Merging with the help of a cover.

Let c be a positive integer. A sorted sequence X be called a c-cover of another sorted sequence Y if Y has at most c elements between each pair of consecutive elements in X_∞

$= (-\infty, X, +\infty)$

X is a cover of Y

More precisely, given any two consecutive elements α and β of X_∞, then the set $\{y_i \mid y_i \in Y \text{ and } \alpha < y_i \leq \beta\}$ has at most c elements.

Ex $X = (-1, 15, 21, 23)$ is a 4-cover of $Y = (-10, -5, -2, -1, 4, 5, 10, 12, 20, 22, 26, 31, 50)$

4 elements lie between -1 & 15.

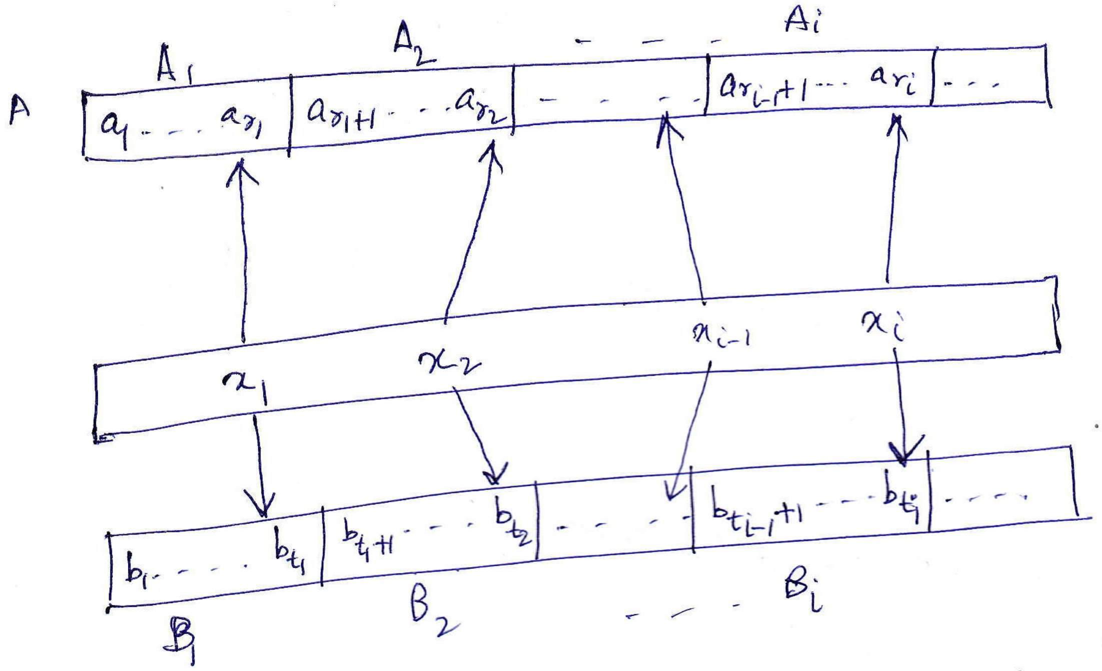
Result Let A and B be two sorted sequences of lengths n and m respectively.

Let X be a C-cover of A and B for some constant c.

If rank(X : A) and rank(X : B) are known, then the problem of merging A and B can be solved in O(1) time, using O(|X|) operations.

Let $X = (x_1, \dots x_s)$, $rank(X : A) = (r_1 \dots r_s)$

$rank(X : B) = (t_1 \dots t_s)$.



Let, $a \in A_i$. $Rank(a : B) = t_{i-1} + rank(a : B_i)$

$(\because b_{t_{i-1}} \leq x_{i-1} < a_{r_{i-1}+1} \leq a \leq a_{r_i} \leq x_i < b_{t_i+1})$

Hence, the problem reduces to determine $rank(a : B_i)$

But, $|B_i| \leq c$, since X is a c-cover of B.

$\Rightarrow$ a can be ranked in B_i in $O(1)$ sequential time.

Thus array $rank(A : B)$ can be found in $O(1)$ time.

Also number of operations is linear.

Merge Sort: $X = (12, -5, -7, 51, 6, 28, 3, -8)$.

Two parallel implementations of merge-sort strategy on the PRAM model.

1st algo:

$T(n) = T(n/2) + O(\log\log n) \Rightarrow T(n) = O(\log n \log\log n)$.

$W(n) = 2W(n/2) + O(n) \Rightarrow W(n) = O(n \log n)$.

Simple Merge Sort.

