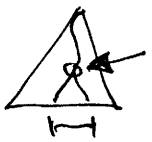
Dynamization (updates)

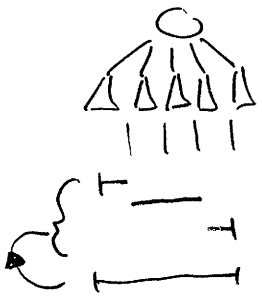
I) Update base tree
- insert/delete two endpoints
- rebalance [incl. rebuilding of side structures]

II) Update secondary structures
- insert/remove interval in one node

II): i) Find relevant node [highest node where interval endpoints go to different subtrees (~ first part of a range search (or delete from))]



ii) Insert in 3 side structures:
a) two single slab list
b) one multislab list (or zero)



a) just update the single slab list (B-trees).

b) For each of the $\binom{k}{2}$ multislab list, keep a counter in node storing its size.

⇒ we know if list is small or not.
(i.e. is a B-tree, or is part of the underflow structure).

If a list changes status (small/not small), it should enter/leave the underflow structure. Price for this:

Scan underflow str. : $O\left(\frac{B^2}{B}\right) = O(B)$

Rebuild —"— : $O\left(\frac{B^2}{B} \cdot \log_{M/B}\left(\frac{B^2}{M}\right)\right)$

Build B-tree : —"—

We will assume $M \geq B^{1+\varepsilon}$. [By Fact on page 12]

Then $\log_{M/B}(x) \leq \log_{B^\varepsilon}(x)$

$$= \frac{\log_2(x)}{\log_2(B^\varepsilon)} = \frac{1}{\varepsilon}\frac{\log_2(x)}{\log_2(B)} = \frac{1}{\varepsilon}\cdot\log_B(x)$$

So $O\left(\frac{B^2}{B}\cdot\log_{M/B}\left(\frac{B^2}{M}\right)\right) = O(B\cdot 1) = O(B)$.

Idea 8 Move multislab list from B-tree to underflow str. when size $\leq B/2$

Move multislab list to B-tree from underflow str. when size $\geq B$.

Idea 9 Keep a single block holding info on the last $\leq B$ (single element) updates to underflow str.

If an update in part II, ii), b) makes a multislab list change status, then by Idea 8, at least $B/2$ such updates took place on this multislab list since last time it changed status. Thus, the $O(B)$ work can be covered by charging these updates $O(1)$ I/O's.

The underflow structure is static. By Idea 9, we can store info on the last $\leq B$ updates to it. These updates are not done, but stored in $O(1)$ block. When the search passes the node, the answer from the query to the underflow structure is modified using this info [once the $O(1)$ info blocks are read into memory, this cost zero additional I/O's]. When these info blocks get full, we rebuild the underflow structure, incorporating these updates. This takes $O(B)$ I/O's, which can be covered by charging $O(1)$ to each of the updates stored.

Cost of II):

i) $O(\log_B(N))$ search in base tree

ii) a) $O(\log_B(N))$ update list/B-tree

ii) b) $O(\log_B(N))$ update large list/B-tree

+ $O(1)$ amortized maintain underflow structure (cf. page 16 and 17).

Total: amortized $O(\log_B(N))$.

I): i) Insert two new endpoints [or delete] in leaves.
[Standard search + update leaves]

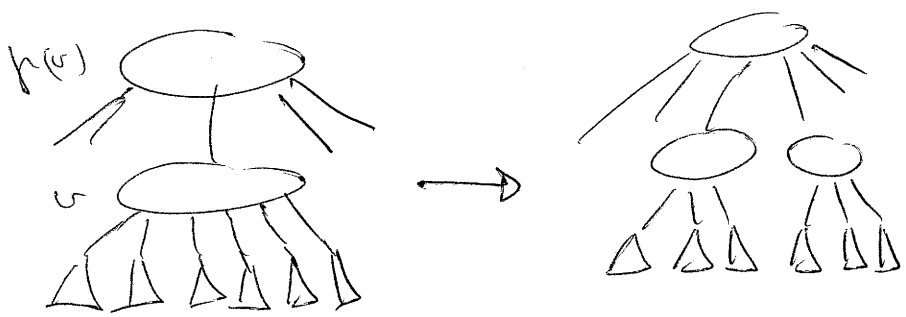
ii) Rebalance if needed.

Idea 10: Use weight-balanced B-trees [with fan-out parameter $\sqrt{B}$] as base tree.

