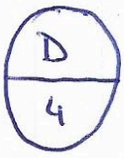
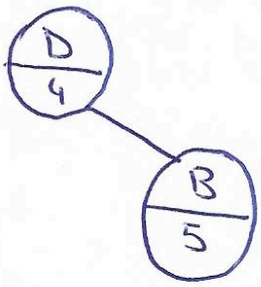
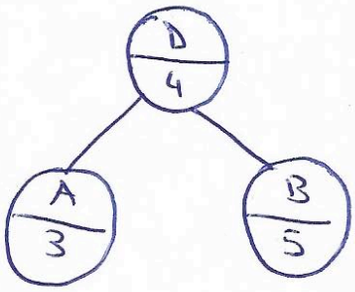
Algorithm Engineering

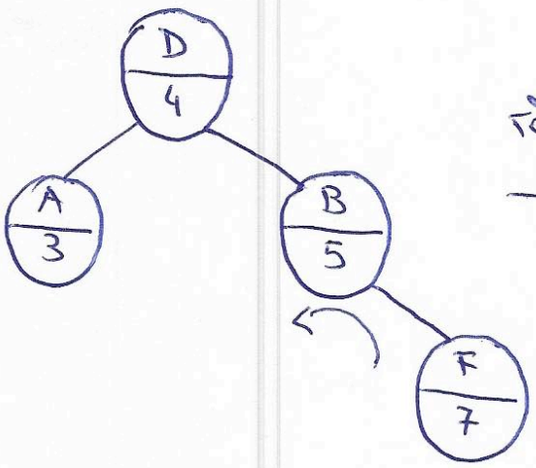
14/1/22

Q1:

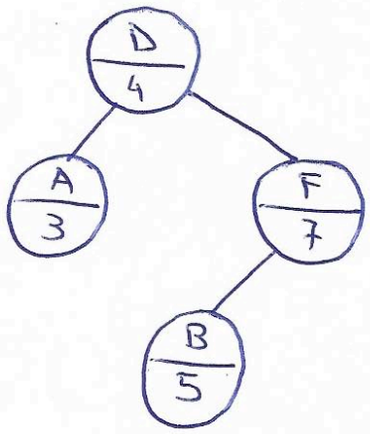




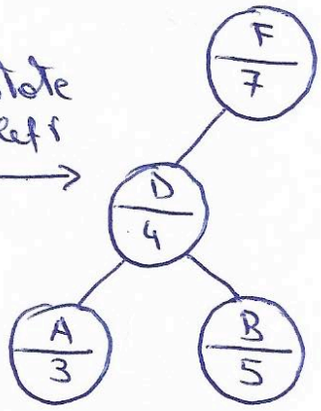


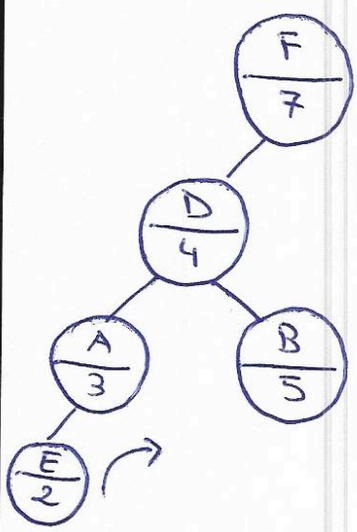


rotate left →

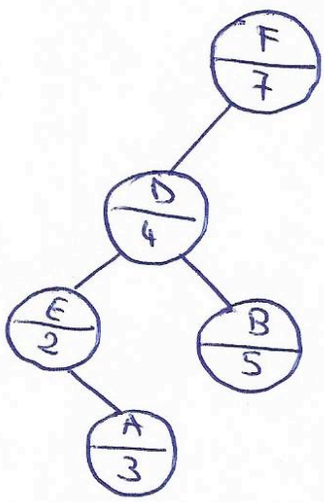


rotate left →

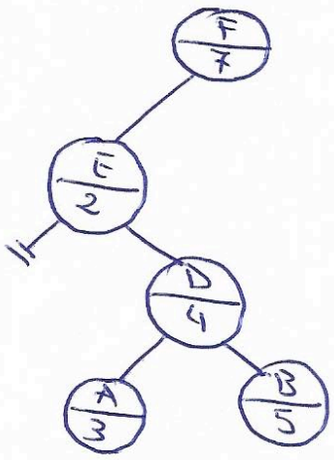




rotate right →



rotate right →



To solve the 3-sided range query $[-\infty, 6] \times [c, +\infty]$ we visit the Treap and stop when a node has a priority $< c$, and recurse left/right until a subtree can include keys in the range $[3, 6]$. Hence we visit: $\langle 7, F\rangle$, $\langle 2, E\rangle$, $\langle 4, D\rangle$, $\langle 5, B\rangle$

- $\langle 7, F\rangle$: no right, yes left
- $\langle 2, E\rangle$: no left, yes right
- $\langle 4, D\rangle$: no left
- $\langle 5, B\rangle$: stop

output $\langle 4, D\rangle$

Q2 : $A = (2, 3, 5, 6, 8, 10, 11)$ (indices 1, 2, 3, 4, 5, 6, 7)