Input: An array X of order n, where $n = 2^l$ for some integer l.

Output: A balanced binary tree with n leaves, s.t. for each $0 \le h \le \log n$, $L(h,j)$ contains the sorted sequence consisting of the elements stored in the subtree rooted at node (h,j), for $1 \le j \le n/2^h$.

That is node (h,j) contains the sorted list of the elements $X(2^h(j-1)+1), X(2^h(j-1)+2), \ldots X(2^h j)$.

For each element v of the balanced tree T, Algo generates the sorted list L[V] consisting of the elements stored in the subtree rooted at v.

Run time: $O(\log n \log\log n)$.

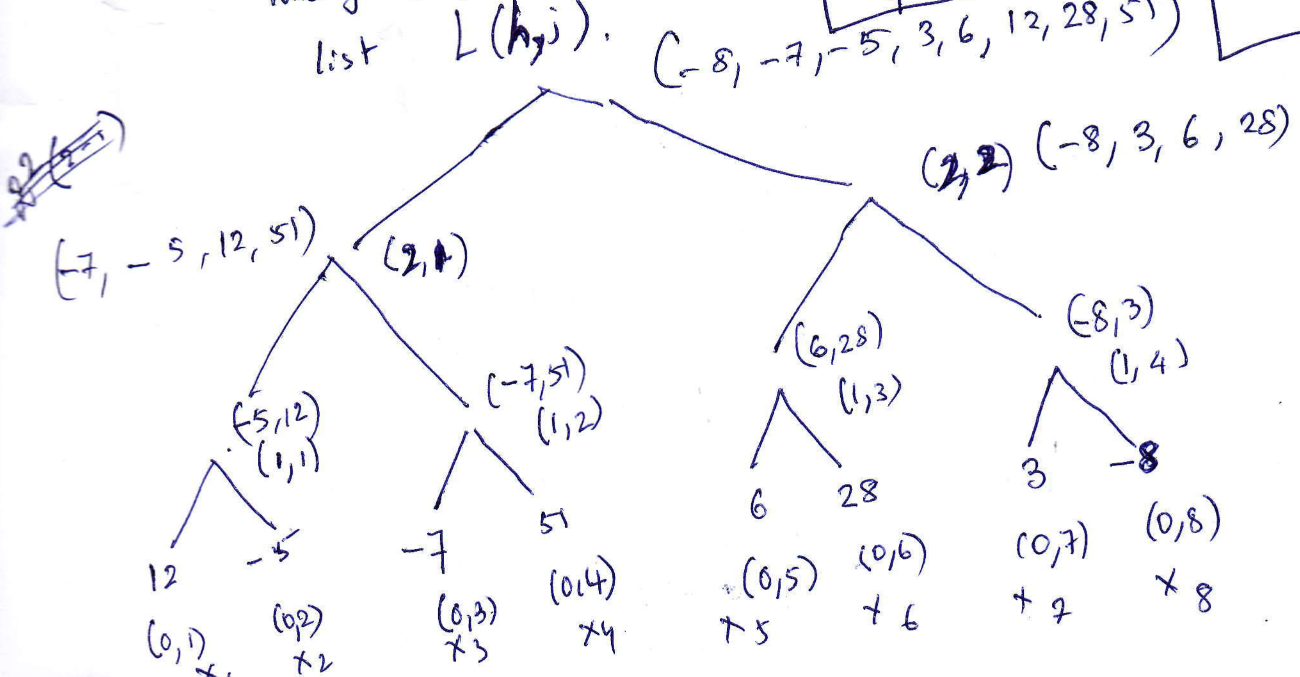
Work: $O(n \log n)$

Proof: Total # of elements involved at each level is n ⇒ each iteration takes $O(\log\log n)$ time, using a total of $O(n)$ operations.

