Integer Priority Queues with Decrease Key in Constant Time and the Single Source Shortest Paths Problem

Mikkel Thorup

AT&T Labs—Research

Based on:
Mikkel Thorup. Integer priority queues with decrease key in constant time and the single source shortest paths problem. *Journal of Computer and System Sciences*, 69(3):330–353, 2004. Announced at STOC'03.

The short story

Directed graph G, n nodes, m edges
with integer weights in $[0, C)$.

We solve single source shortest path problem in time

$$O(m + n \log\log n) \text{ or } O(m + n \log\log C)$$

Ahuja, Mehlhorn, Orlin, and Tarjan (1990):

> *Based on van Emde Boas' $O(m \log\log C)$ one might hope for $O(m + n \log\log C)$.*
>
> They got $O(m + n(\log C)^{1/2})$.

Previous best $O(m + n(\log C)^{1/4-\varepsilon})$
by Raman (1997) using randomization.

History in terms of C
—all using integer RAM model

$O(m + nC)$ Dial 1969

$O^r(m \log\log C)$ van Emde Boas' data structure 1977

$O(m + n \log C)$ Denardo and Fox 1979

$O(m + n(\log C)^{1/2})$ Ahuja, Mehlhorn, Tarjan 1990

$O^r(m + n(\log C)^{1/3+\varepsilon})$ Cherkassky, Goldberg, Silverstein 1997.

$O^r(m + n(\log C)^{1/4+\varepsilon})$ Raman 1997.

$O(m + n \log\log C)$ this paper

History in terms of n

$O(m + n^2)$ Dijkstra 1959

$O(m \log n)$ William's heap 1964

$O(m + n \log n)$ Fredman and Tarjan 1984

—now switch to integer RAM model

$O(m + n(\log n)/(\log \log n))$ Fredman and Willard 1993

$O^r(m + n(\log n)^{1/2+\varepsilon})$ and $O^r(m(\log \log n))$ Thorup 1995

$O^r(m + n(\log n)^{1/3+\varepsilon})$ Raman 1997

$O(m + n \log \log n)$ this paper

Technical contribution

Fibonacci heap style priority queue Q
over up to n keys a with values $v(a) \in [0, N)$

find-min(Q): returns min-key in Q
insert(Q, a): adds a to Q
dec-key(Q, a, x): sets $v(a) := v(a) - x$
delete(Q, a) deletes a from Q

All operations but delete in constant time.

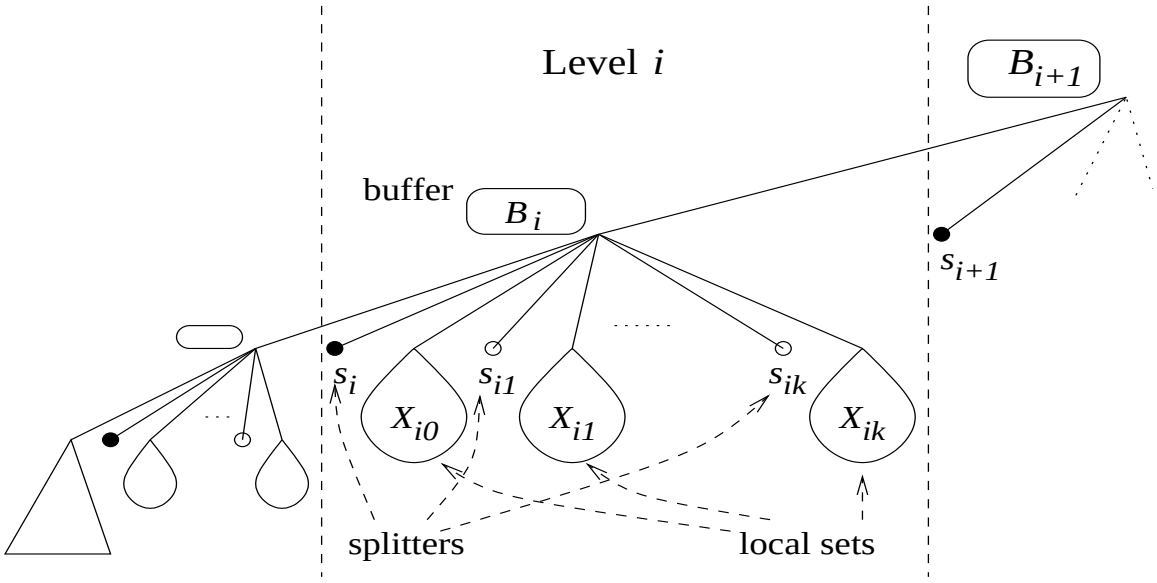
Delete time $d \rightarrow$
single source shortest paths time $O(m + nd)$

