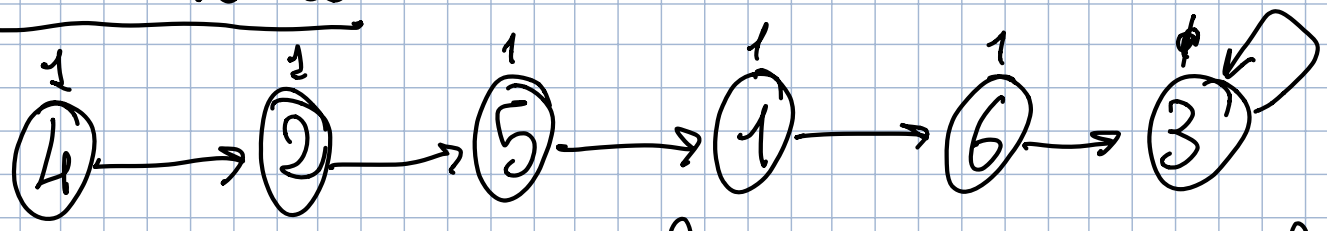
ESERCIZIO 2



Simulare l'operazione Pointer Jumping nel 2 level Model con Sort & Scan:

Si devono creare delle triple, una per ogni nodo: Se valido aggiornare il Rank:

A	B	rank
4	2	ϕ
2	5	ϕ
5	1	ϕ
1	6	ϕ
6	3	ϕ
3	ϕ	ϕ

Ordiniamo su B e calcoliamo il Rank(B[i]):

A	B	rank
3	0	-
5	1	1
4	2	1
6	3	ϕ
2	5	1
1	6	1

Ordiniamo su A e Sommiamo il rank con il rank di A

A	B	rank
1	6	2
2	5	2

⟨3, ϕ, ϕ⟩
⟨4, 2, 2⟩
⟨5, 1, 2⟩
⟨6, 3, 1⟩

Se voglio fare lo stesso cosa per trovare il successore

A	B	succ(B)
⟨4,	2,	ϕ⟩
⟨2,	5,	ϕ⟩
⟨5,	1,	ϕ⟩
⟨1,	6,	ϕ⟩
⟨6,	3,	ϕ⟩
⟨3,	ϕ,	ϕ⟩

Ordiniamo su B e calcoliamo il successore (B)

A	B	succ(B)
⟨3,	0,	ϕ⟩
⟨5,	1,	6⟩
⟨4,	2,	5⟩
⟨6,	3,	ϕ⟩
⟨2,	5,	1⟩
⟨1,	6,	3⟩

Poi ordino su A e assegno a B il succ(B)

ESERCIZIO 1

Simulare il Reservoir Sampling:

n=9 [a, b, c, d, e, f, g, h, i]

Random Integer:

[2, 4, 1, 2, 3, 1, ~~3~~]

R=[a, b, c]

h=2 → R=[a, d, c]

h=4 → NO

h=1 → R=[f, d, c]

h=2 → R=[f, g, c]

h=3 → R=[f, g, h]

h=1 → R=[i, g, h]

ESERCIZIO 3

① $S_1 = [1, 3, 10]$

$S_2 = [1, 3, 5, 7, 10, 15, 20]$

Intersect with mutual partitioning:
Prendiamo il pivot in S_1 perché è l'array più breve. Il pivot è l'elemento nel mezzo, quindi 3 e lo cerchiamo in S_2.
3 viene trovato in S_2 quindi va in output ora abbiamo:

S_1 [1] [10]

S_2 [1] [5, 7, 10, 15, 20]

Dobbiamo cercare 1 in 1 e 10 in $\{5, 7, 10, 15, 20\}$.
Viene restituito 10, poi con ricerca binaria cerco 10 nell'array e viene restituito anche 10.

Risultato set intersection $\{1, 3, 10\}$

② Dimostrare che il doubling mi costa

in tempo $O\left(m\left(1+\log_2 \frac{n}{m}\right)\right)$

Abbiamo che la ricerca viene fatta tra $[i+2^{k-1}, i+2^k]$ quindi i' che troviamo sta:

$$i+2^{k-1} < i' < i+2^k$$

$$2^{k-1} < i'-i < 2^k \rightarrow 2^k < 2(i'-i)$$

Chiamiamo $i' = i_j$ e il precedente elemento che abbiamo trovato è i_{j-1}.