Begin

for $1 \le j \le n$ parallel do
  Set, $L(0,j) = X(j)$

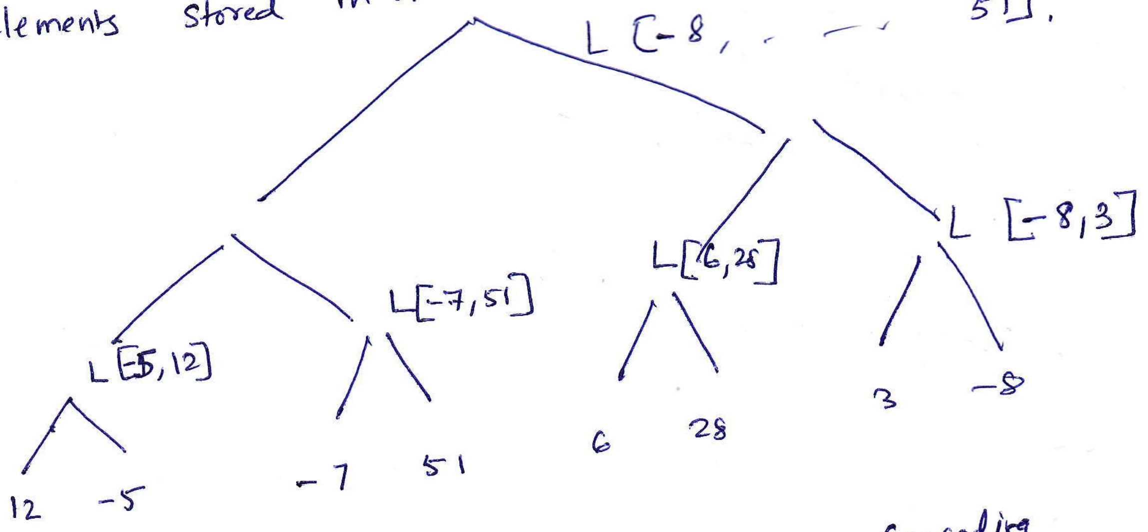
for $h = 1$ to $\log n$, do
  for $1 \le j \le n/2^h$ parallel do
    Merge $L(h-1, 2j-1)$ and $L(h-1, 2j)$ into the sorted list $L(h,j)$.



Pipelined Merge-Sort Algorithm.

~~Assume~~ Let, T be a binary tree such that each leaf u contains an unsorted list $A(u)$ drawn from a linearly ordered set.

We consider the problem, for each internal node v, determine the sorted list $L[v]$ that contains all the elements stored in the subtree rooted at v.



Cole's Parallel Algorithm. (pipelined or Cascading merge sort).

- Compute $L[v]$ over a number of stages.
  - at stage s, $L_s[v]$ is an approximation of $L[v]$ that is improved at next stage $(s+1)$.
  - a <u>sample</u> (every cth element) of $L_s[v]$ is propagated <u>upward</u> to be used to improve approximations above.

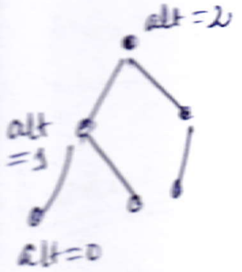
<u>def</u> c-sample of a sorted list L, denoted by $\text{Sample}_c(L)$ is the sorted sublist of L consisting every cth element of L.

i.e if $L = (l_1, l_2, \ldots) \Rightarrow \text{Sample}_c(L) = (l_c, l_{2c}, \ldots)$ (starting from c).

• altitude : $alt(v) = h(T) - level(v)$.

$alt(root) = h(T)$.

(length of path from root to node v)



$L_s[v]$ is updated over stages s, s.t. $alt(v) \leq s \leq 3\,alt(v)$

initialize

$L_0[v] = \phi$ if v is internal.

$L_0[v]$ = value at leaf v if v is leaf.

The algorithm updates $L[v]$ at stages $alt(v) \leq s \leq 3 \cdot alt(v)$

• after stage $3\,alt(v)$, $L[v]$ is full or finished.

i.e $L_s[v] = L[v]$, when $s > 3\,alt(v)$

⇒ algorithm runs in $3 \cdot h(T)$ stages which is $O(\log n)$

if each stage runs in $O(1)$ time.

At stage s :

for all active nodes parallel do /* during stage s if $alt(v) \leq s \leq 3\,alt(v)$. */

1. let u & w be children of v

$L'_{s+1}[u] = \text{Sample}(L_s[u])$

$L'_{s+1}[w] = \text{Sample}(L_s[w])$

2. Merge $L'_{s+1}[u]$ and $L'_{s+1}[w]$ into sorted list $L_{s+1}[v]$.

end parallel do.

