Range Queries

Ajay Kumar

Date 28 Jan 2023

Sparse Table

- Sparse Table is a data structure, that allows answering range queries. It can answer most range queries in O(logn)
- but its true power is answering range minimum queries (or equivalent range maximum queries).
- For those queries it can compute the answer in O(1) time.
- The only drawback of this data structure is, that it can only be used on *immutable* arrays. This means, that the array cannot be changed between two queries.
- If any element in the array changes, the complete data structure has to be recomputed.

Intuition

- Any non-negative number can be uniquely represented as a sum of decreasing powers of two.
- This is just a variant of the binary representation of a number. E.g. 13=$(1101)_2$=8+4+1.
- For a number x there can be at most $\lceil log_2 x \rceil$ summands.
- By the same reasoning any interval can be uniquely represented as a union of intervals with lengths that are decreasing powers of two. E.g. [2,14] = [2, 9] ∪ [10, 13] ∪ [14, 14], where the complete interval has length 13, and the individual intervals have the lengths 8, 4 and 1 respectably.
- And also here the union consists of at most $\lceil log_2(length\ of\ interval) \rceil$ many intervals.
- The main idea behind Sparse Tables is to precompute all answers for range queries with power of two length.
- Afterwards a different range query can be answered by splitting the range into ranges with power of two lengths, looking up the precomputed answers, and combining them to receive a complete answer.

7.2.2 Precomputation

We will use a 2-dimensional array for storing the answers to the precomputed queries. st[i][j] will store the answer for the range $[i, i + 2^j - 1]$ of length 2^j. The size of the 2-dimensional array will be MAXN × (K + 1), where MAXN is the biggest possible array length. K has to satisfy $K \geq \lfloor \log_2 \text{MAXN} \rfloor$, because $2^{\lfloor \log_2 \text{MAXN} \rfloor}$ is the biggest power of two range, that we have to support. For arrays with reasonable length ($\leq 10^7$ elements), K = 25 is a good value.

```
int st[MAXN][K + 1];
```

Because the range [i, i + 2^j − 1] of length 2^j splits nicely into the ranges [i,i + 2^{j-1} − 1] and [i + 2^{j-1},i + 2^j − 1], both of length 2^{j-1}, we can generate the table efficiently using dynamic programming:

```
for (int i = 0; i < N; i++) st[i][0] = f(array[i]);
for (int j = 1; j <= K; j++)
for (int i = 0; i + (1 << j) <= N; i++)
    st[i][j] = f(st[i][j-1], st[i + (1 << (j - 1))][j - 1]);
```

7.2.2 Precomputation

We will use a 2-dimensional array for storing the answers to the precomputed queries. st[i][j] will store the answer for the range $[i, i + 2^j - 1]$ of length 2^j. The size of the 2-dimensional array will be MAXN × (K + 1), where MAXN is the biggest possible array length. K has to satisfy $K \geq \lfloor \log_2 \text{MAXN} \rfloor$, because $2^{\lfloor \log_2 \text{MAXN} \rfloor}$ is the biggest power of two range, that we have to support. For arrays with reasonable length ($\leq 10^7$ elements), K = 25 is a good value.