Range Minimum Maximum Query

Robert Bakarić
rbakaric@irb.hr
bakaric@evolbio.mpg.de
27.03.2015
rmmq-1.0

Abstract

Given an array of numbers (a) and two position $i, j \in [1, |a|]$, the range minimum/maximum query problem is defined as a problem of finding the minimum/maximum value of an element within an sub-array defined by the two position values. The solution presented here is a *sparse table* based algorithm with $O(|a| \log(|a|))$ construction and $O(1)$ runtime complexity, implemented in C++.

Contents

1 Installation

The simplest way to compile this program is to:

1. Unpack the rmmq package (`rmmq-XXX.tar.gz`):

   ```
   tar -xvzf rmmq-XXX.tar.gz
   ```

2. Change the current directory to `rmmq-XXX`:

   ```
   cd rmmq-XXX/
   ```

3. Configure the program for your system (–bindir is optional):

   ```
   ./configure --bindir=/absolute/directory/path/rmmq-xxx/bin
   ```

4. Compile the program:

   ```
   make
   ```

5. Install the program:

   ```
   make install
   ```

Your binaries should be located in your local bin directory if `--bindir` option has been set. Otherwise installation needs to be carried out with root privilages in order to be installed into `/usr/local/bin` directory.

2 Input files

The rmmq takes a simple array of integers specified in a one-column ascii file. An example of the input file can be found in `./rmmq-xxx/examples` and it should look like this:

`range.txt`:

```
1
2
4
1
5
67
6
4
5
67
3
...
```

3 Program options

I order to see program options type:

```
./bin/rmmq -h
```

Expected output:

```
Usage: ./program [options]

**************************************************************************************
                              rmmq - Range minimum/maximum query
                                          by
                                    Robert Bakaric

CONTACT:
 Code written and maintained by Robert Bakaric,
 email: rbakaric@irb.hr , bakaric@evolbio.mpg.de

ACKNOWLEDGEMENT:
 http://www.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor

LICENSE:
 The program is distributed under the GNU General Public License. You should have
 received a copy of the licence together  with this software. If not, see
 http://www.gnu.org/licenses/
***************************************************************************************


Allowed options:
  -h [ --help ]           produce help message
  -v [ --version ]        print version information
  -i [ --input-file ] arg input file
```

4 Functions and classes

Sparse table:

`ST` : Sparse table class.

`make` : Explicit constructor. Given a range the function creates a sparse table for Min/Max RMMQ function calls.

`destroy` : Explicit destructor.

Range minimum/maximum query:

`RMMQ` : Range minimum maximum query class.

`make` : Explicit constructor. Given a range, the function creates a local map and calls ST constructor.

`destroy` : Explicit destructor. Destroys the local range map and all subsequent ST tables.

`MinVal` : Given two index positions the function returns the minimum value within a given range.

`MinPos` : Given two index positions the function returns the array location of a minimum value within a given range.

`MaxVal` : Given two index positions the function returns the maximum value within a given range.

`MaxPos` : Given two index positions the function returns the array location of a maximum value within a given range.

Note: all positions refer to array index values, therefore, if necessary `-1` can be added to any input or output value.

5 Example

5.1 rmmq.cpp

A minimal example demonstrating the usage of rmmq demo program:

```
./bin/rmmq -i examples/range.txt
Positions: [1] [2] [3] [4] [5] ... [25] [26] [27] [28]
Values:    1    2   4   1   5   ... 35   36    3   37
Note: Ctrl-c to quit
Start: 4
Stop: 14

min(4,14): val(1) pos(4)
max(4,14): val(68) pos(12)
```

5.2 rmmq.hpp

Adding the `rmmq.hpp` header file to your personal script will allow you to include all the functions described in section 4. A minimal example:

```
 #include<vector>
 #include<rmmq.hpp>

 vector<int> vec {1,5,23,7,8,3,12,5,3,44,56};

/* sparse table */
    /* Construct */
    ST<int> sptab(vec);
    /* OR */
    ST<int> sptab;
    sptab.make(vec)

    /* Explicite Destructor */
    sptab.destroy();




/* rmmq */
    /* Construct */
    RMMQ<int> rmmq(vec);
    /* OR */
    RMMQ<int> rmmq;
    rmmq.make(vec)

    /* Explicite Destructor */
    rmmq.destroy();





    /* Range Minimum Query for (5,3) */
    /* return value */
    rmmq.MinVal(5-1,3-1); // returns 7  (-1 is for vector index positions since
                          // indexing starts from 0)

    /* return position */
    rmmq.MinPos(5-1,3-1); // returns 3 since indexing starts from 0 (-1 is for
                          // vector index positions since indexing starts from 0)

    /* Range Maximum Query for (3,5) */
    /* return value */
    rmmq.MaxVal(3-1,5-1); // returns 23  (-1 is for vector index positions
                          // since indexing starts from 0)

    /* return position in array */
    rmmq.MaxPos(3-1,5-1); // returns 2 since indexing starts from 0 (-1 is for
                          // vector index positions since indexing starts from 0)
```

6 Acknowledgement

This algorithm was written according to:

`http://www.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor`

For further reference see:

Fischer, J. and V. Heun (2006). "Theoretical and practical improvements on the RMQ-problem, with applications to LCA and LCE". Combinatorial Pattern Matching: 36-48.

Fischer, J. and Heun, V. (2007). A New Succinct Representation of RMQ-Information and Improvements in the Enhanced Suffix Array. Proceedings of the International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies. LNCS 4614. Springer. pp. 459-470.

7 Future work

1. Implement $O(|a|^2)$ construction algorithm.
2. Implement ±1RMMQ $O(|a|)$ construction algorithm.