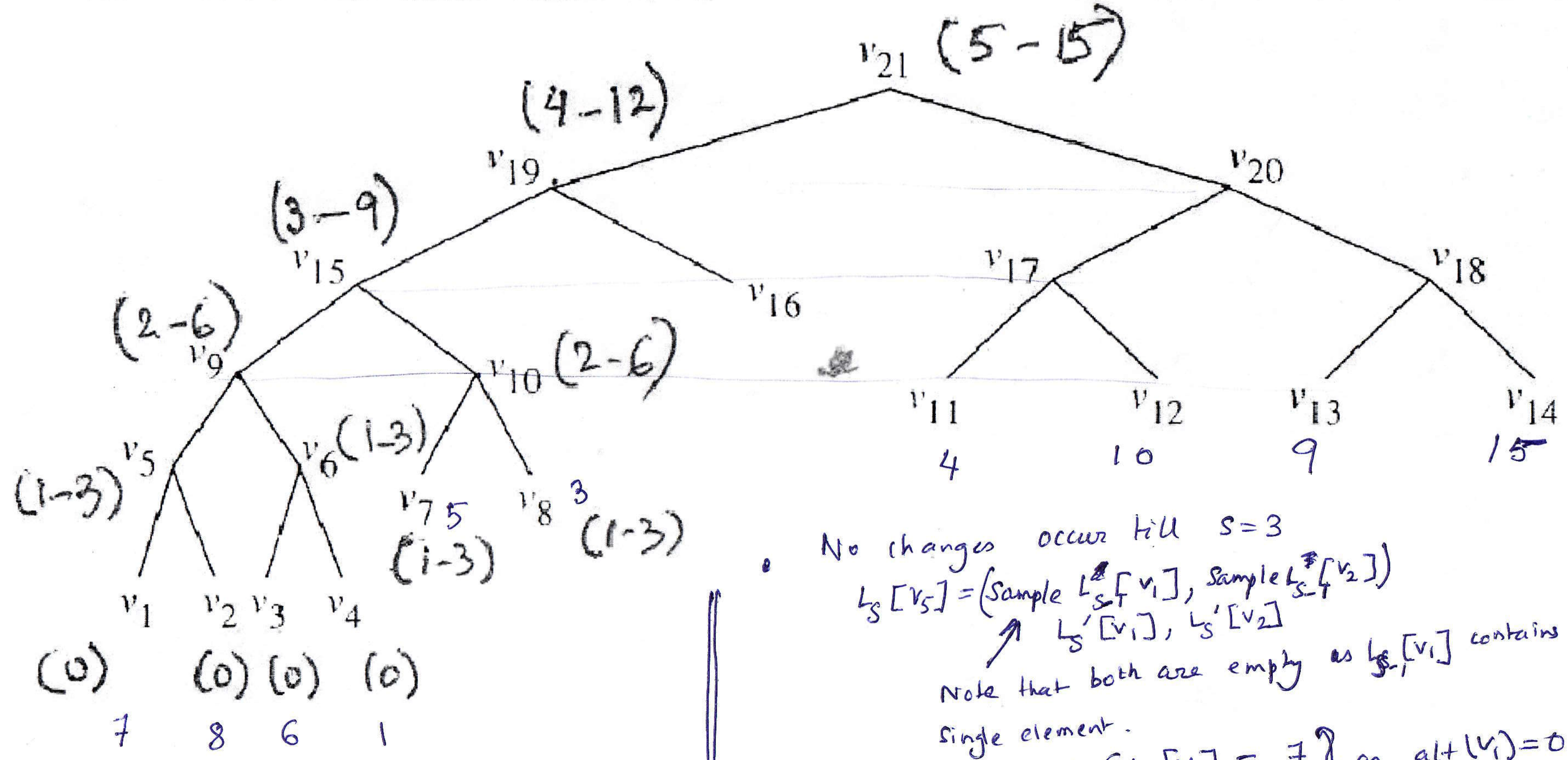
where

$$\text{Sample}(L_s[x]) = \begin{cases} \text{sample}_4(L_s[x]) & \text{if } s \leq 3 \cdot alt(x) \ (L(x) \text{ not full yet}) \\ \text{sample}_2(L_s[x]) & \text{if } s = 3alt(x)+1 \ (\text{1st iteration } L(x) \text{ full}) \\ \text{sample}_1(L_s[x]) & \text{if } s = 3alt(x)+2 \ (\text{2nd iteration } L(x) \text{ full}). \end{cases}$$

sample_4: every 4th element L_s

sample_2: every other element L_s.

sample_1: every element L_s



- No changes occur till $s = 3$

$L_3[v_5] = (\text{Sample } L_{s-1}[v_1], \text{Sample } L_{s-1}[v_2])$

$L_s'[v_1], L_s'[v_2]$

Note that both are empty as $L_{s-1}[v_1]$ contains single element.

At $s = 3$, $\left\{ \begin{array}{l} L_2[v_1] = 7 \\ L_2[v_5] = 8 \end{array} \right\}$ as, $alt(v_1) = 0$, $alt(v_2) = 0$.

