```cpp
#include "../bits/stdc++.h"
/**
 *  Disjoint Sparse Table
 *  静的なデータ列の区間に対し，
 *  演算 op を 構築 O(NlogN)，取得 O(1)
 *  モノイドを乗せる
 */
// english: https://discuss.codechef.com/t/tutorial-disjoint-sparse-table/17404
// japanese: http://noshi91.hatenablog.com/entry/2018/05/08/183946
// verified: https://www.spoj.com/problems/RMQSQ/ submission:23714831
template <typename M>
class DisjointSparseTable
{
    using T = typename M::type;
    std::vector<std::vector<T>> memo;

  public:
    DisjointSparseTable(const std::vector<T> &v)
    {
        int b = 0;
        while ((1 << b) <= static_cast<int>(v.size()))
            b++;
        memo.resize(b, std::vector<T>(static_cast<int>(v.size()), M::id()));
        for (int i = 0; i < static_cast<int>(v.size()); i++)
            memo[0][i] = v[i];
        for (int i = 1; i < b; i++)
        {
            int st = 1 << i;
            for (int j = 0; j < static_cast<int>(v.size()); j += st << 1)
            {
                int t = std::min(j + st, static_cast<int>(v.size()));
                memo[i][t - 1] = v[t - 1];
                for (int k = t - 2; k >= j; k--)
                    memo[i][k] = M::op(v[k], memo[i][k + 1]);
                if (static_cast<int>(v.size()) <= t)
                    break;
                memo[i][t] = v[t];
                int r = min(t + st, static_cast<int>(v.size()));
                for (int k = t + 1; k < r; k++)
                    memo[i][k] = M::op(memo[i][k - 1], v[k]);
            }
        }
    }

    // [a, b) の op 結果
    T query(int l, int r)
    {
        if (l >= --r)
            return memo[0][l];
        int b = 31 - __builtin_clz(l ^ r);
        return M::op(memo[b][l], memo[b][r]);
    }
};

```