Fibonacci heap gets delete time $d = O(\log n)$

For integers, we get delete time $d = O(\log \log n)$

Structure of our priority queue

left branch of buffered exponential search tree



$$n_i = (\log^2 n)^{(5/4)^i}$$

Level $i > 0$ has

- $\Theta(n_i)$ keys
- $\Theta(n_i^{1/3})$ splitters and local sets
- local set size $\Theta(n_i^{2/3})$
- buffer size $O(n_i^{1/2})$

and level $i-1$ has $\Theta(n_i^{4/5})$ keys

Lemma (Han&Thorup FOCS'02)
Split q keys over $q^{1-\varepsilon}$ splitters in linear time
$\downarrow$
When B_i full with $|B_i| = \Theta(n_i^{1/2})$, split over $\Theta(n_i^{1/3})$ splitters in linear time.
$\downarrow$
Using buffers, distribute keys over local sets in $O(1)$ time per key.

The $O(1)$ includes re-balancing between local sets:

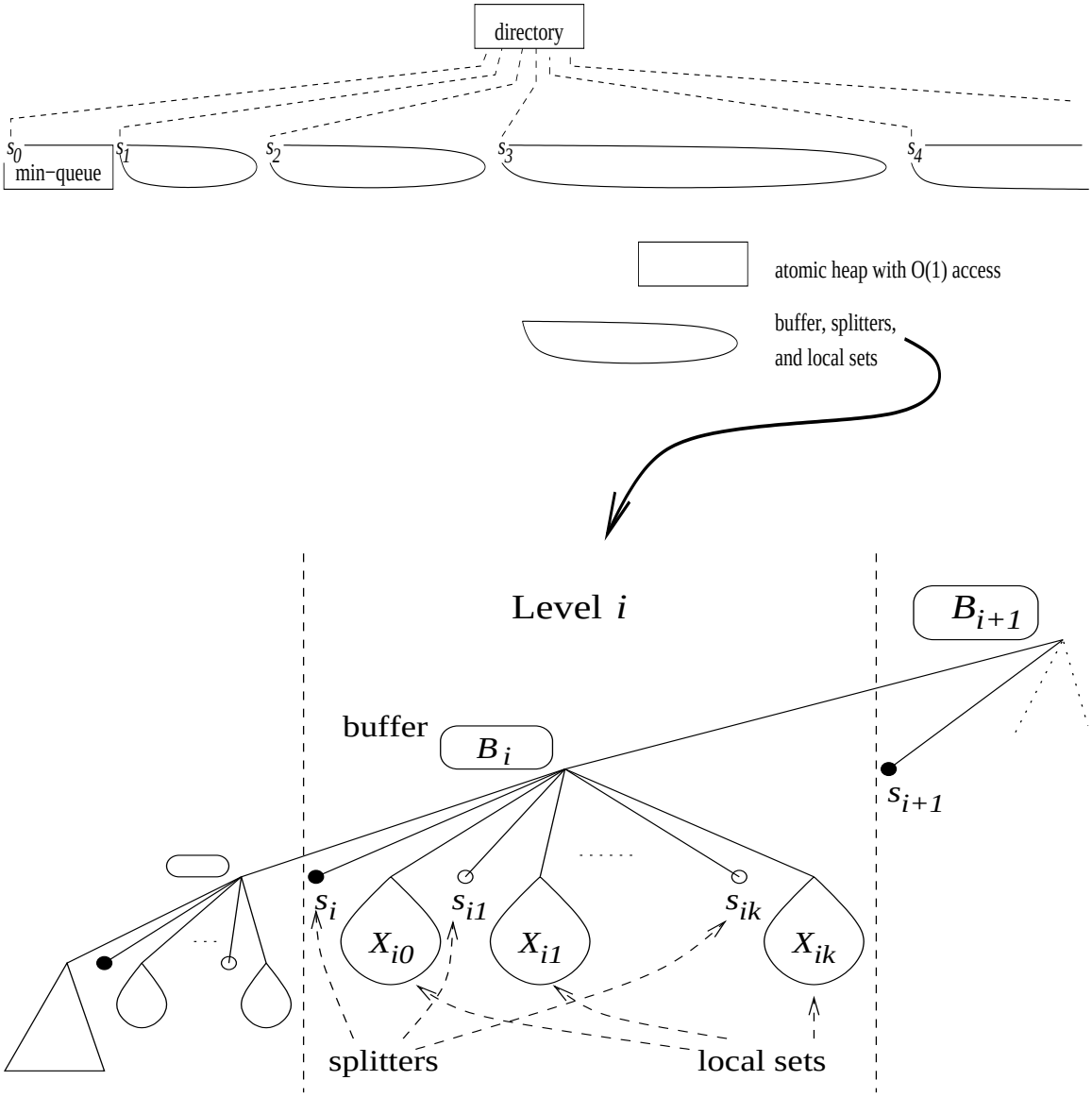
- Too much added to a local set pays split.