$\therefore L_3[v_5] = (7, 8)$.

nodes at alt 2 becomes full.

|| Root v_{21} is active for all stages s, $5 \le s \le 15$.
$L_s[v_{21}]$ remains empty until stage $s=13$.
(since till stage $s=12$, v_{19} & v_{20} contains less than 4 elements).

v	s = 0	s = 3	s = 5	s = 6	s = 8	s = 9	s = 11	s = 13
1	(7)	(7)	(7)	(7)	(7)	(7)	(7)	(7)
2	(8)	(8)	(8)	(8)	(8)	(8)	(8)	(8)
3	(6)	(6)	(6)	(6)	(6)	(6)	(6)	(6)
4	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)
5	0	(7, 8)	(7, 8)	(7, 8)	(7, 8)	(7, 8)	(7, 8)	(7, 8)
6	0	(1, 6)	(1, 6)	(1, 6)	(1, 6)	(1, 6)	(1, 6)	(1, 6)
7	(5)	(5)	(5)	(5)	(5)	(5)	(5)	(5)
8	(3)	(3)	(3)	(3)	(3)	(3)	(3)	(3)
9	0	0	(6, 8)	(1, 6, 7, 8)	(1, 6, 7, 8)	(1, 6, 7, 8)	(1, 6, 7, 8)	(1, 6, 7, 8)
10	0	0	0	(3, 5)	(3, 5)	(3, 5)	(3, 5)	(3, 5)
11	(4)	(4)	(4)	(4)	(4)	(4)	(4)	(4)
12	(10)	(10)	(10)	(10)	(10)	(10)	(10)	(10)
13	(9)	(9)	(9)	(9)	(9)	(9)	(9)	(9)
14	(15)	(15)	(15)	(15)	(15)	(15)	(15)	(15)
15	0	0	0	0	(5, 6, 8)	(1, 3, 5, 6, 7, 8)	(1, 3, 5, 6, 7, 8)	(1, 3, 5, 6, 7, 8)
16	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)
17	0	0	0	0	0	(4, 10)	(4, 10)	(4, 10)
18	0	0	0	0	0	(9, 15)	(9, 15)	(9, 15)
19	0	0	0	0	0	0	(3, 6, 8)	(1, 2, 3, 5, 6, 7, 8)
20	0	0	0	0	0	0	(10, 15)	(4, 9, 10, 15)
21	0	0	0	0	0	0	0	(5, 15)

After stage $s=13$, $L_s[v_{21}] = (L'_s[v_{19}], L'_s[v_{20}]) = (\text{sample}_4(L(v_{19})), \text{sample}_4(L(v_{20})))$

$= \text{sample}_4(1,2,3,5,6,7,8), \text{sample}_4(4,9,10,15)$

$= (5, 15).$

Correctness | Let v be an arbitrary node of the binary tree T. Then, at the end of stage $s = 3\,alt(v)$, v becomes full, i.e. $L_s[v] = L[v]$.

Proof by induction on $alt(v)$.

Basis. For all nodes with $alt(v) = 0$, i.e. leaves ~~$alt(v) = 0$~~. we have $L_0(v) = L[v]$ (trivially true).

Let v be a node with $alt(v) = k > 0$.

If v is a leaf, then $L_0[v] = L[v]$, there is nothing to prove.

Assume v is an internal node, with children u and w.

Clearly, $alt(u) = alt(w) = k-1$.

By induction hypothesis u and w will become full at stage s', s.t. $s' = 3(k-1)$

During stage $s'+1$, $L_{s'+1}[v] \leftarrow (L'_{s'+1}[u], L'_{s'+1}[w])$
$= (\text{Sample}(L'_s[u]), \text{sample}(L'_s[w]))$

But $\text{Sample}(L_s'[u]) = \text{Sample}_4(L_s'[u])$
$= \text{Sample}_4(L[u])$
[$\because$ at stage s $L[u]$ is full].

Likewise, $\text{Sample}(L_s'[w]) = \text{Sample}_4(L[w])$.

During stage $s'+2$, $\text{Sample}(L_{s'+1}[u])$ will contain every other element of $L[u]$.

During stage $s'+3$, $\text{Sample}(L_{s'+2}[u]) = L[u]$
Likewise, $\text{Sample}(L_{s'+2}[w]) = L[w]$.

$\therefore$ At stage $s'+3 = 3(k-1)+3 = 3k$.
$L'_{s+3}[v] = L[v]$. QED.

Correctness is not difficult.

