Soft-heaps according to Kaplan and Zwick

A node v stores:

- $\ell(v)$, $r(v)$ – left and right child
- $rank(v)$ – rank; never changes; ranks of children are smaller by 1
- $list(v)$ – a list of items stored in this node
- $ckey(v)$ – a key common to all keys in $list(v)$
- $size(v)$ – planned size of $list(v)$

We build trees out of the nodes:

- $ckey$s of nodes are heap-ordered
- $rank(T)$ and $ckey(T)$ inherited from the root node

A heap $\mathcal{H}$ contains a list of trees in order of increasing rank. A rank of the heap is a maximum of tree ranks. For each tree T, we store:

- $sufmin(T)$ – pointer to the tree with minimum $ckey$ following T

Setup of parameters:

- $r = \lceil \log_2(1/\varepsilon) \rceil + 5$
- $s_k = 1$ for $k \le r$, else $s_k = \left\lceil \frac{3s_{k-1}}{2} \right\rceil$
- $size(v) = s_k$, where $k = rank(v)$

Observation: $(3/2)^{k-r} \le s_k \le 2 \cdot (3/2)^{k-r} - 1$ pro $k \ge r$.

Sift(v)**:**

1. While $|list(v)| < size(v)$ and v is not a leaf:
2. If $\ell(v) = \emptyset$ or $ckey(\ell(v)) > ckey(r(v))$: $\ell(v) \leftrightarrow r(v)$.
3. Move all items from $list(\ell(v))$ to $list(v)$.
4. $ckey(v) \leftarrow ckey(\ell(v))$.
5. If $\ell(v)$ is a leaf, remove it; else $Sift(\ell(v))$.

Invariant L: $size(v)/2 \le |list(v)| \le 3 \cdot size(v)$ for nodes of rank at least r; otherwise $|list(v)| \ge 1$.

Invariant R: #nodes of rank $k \le n/2^k$.

Invariant C: #corrupted items $\le \varepsilon n$.

Potential:

- Heap of rank k contributes $k+1$.
- A tree with root x contributes $(r+2) \cdot del(x)$, where $del(x)$ is the number of items deleted from $list(x)$ since the previous call to $Sift$ or creation of the root.
- A root of rank k contributes $k+7$.
- Every other node contributes 1.