3) Write pseudocode that takes as input a binary tree T with nodes labeled with letters (possibly repeated) and represented via LOUDS plus an array L of those letters, and taken in input also a letter X and a positive integer d, prints true if ∃ a node in T labeled X and lies at distance d from the root (distance = # of edges). You have rank, select

Procedure α (T, L, X, d)

m = # of nodes of T

for i = 1 to m do

if (L[i] == X) then

y = i ; distance = 0;

while (y > 1)

$y = \left\lfloor \frac{select_1(y)}{2} \right\rfloor$

distance ++;

if (distance == d)

return true

RETURN FALSE

• Pseudocode that given a binary ARRAY that succinctly encodes a binary tree (with rank/select) CALCOLA la lunghezza del path che contiene solo nodi a sinistra (o equivalentemente, ~~if~~ la profondità del puntatore NULL più a sx)

```
length (B)
   left = 0;
    x = 0
      while (B[x] == 1) do
            left ++;
            x = rank_1 (2 + x)
            i = select_1 (x)
         end while
      return LEFT
```

• Scrivi pseudocodice che implementa intersezione tra 2 liste ordinate L_1, L_2 di n_1 e n_2 elementi, decodificate via Elias-phano usando access & next-geq

```
intersezione(L1, n1, L2, n2)
  i = 0
  j = 0
    while (i < n1 && j < n2) do
      x = access(L1, i)
      y = access(L2, j)

       if (x == y)
                   print x;
                   x++ ;
                   j++ ;
       else if (x < y)
                   i = geq(L1, y)
             else
                   j = geq(L2, x)
```

access $O(1)$ eseguiti n_1 & n_2 volte

geq $O(\log \frac{U}{n})$ $\Rightarrow O\left(n_1 + n_2 + n_1 \log \frac{U}{n_2} + n_2 \log \frac{U}{n_1}\right)$

• Dato un testo $T[1,n)$ costruisci un algo che sfrutta il suffix array su T per vedere:

Data una stringa $P[1,p)$ e 2 interi U e q se esiste un intervallo di U posizioni contigue in T (diciamo $T[i, i+k-1]$) in cui ci ~~sono~~ q occorrenze di P in T, in tal caso ritorna i altrimenti -1

1) Costruisco suffix array per T $SA[1,n]$
2) Binary search nell' SA: trovo gli indici L e R dell' intervallo delle occorrenze del pattern P
3) Poiché $SA[L,R]$ non è ordinato per posizioni, lo ordino

```
for (i=L to R)
    if (i+q-1 ≤ R) AND
            (SA[i+q-1] - SA[i] ≤ U)
        print i;
print -1
```

IL numero ottimale di funzioni di hash per il Bloom filter of size m and n keys

is $k = \frac{m}{n} \ln 2$

probabilità di avere una cella = 0

$$e^{-\frac{kn}{m}} = e^{-\frac{\frac{m}{\cancel{n}} \ln 2 \cdot \cancel{n}}{\cancel{m}}} = e^{-\ln 2} =$$

$$= \frac{1}{e^{\ln 2}} = \frac{1}{2}$$

UNIVERSAL HASH FUNCTIONS

$U = [0, 23)$ & keys $= 10$

=> numero medio di collisioni

$$\binom{n}{2} \cdot \frac{1}{m} = \frac{n(n-1)}{2m} = \frac{10 \cdot 9}{2 \cdot 23} = \frac{45}{23}$$

DISK STRIPING

$$\text{SORT} = O\left(\frac{n}{DB} \cdot \log_{\frac{M}{DB}} \frac{n}{M}\right)$$

su D dischi con disk striping

LOWER BOUND $\Omega\left(\frac{n}{DB} \log_{\frac{M}{B}} \frac{n}{M}\right)$

$$\frac{\frac{n}{DB} \cdot \log_{\frac{M}{DB}} \frac{n}{M}}{\frac{n}{DB}} \dots \frac{1}{1 - \log_{\frac{M}{B}} D}$$

se $M \to \infty \Rightarrow 1$ OPTIMAL

se $M \to DB \Rightarrow \infty$ NOT OPTIMAL