More complicated is to see $O(1)$ time per stage

Exercise

Prove by induction: Let v be an arbitrary node of T and let $s \geq 1$. Then $|L_{s+1}[v]| \leq 2|L_s[v]| + 4$.

- **Remark:** For a given stage s, the total number of elements stored in all the active nodes of T is given by $n_s = \sum_{v \text{ active}} |L_s[v]| = \sum_{\lfloor s/3 \rfloor \leq alt(v) \leq s} |L_s[v]|$

Note that, if a node v is full, where $alt(v) = \lfloor s/3 \rfloor$

$$\sum_{alt(v) = \lfloor s/3 \rfloor} |L_s(v)| = n.$$

Consider, the active nodes at the level just above these full nodes, there can be at most $n/2$ elements & so. $(= n + \frac{n}{2} + \dots + 1)$.

Likewise $n_s = O(n)$.

// only essential step left is to show that the merging can be done in $O(1)$ time using $O(n_s) = O(n)$ operations. //

We have seen that the merging of two sorted lists A and B can be done optimally in $O(1)$ time ⊙ if we are given a c-cover X for A and B are given as input.

⊙ if rank (X : A) and rank (X : B) are given as input.

~~for each no.~~

~~We show that~~

We show that $L_s[v]$ is a 4-cover for $L'_{s+1}[u]$ and for $L'_{s+1}[w]$ where u and w are children of v.

- Also rank $(L_s[v] : L'_{s+1}[u])$ and rank $(L_s[v] : L'_{s+1}[w])$ can be generated efficiently.

We first show Sample $(L_{s-1}[v]) = L'_s[v]$ is a 4-cover of Sample$(L_s[v]) = L'_{s+1}[v]$.

i.e. $L'_s[v]$ is a 4-cover of $L'_{s+1}[v]$

Lemma Let v be an arbitrary node of T and let $s \geq 1$. Then $L'_s[v]$ is a 4-cover of $L'_{s+1}[v]$.

Defn Let $[a,b]$ be an interval with $a, b \in (-\infty, L'_s[v], +\infty)$. $[a,b]$ intersects $(-\infty, L'_s[v], +\infty)$ in k items (or there are k items in common) if the number of elements $x \in (-\infty, L'_s[v], +\infty)$ s.t. $a \leq x \leq b$ is equal to k.

Claim If $[a,b]$ intersects $(-\infty, L'_s[v], +\infty)$ in $k \geq 2$ items, then $[a,b]$ intersects $L'_{s+1}[v]$ in at most 2k items.

By induction on s.

Proof

$s = 1$. No list has more than 1 elements

Thus, $L_s'[v]$ and $L_{s+1}'[v]$ are both empty.

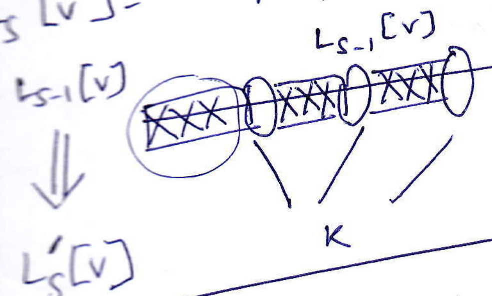
Assume for any stage $t < s$, any interval $[a', b']$ intersects $L_{t+1}'[v]$ in at most $2h$ items, where h is the number of items common between $[a', b']$ and $(-\infty, L_t'[v], +\infty)$

We show claim for s.

Let $[a, b]$ be an interval with K common items with $L_s'[v]$.

$s \le 3\,alt(v)$ ($s \ge 3\,alt(v)+1$ is straightforward as $L_{s-1}[v] = L_s[v] = L[v]$)

$L_s'[v] = \text{Sample}_4(L_{s-1}[v]) = \text{Sample}_4(L_{s-1}[v])$ $\sim 4K-3$.

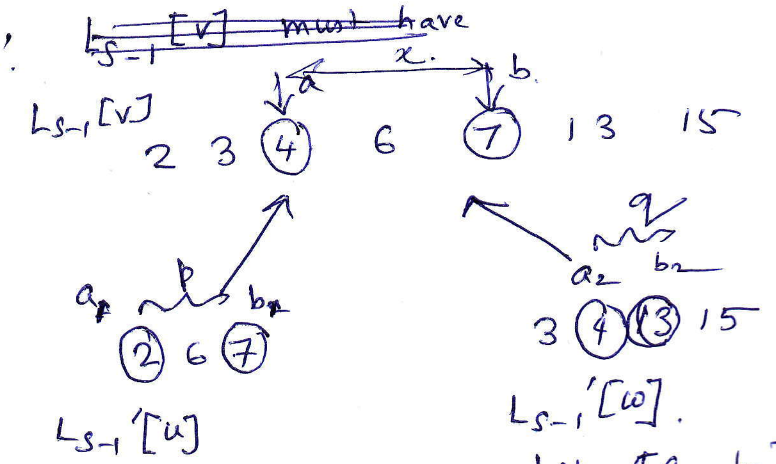


$\therefore$ $[a, b]$ intersects $(-\infty, L_{s-1}[v], +\infty)$ in $4K-3$ items.