- Too much removed from a local set pays for merge or exchange with neighbor.

With $O(\log\log n)$ levels, easy priority queue with insert and delete in $O(\log\log n)$ time ...
but that's known :-(

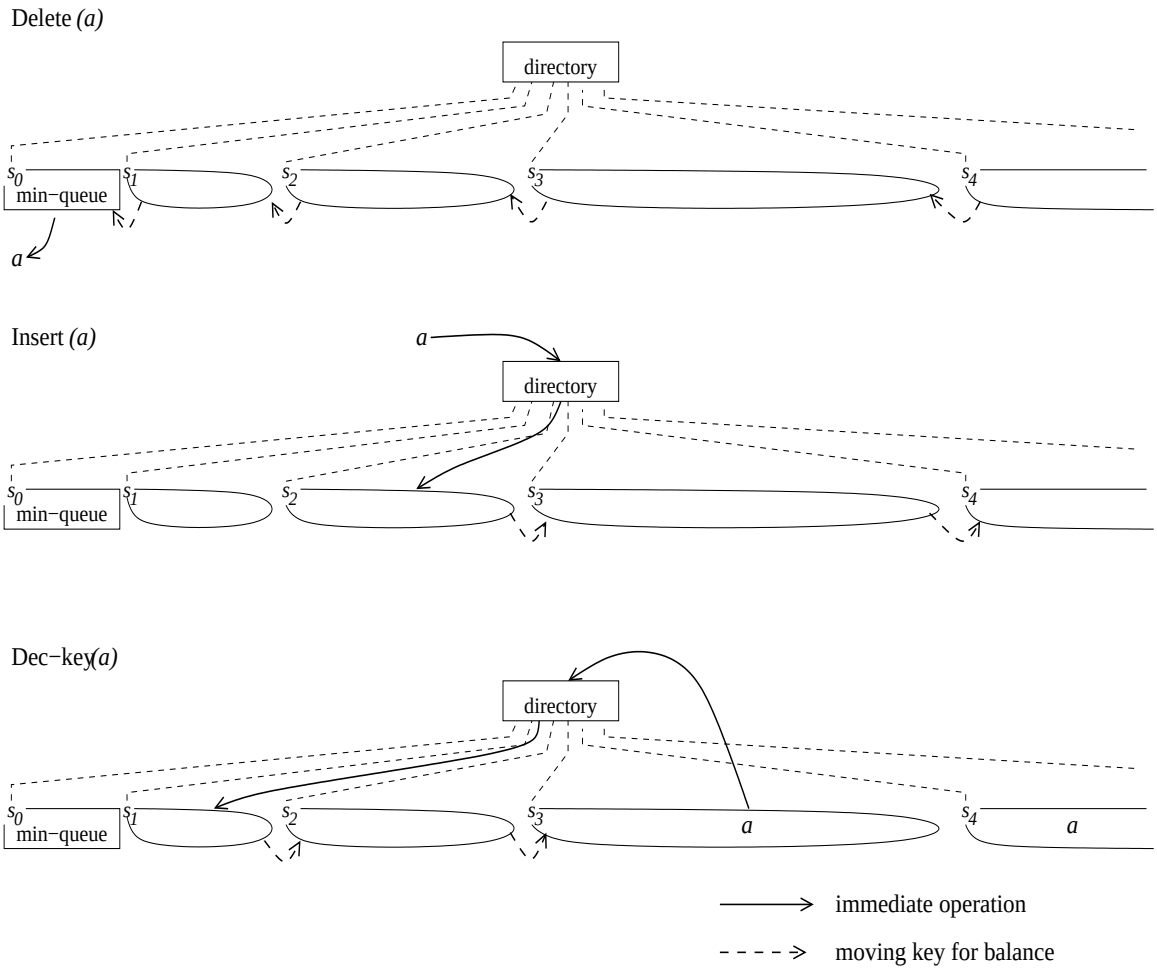
What about dec-key?