$l = 1$, $r = 7$, $low = 2$, $high = 11$ } $m = \frac{l+r}{2} = 4$, $A[4] = 6$

BinaryCode(6, $low + m - l = 5$, $high + m - r = 8$)

$=$ BinaryCode(6, 5, 8)

$=$ encode $6-5=1$ in $\lceil \lg_2 8-5+1 \rceil = 2$ bits

$\Longrightarrow (01)_2$

Left recursive call

$l = 1$, $r = 3$, $low = 2$, $high = 5$ } $m = \frac{l+r}{2} = 2$, $A[2] = 3$

BinaryCode(3, $low + m - l = 3$, $high + m - r = 4$)

$=$ BinaryCode(3, 3, 4)

$=$ encode $3-3=0$ in $\lceil \lg_2 (4-3+1) \rceil = 1$ bits

$= (0)_2$

Right recursive call

$l = 5$, $r = 7$, $low = 7$, $high = 11$ } $m = \frac{l+r}{2} = 6$, $A[6] = 10$

BinaryCode(10, $low + m - l = 8$, $high + m - r = 10$)

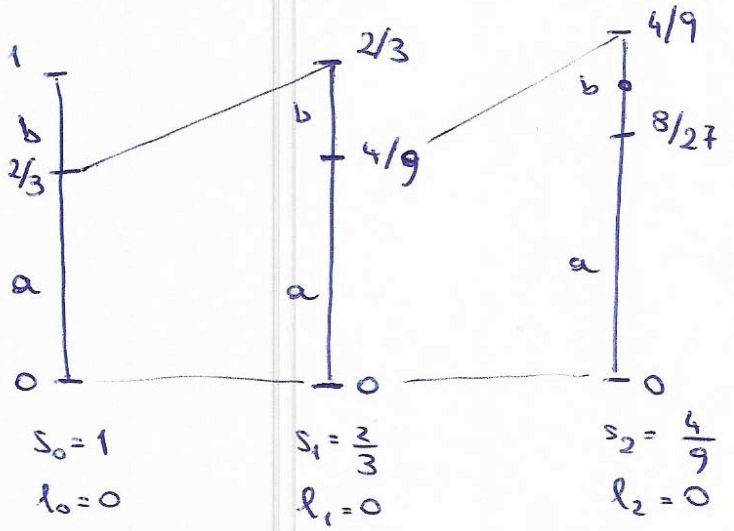
$=$ BinaryCode(10, 8, 10)

$=$ encode $10-8=2$ in $\lceil \lg_2 10-8+1 \rceil = 2$ bits

$= (10)_2$

Q3.

$T = aab \longrightarrow f(a) = \frac{2}{3} \quad f(b) = \frac{1}{3}$



$s_0 = 1 \quad l_0 = 0$

$s_1 = \frac{2}{3} \quad l_1 = 0$

$s_2 = \frac{4}{9} \quad l_2 = 0$

$s_3 = \frac{4}{27}, \quad l_3 + \frac{s_3}{2} = \frac{8}{27} + \frac{2}{27} = \frac{10}{27}$

$l_3 = \frac{8}{27}$

$$d = \left\lceil \lg_2 \frac{2}{s_3} \right\rceil = \left\lceil \lg_2 \frac{27}{2} \right\rceil = \left\lceil \lg_2 13.5 \right\rceil = 4 \text{ bits}$$

Execute the coding algorithm for emitting 4 bits for $\frac{10}{27}$.

$\frac{10}{27} \xrightarrow{\times 2} \frac{20}{27} < 1 \longrightarrow$ emit 0

$\frac{20}{27} \xrightarrow{\times 2} \frac{40}{27} > 1 \longrightarrow \frac{13}{27}$, emit 1

$\frac{13}{27} \xrightarrow{\times 2} \frac{26}{27} < 1 \longrightarrow$ emit 0

$\frac{26}{27} \xrightarrow{\times 2} \frac{52}{27} > 1 \longrightarrow \frac{25}{27}$, emit 1

$< .0101, 3 >$ output of Arithmetic code

Q4.

Key	2x mod 7	3x mod 7
1	2	3
2	4	6
3	6	2
4	1	5

SBF

0	1	2	1	1	1	2
0	1	2	3	4	5	6

Query (5) = min { SBF[3] = 1, SBF[1] = 1 } = 1

but this is wrong since 5 is not in the indexed set.

Q5.

The Elias-Fano code offers two functions to operate on its elements:

Access (i) = returns the value of the i-th element in the list

GEQ (x) = returns the position of the smallest element, which is $\geq x$.

```
Intersect (L1, n1, L2, n2)
    i=0, j=0;
    while (i < n1) and (j < n2) do
        x = Access (L1, i);
        y = Access (L2, j);
        if (x == y) then {print x; i++; j++;}
        elsif (x < y) then  i = GEQ (L1, y);
                      else  j = GEQ (L2, x);
```

As for as the complexity is concerned, since items are assumed from a universe of size u, and Access costs $O(1)$, and GEQ costs $O(\lg \frac{u}{n})$ Time, where n is the size of the list over which we execute the operation, the cost is

$O(n_1 + n_2 + n_1 \lg \frac{u}{n_2} + n_2 \lg \frac{u}{n_1})$ Time.