Recall:

Recall, $L_{s-1}[v]$ is obtained by merging $L_{s-1}'[u]$ and $L_{s-1}'[w]$.

∴ ~~$L_{s-1}[v]$ must have~~



Let $[a_1, b_1]$ be the smallest interval containing $[a, b]$ st. $a_1, b_1 \in (-\infty, L_{s-1}'[u], +\infty)$.

Let, $[a_2, b_2]$ be the smallest interval containing $[a, b]$ st $a_2, b_2 \in (-\infty, L_{s-1}'[w], +\infty)$

A ∵ all elements are distinct, ~~p+q~~

if p is number of elements in common between $[a_1, b_1]$ and $(-\infty, L_{s-1}'[u], +\infty)$ and q is likewise between $[a_2, b_2]$ and $(-\infty, L_{s-1}'[w], +\infty)$,

$p + q \leq (4k-3) + 2$ (since two additional elements from $\{a_1, b_1, a_2, b_2\}$ are included).

∵ ~~a two~~ a, b must belong to the sublists, either two of $\{a_1, b_1, a_2, b_2\}$ are part of the merged list. ∴ 2 are extra

$p = 3, \quad q = 2.$

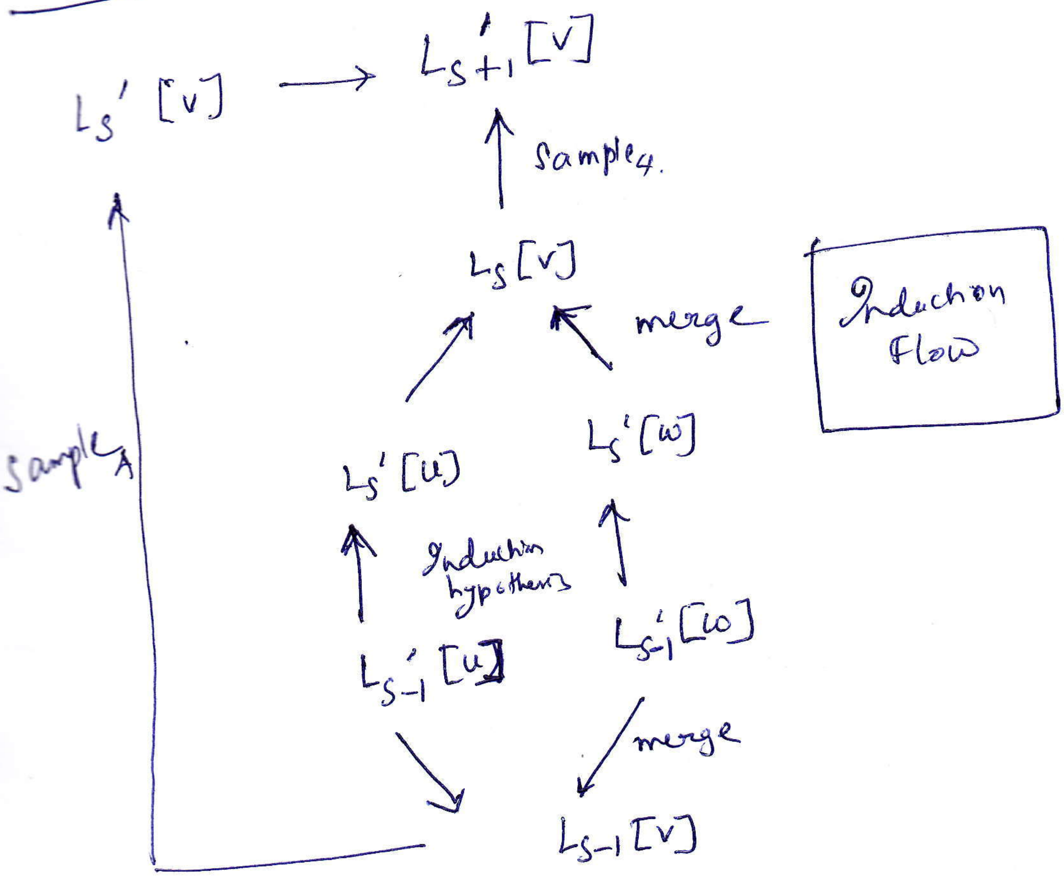
$x = 3 = (p + q - 2)$

By induction hypothesis, $[a, b_1]$ intersects $L_s'[u]$ in at most $(2p)$ elements & $L_s'[w]$ in at most $2q$ elements.

