Degree Project

Presentation #1

Erik Regla

Universidad de Talca

April 28, 2020

- Advisor: Rodrigo Paredes (rapa)
- Thema: Worst-Case optimal incremental sorting for discrete classes
- Current Status: A paper. Initial tests and code ready. Pending documentation.
- Motivation: Everyone loves to talk big about algorithms, but no one actually implements them.

IQS (Set A, Index idx, Stack S)

// Precondition: $idx \leq S.\mathbf{top}()$

If $idx = S.\mathbf{top}()$ **Then** $S.\mathbf{pop}()$, **Return** $A[idx]$

$pidx \leftarrow \mathbf{random}[idx,\ S.\mathbf{top}()-1]$

$pidx' \leftarrow \mathbf{partition}(A,\ A[pidx],\ idx,\ S.\mathbf{top}()-1)$

// Invariant: $A[0] \leq \ldots \leq A[idx-1] \leq A[idx, pidx'-1] \leq A[pidx']$

// $\leq A[pidx'+1, S.\mathbf{top}()-1] \leq A[S.\mathbf{top}(), m-1]$

$S.\mathbf{push}(pidx')$

Return IQS$(A,\ idx,\ S)$

Figure: IQS algorithm as published in 10.1007/s00453-010-9400-6

```
procedure IIQS(A, S, k)
    while k < S.top() do
        pidx ← random(k, S.top() − 1)
        pidx ← partition(A_{k,S.top()−1}, pidx)
        m ← S.top() − k
        α ← 0.3
        r ← −1
        if pidx < k + αm then
            r ← pidx
            pidx ← pick(A_{r+1,S.top()−1})
            pidx ← partition(A_{r+1,S.top()−1},
                             pidx)
        else if pidx > S.top() − αm then
            r ← pidx
            pidx ← pick(A_{k,pidx})
            pidx ← partition(A_{k,r}, pidx)
            r ← −1
        end if
        S.push(pidx)
        if r > −1 then
            S.push(r)
        end if
    end while
    S.pop()
    return A_k
end procedure
```

Figure: IQS algorithm as published in 10.1109/SCCC.2015.7416566

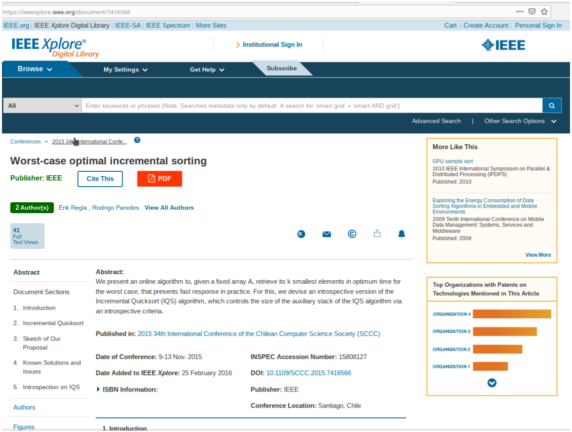

Figure: IIQS implmementation changes the partition method in order to guarantee a partition of linear time and at the same time guarantee a reduction on the search space. (10.1109/SCCC.2015.7416566).

Current scope is limited as an experimental algorithm design [1] to extend (I)IQS usage for haplotype plot[2] generation, which is an instance of the worst case for IQS but on a discrete space when $C << n$.

Tested variants of the original implementation are as follows:

- Test 1: Add incremental version of BFPRT algorithm
- Test 2: Change rules for introspective step
- Test 3: Bias the three-way-median returned index
- Test 4: Store the three-way-median result on the stack
- Test 5: Alter rules for storing pivots

[1] doi:10.1017/CBO9780511843747
[2] doi:10.1111/2041-210X.12747

The beforementioned changes can induce new problems and behaivours, known but not limited to stack size, cache trashing, non-existant DRAM bursting, memory corruption, etc.

- Test 1: No changes to running speed, improved median selection with less memory ussage
- Test 2: Depends on the distribution (not applicable if it is not known).
- Test 3: Stabilization over running time and stack usage. Increased DRAM burst toll.
- Test 4: Cache and DRAM burst trashing on low-power devices. Trade-off between use cases as it doesn't perform on the general case.
- Test 5: Depends on the input distribution, useless if not combined with one of the former tests.

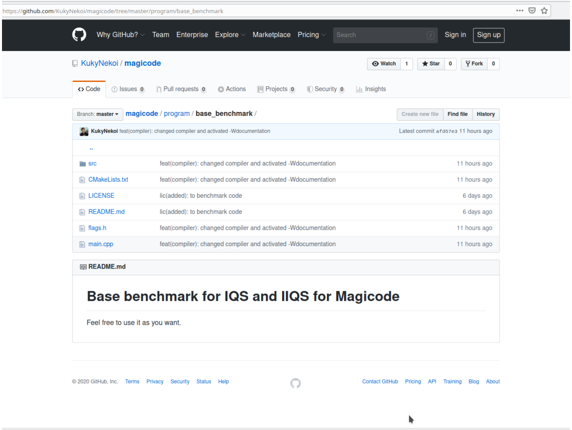

Figure: Implementation for the tests on C++ (STL-container-friendly implementations and without STL) available under GNU GPL license at GitHub

- Scope: Experimental design, setup and experimentation
- Part of magicode, a personal research on FPGA implementation of hardware accelerators for similarity search (the original thema).
- Got someone interested on using this algorithm for solving haplotype plots.

FIN