[Recall: when a node v triggers a rebal. operation, at least weight(v) updates took place below v since it last took part in rebal. operation]

Thus, if an update [of base tree] is charged $O(1)$ pr. node it passes (ie., is charged $O(\log_B(N))$ in total), a rebalancing operation can cost $O(w(v))$ I/O's and still be covered by this charging.

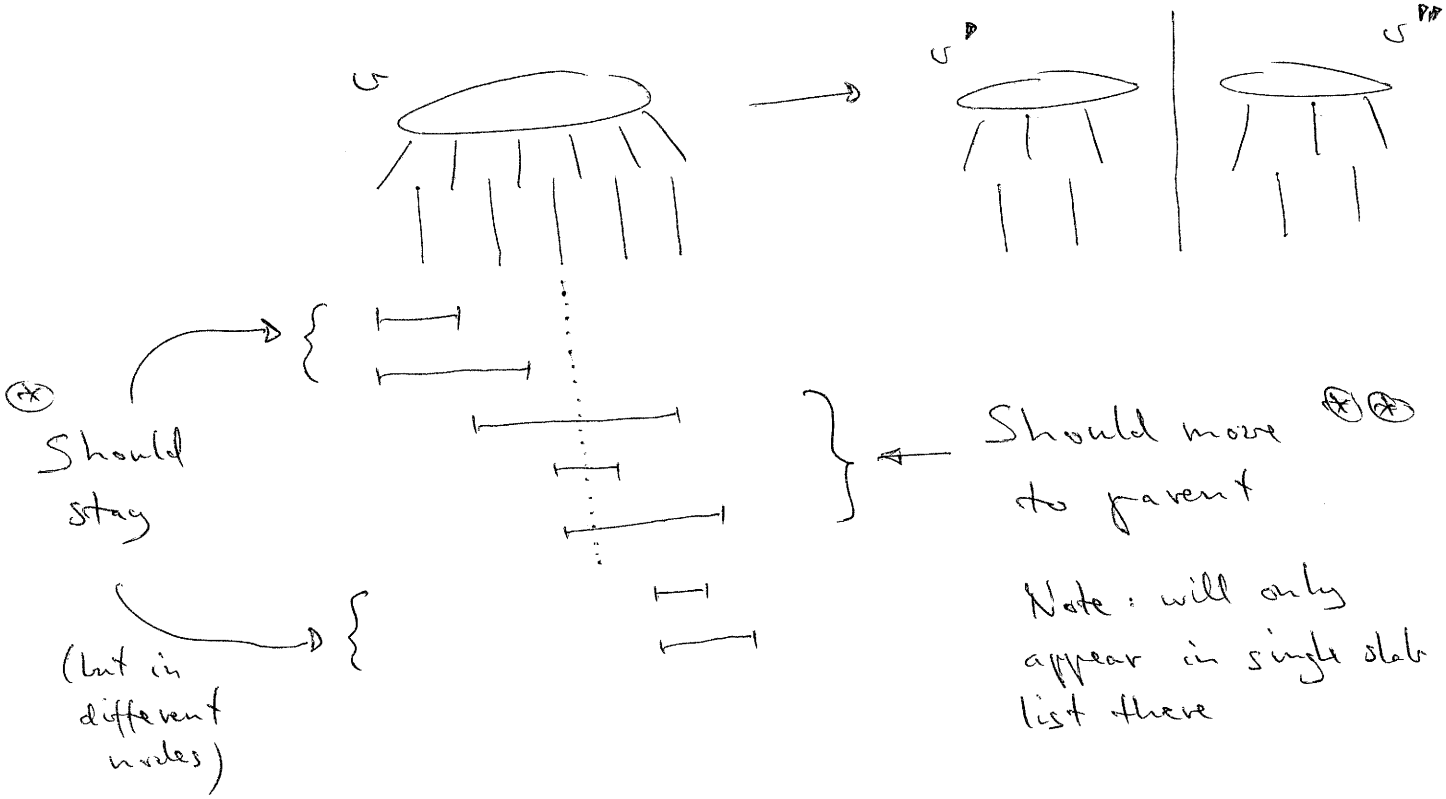
Consider a SPLIT at a node v.



The Only intervals to change node are the one residing at v [some should now reside at v's parent $p(v)$].

The only side structures to change are those of v and $p(v)$. [rest of nodes have unchanged subtree contents].

In v:



Recall: an interval is stored 3 times: a left single slab list, a right single slab list, and (possibly) a multi slab list (either full or in underflow str.)

Single slab lists: scan the "left" lists from left to right wrt. v's children, split into i) new left lists for v' and v'' and ii) the (only) new left single list in parent. Note: the latter is generated in correct sorted order.

Do the same for the "right" single slab lists.