Now, $L_s[v]$ is just the list obtained after merging $L_s'[u]$ and $L_s'[w]$ $\Rightarrow$ $[a, b]$ intersects $L_s[v]$ in at most $2p + 2q \leq 8K - 2$ elements.

Since, $L_{s+1}'[v] = sample_4(L_s[v])$, we have $[a, b]$ intersects $L_{s+1}'[v]$ in at most $2K$ items.

|| note $sample(L_s[v]) = sample_4(L_s[v])$ as $s < 3alt(v)$.



<u>Implication</u> Let, a, b be adjacent items in $L_s'[v]$.

$\therefore$ $[a, b]$ intersects $(-\infty, L_s'[v], +\infty)$ in exactly two items.

$\therefore$ $[a, b]$ " $L_{s+1}'[v]$ in at most 4 elements.

$\therefore$ **$L_s'[v]$ is a 4-cover of $L_{s+1}'[v]$.**

Corollary

For each internal node v of T, and for each stage $s \geq alt(v)$, $L_s(v)$ is a 4-cover of $L_{s+1}'[u]$ and $L_{s+1}'[w]$.

Let, a & b be two adjacent elements of $(-\infty, L_s[v], +\infty)$.

Recall, $L_s[v]$ is obtained by merging $L_s'[u]$ and $L_s'[w]$.

Let, $[a', b']$ be smallest interval containing a and b st. $a', b' \in (-\infty, L_s'[u], +\infty)$. Clearly, a' & b' are adjacent in $(-\infty, L_s'[u], +\infty)$.

2 3 4 (6 7) 13 15

$L_s[v]$ 2 3 [4 6] 7 [13 15]

3 4 13 15

$L_s'[u]$ $L_s'[w]$ 2 (6 7)

3 4 13 15

$-\infty$ 2 6 7 $+\infty$

$\therefore$ By previous lemma, there are at most 4 elements in $L_{s+1}'[u]$ between a' & b'.

$\Rightarrow$ There are at most 4 elements in $L_{s+1}'[u]$ between a & b $\Rightarrow$ **$L_s[v]$ is a 4-cover of $L_{s+1}'[u]$.**

Likewise, **$L_s[v]$ is a 4-cover of $L_{s+1}'[w]$**

Lemma

Let, $S \geq 2$ be a given stage number.

Suppose, for every internal node v of T and its two children u and w, we are given:

1. $rank(L_s'[v] : L_{s+1}'[v])$
2. $rank(L_s'[u] : L_s'[w])$
3. $rank(L_s'[w] ; L_s'[u])$

Then using $O(|L_{s+1}'[u]| + |L_{s+1}'[w]|)$ operations, in $O(1)$ time we can calculate:

1. $rank(L_{s+1}'[v] : L_{s+2}'[v])$
2. $rank(L_{s+1}'[u] : L_{s+1}'[w])$
3. $rank(L_{s+1}'[w] : L_{s+1}'[u])$

(Proof: Pls refer text).

Corollary We can determine $rank(L_s[v] : L_{s+1}'[u])$ and $rank(L_s[v] : L_{s+1}'[w])$ in $O(1)$ time using a linear no. of operations.

The algorithm to merge and create $L_{s+1}[v]$ is thus:

Input: 1) $rank(L_s'[u] : L_{s+1}'[u])$ 2) $rank(L_s'[w] : L_{s+1}'[w])$ 3) $rank(L_s'[u] : L_s'[w])$ 4) $rank(L_s'[w] : L_s'[u])$.

1. Compute $rank(L_s[v], L_{s+1}'[u])$, $rank(L_s[v] : L_{s+1}'[w])$ (Corollary)
2. Merge $L_{s+1}'[u]$ & $L_{s+1}'[w]$ using $L_s[v]$ as a 4-cover for both lists. The resulting list is $L_{s+1}[v]$.
3. Update necessary information for stage $(S+2)$. [Lemma].