Global-local view



Using min-queue, find-min in $O(1)$ time

Updates

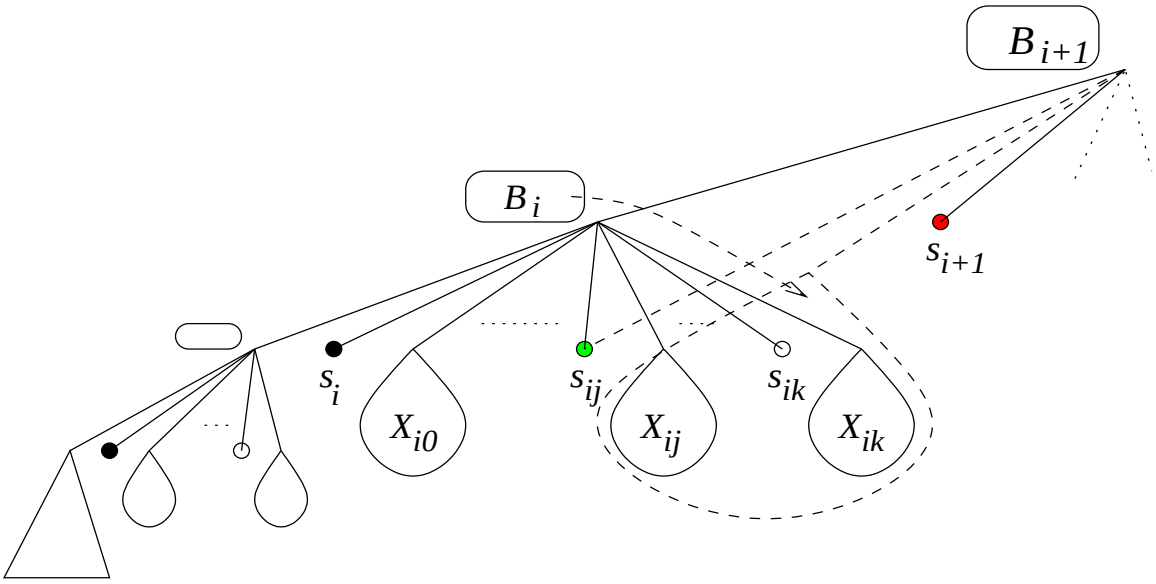


Moving key down in constant time
$\rightarrow$ delete in $O(\log \log n)$ time

If we can move keys up in $O(1/n_i^{1/3})$ time
$\rightarrow$ insert and dec-key time

$$O(1 + \sum_i n_i^{-1/3}) = O(1)$$

Moving up



Collect local level i sets creating a new local level $i + 1$ set

Preceding local splitter new level splitter s_{i+1}

Include keys from B_i bigger than new s_{i+1}

Analysis

Move up $\Theta(n_{i+1}^{2/3}) = \Theta(n_i^{5/6})$ keys

Collect $\Theta(n_i^{5/6}/n_i^{2/3}) = \Theta(n^{1/6})$ local level i sets in $\Theta(n_i^{1/6})$ time

Scan level i buffer in $O(n_i^{1/2})$ time

Moving up in $O(n_i^{1/2}/n_i^{5/6}) = O(n^{-1/3})$ time per key

⁝ details and de-amortization in journal paper.

Theorem Priority queue over n integer keys in $[0, N)$ supporting find-min, insert, and dec-key in constant time, and delete in time $O(\log\log n)$ or $O(\log\log N)$ ■

Corollary For directed graph with n nodes and m edges with integer weights in $[0, C)$, single source shortest path in time $O(m+n(\log\log n))$ or $O(m + n(\log\log C))$ ■

Concluding remarks

Without multiplication, monotone priority queue with delete in $O((\log\log n)^{1+\varepsilon})$ time, hence single shortest paths in time

$$O(m + n(\log\log n)^{1+\varepsilon})$$

Open problems

(1) Without dec-key in constant time, using randomization, we can insert and delete in $O(\sqrt{\log\log n})$ expected time (Thorup FOCS'02).

Can we get such delete time with insert and dec-key in constant time?

(2) A general reduction (Thorup FOCS'02) shows that the per key costs of sorting and priority queues are asymptotically equivalent.

Is there a general construction augmenting a priority queue supporting insert and delete with dec-key in constant time?

Above **(2)** $\Rightarrow$ **(1)**

Exercises for directed SSSP

- The result uses that we can split q keys over $q^{1-\varepsilon}$ splitters in linear time. What happens if we could only handle $\log q$ splitters? or if we could handle $q/\log n$ splitters.

- Currently the directory is implemented with an atomic heap. Show that we can implement a directory with splitters instead.

- Is there any reasonable way of using the ideas in practice?