[The part in ii) is just new contents for parents single slab list (not the entire such list)].

The full multislab list: should just be moved to right new node (v' or v'') [for ⊛] or be deleted [for ⊛⊛].

The underflow structure: Should be scanned, its ⊛ contents divided in two (for v' and v'') and its ⊛⊛ contents deleted. Then two new structures (for v' and v'') should be built.

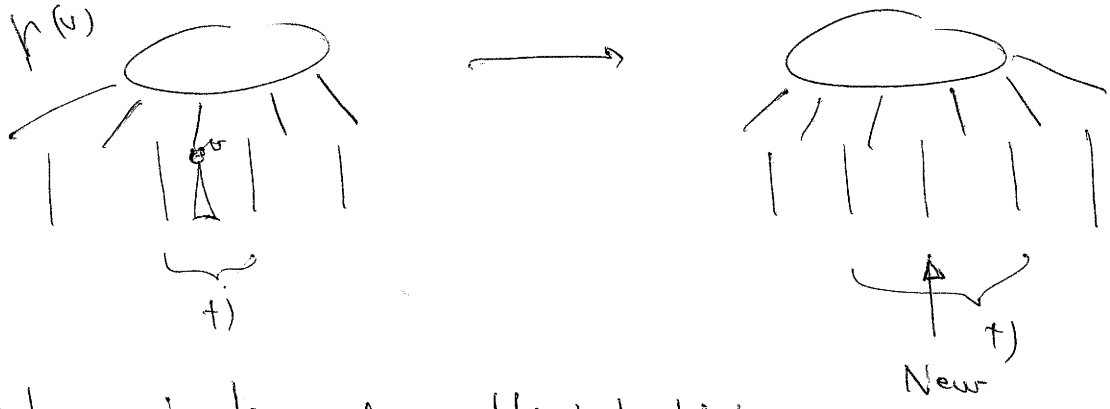
By Idea 6, the number of intervals stored at v is $O(w(v))$.

Since a rebuild of the underflow structure U takes [Fact p. 12] $O\left(\frac{|U|}{B} \log_{M/B}\left(\frac{|U|}{M}\right)\right) = O\left(\frac{|U|}{B}\right)$

[if we assume $M \geq B^{1+\varepsilon}$, and recall $|U| \leq B^2$],

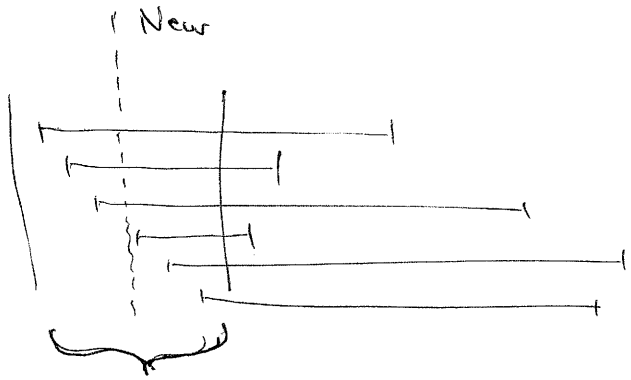
this is all $O\left(\frac{w(v)}{B}\right)$ I/O's.

In parent $p(v)$ of v :



Only single and multi slab lists with an endpoint in $t)$ need change.

The new contents from $p(v)$ in these can be found during a scan of the two single slab lists in $t)$. (recall, an endpoint knows about the other endpoint of the interval):



This scan is $O(\frac{w(v)}{B})$, since all relevant intervals have an endpoint in v's subtree [as noted above], and endpoints are unique [by Idea 6].

There are 4 (2×2) new single slab lists, and two of these should receive new intervals

from the split v (those denoted ⊛⊛ on p. 20). This is a merge (of sorted lists) and is again $O(w(v)/B)$.

However, the affected multislab lists in $\rho(v)$ [deletion of old, creation of new] could affect the underflow structure U' of $\rho(v)$. So it needs to be rebuilt. This takes $O(|U'|/B)$ I/O's.

We note that $|U'|$ is $O(\max\{w(\rho(v)), B^2\})$ and $w(\rho(v)) = \Theta(\sqrt{B} \cdot w(v))$.

Thus, on the two lowest levels, this cost is $O(w(v)/\sqrt{B})$, on the rest (where $w(v) \geq B^2$) it is $O(w(v)/B)$.

Since we are using weight bal. B-trees for the base tree, we can charge $\Theta(w(u))$ updates that passed u.

So charging each update $O(1)$ per node passed will cover all (more than 1) such rebal. expenses.

This is amortized $O(\log(N))$ per update.

□