Succinct Data Structures for Representing Equivalence Classes

Ian Munro

University of Waterloo

Joint work with
Moshe Lewenstein and
Venkatesh Raman

Succinct Data Structures … The Game

Represent a combinatorial object, size n, using lg(# objects) "or so" bits

Perform the necessary queries "quickly"

"Naming" can be an issue. Primary interest in names 1…n.

Not an issue for "dense graphs", but is for trees, planar graphs, and definitely for equivalence classes

Label Space: Direct Equivalence Queries

- But first … give elements "names" over "smallest space" so that elements in same class can be recognized, with no extra space
- Katz, Katz, Korman & ~~K~~ Peleg studied k-connectivity: $\Omega(k \lg n)$ bit lower bound
- For our problem, k=1

Thm: Label space $\sum_{i=1}^{n} \lfloor n/i \rfloor$ necessary and sufficient.

With Care

Can support constant time equivalence class queries using labels of

$\lg n + \lg\lg n + 2$ bits.

(Put elements in appropriate n/i size buckets; need care with "breakpoints" between buckets)

Under this model $\lg n + \lg\lg n - \Theta(1)$ are necessary (from previous theorem)

Our Real Problem Labels 1 … n

- This means we have to store a data structure that distinguishes between partitions:

Hardy-Ramanujan formula $\approx \frac{1}{4n\sqrt{3}} e^{(\pi\sqrt{2n/3})}$

Taking lg

Lower Bound: $((\pi\sqrt{\frac{2}{3}}\lg e)\sqrt{n}$ – lg n +O(1)) bits

$O(n^{1/2})$ bit Structure $O(\lg n)$ Time

Elements of same class numbered consec.

Classes of same size ordered consecutively

- k distinct classes sizes: s_i
- n_i classes of size s_i
- $\gamma_i = s_i n_i$ elements in a class of size s_i
- Order by γ_i (non-decreasing)
- Note # class sizes $k \leq (2n)^{1/2}$

The Structure

- We store 2 sequences ($i=1,k$)

S: $\delta_i = \gamma_i - \gamma_{i-1} = s_i n_i - s_{i-1} n_{i-1}$ $(s_0 n_0 = 0)$

M: n_i

And Shadows – bit vector 1 indicates start of new term (play a rank/select game on these)

Also store $\sum_{j=1}^{i} s_j n_j$ "occasionally"

Finding a Class

- Store $\sum_{j=1}^{i} s_j n_j$ for every lg nth i value

- So, given x, we find its class
  - Binary search to find the right lg n-block
  - Sequential search in lg n-block to find right class size
  - Compute class in size group

Speeding Up Search

For $\Theta(n^{1/2} \lg n / \lg \lg n)$ bits

And $\Theta(\lg \lg n)$ time

- Store sequences lg lg n apart
- Use y-fast trees

Getting to Constant Time

But $O(n^{1/2} \lg n)$ bits

More work and more details.

Key point is computing (integer) square roots in constant time, by table lookup and O(1) extra work

(Curiously avoiding the lg lg n iterations of Newton…also used by Rajeev)

Updates?

Can support unions .. OK, simplicity

Space: $O(n^{1/2} \lg n)$ bits

Worst case merge: $O(\lg n/\lg \lg n)$ time

Amortized query time: $O(\alpha(n))$

How?

Multiple copies, old, updating, new

"Time sharing" etc

That's All Folks