q)$, which means we need only two fetches from the data structure.

Union of Three Intervals

We now define a data structure that stores $\Theta(n \log \log n)$ intervals and their query values, and where each interval is the union of at most three of the stored intervals.

Ranks

Each integer in $0 \ldots n$ has a *rank*. If $i = p2^k$, where p is odd and $k > 0$ is an integer, we define $\mathit{rank}(i) = \lfloor \log_2 k \rfloor$. Thus $0 \leq \mathit{rank}(i) \leq \lfloor \log_2 \log_2 n \rfloor$ for all $1 \leq i \leq n$. If i is odd, let $\mathit{rank}(i) = -1$. Let $\mathit{rank}(0) = \infty$.

List of Stored Intervals

We say $[p, q]$ is a *left* interval of rank k if $\mathit{rank}(q) = k$ and $\mathit{rank}(i) < k$ for all $p \le i < q$. We say $[p, q]$ is a *right* interval of rank k if $\mathit{rank}(p) = k$ and $\mathit{rank}(i) < k$ for all $p < i \le q$. We also say that $[0, q]$ is a right interval of rank ∞, for any $1 \le q \le n$. We say that $[p, q]$ is a *bridge* interval of rank k if $\mathit{rank}(p) = \mathit{rank}(q) = k$, and $\mathit{rank}(i) \le k$ for all $p < i < q$. Our data structure stores all left, right, and bridge intervals, and their values.

There are fewer than n left intervals of each rank, and fewer than n right intervals of each rank. There are no bridge intervals of rank -1 or 0. If $[p, q]$ is a bridge interval of any rank, we say that $p \sim q$. Note that $\sim$ is an equivalence relation. Each equivalence class of integers of rank $k \ge 1$ under that relation is an arithmetic sequence consisting of at most $2^{2^k} - 1$ consecutive multiples of 2^{2^k}. Thus, there are fewer than n bridge intervals of rank k, hence fewer than $n\lfloor \log_2 \log_2 n \rfloor$ bridge intervals altogether. The number of stored intervals is thus $O(n \log\log n)$, actually $\Theta(n \log\log n)$.

Queries

We need to show that every query interval is the disjoint union of at most three intervals stored in our data structure. Let $Q(p, q)$ be a query. We define the *rank* of that query to be the maximum rank of any $\left[{}^{i\in}_{[p,q]}\right]$. Let $k = \mathit{rank}(k)$. There are several cases.

1. If $\mathit{rank}(p) = \mathit{rank}(q) = k$, then $[p, q]$ is a bridge interval.
2. If $\mathit{rank}(p) = k$ and $\mathit{rank}(q) < k$, let r be the maximum integer of rank k in $[p, q]$. If $r = p$, then $[p, q]$ is a right interval. Otherwise, $[p, q]$ is the disjoint union of the bridge interval $[p, r]$ and the right interval $[r, q]$.
3. The case $\mathit{rank}(q) = k$, $\mathit{rank}(p) < k$, is similar to the previous case.
4. If $\mathit{rank}(p) < k$, $\mathit{rank}(q) < k$ and r is the only integers of rank k in $[p, q]$, then $[p, q]$ is the disjoint union of the left interval $[p, r]$ and the right interval $[r, q]$.
5. If $\mathit{rank}(p) < k$, $\mathit{rank}(q) < k$, and there are two or more integers of rank k in the interval $[p, q]$, let r and s be the minimum and maximum integers of rank k in the interval. Then $[p, q]$ is the disjoint union of the left interval $[p, r]$, the bridge interval $[r, s]$, and the right interval $[s, q]$.

Four or More Intervals

As we increase the number of intervals we are allowed to fetch during a query, the data structure gets smaller.

If a query is allowed to fetch four values from the data structure, we can solve the problem using a data structure of size $\Theta(n \log^* n)$. The construction is far more complex than any of the previous constructions.

We might ask how to "balance" our solution: what is the minimum m such that there are at most m intervals fetched during each query and that the data structure contains nm intervals? It turns out that $m = \Theta(\alpha(n))$, where α is the (notorious) inverse Ackermann function. That is, we can store $\Theta(n\alpha(n))$ query values, and each query requires combining no more than $\alpha(n)$ stored values.