Il Δ_j è il sotto array in cui facciamo la ricerca:

$$i_j - i_{j-1} < \Delta_j < 2^k < 2(i_j - i_{j-1})$$

Abbiamo che

$$\sum_{j=1}^{m} \Delta_j = \sum_{j=1}^{m} 2(i_j - i_{j-1}) = 2n$$

Ora possiamo dire che l'algoritmo esegue $O(\log \Delta_j)$ step, ed dire che abbiamo

$$\sum_{j=1}^{m} \log(\Delta_j) \leq \sum_{j=1}^{m} \log(2(i_j - i_{j-1}))$$

$$\sum_{j=1} \lg(\Delta_j) \approx \sum_{j=1} \lg\left(2(i_j - i_{j-1})\right)$$

$$\sum_{j=1}^{m} 1 + \lg(i_j - i_{j-1}) \approx m + \sum_{j=1}^{m} \lg(i_j - i_{j-1})$$

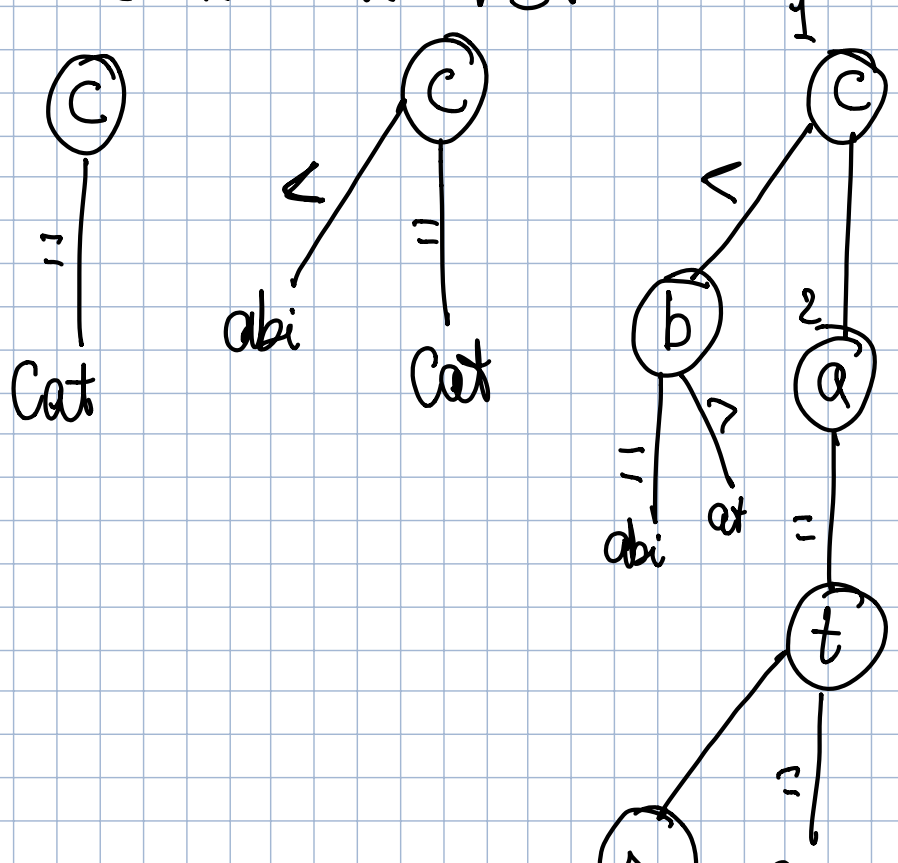
$$m + m \sum_{j=1}^{m} \lg \frac{(i_j - i_{j-1})}{m} \approx m + m \cdot \lg \frac{n}{m}$$

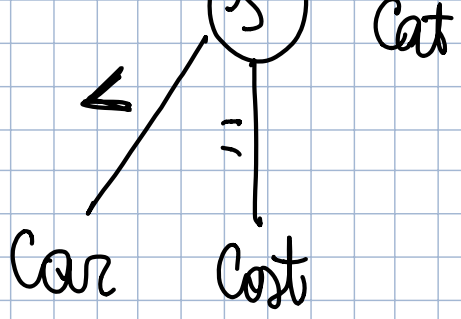
$$m\left(1 + \lg \frac{n}{m}\right)$$

ESERCIZIO 4

$S = [\text{cat}, \text{abi}, \text{cast}, \text{car}, \text{at}]$

Inserire in un TST:





② MultiwayQS(S, i, j, g)

p = pivot presa a caso

$S_< : \{ s : s[i \cdot (g-1) \,..\, i \cdot g] < p[i \cdot (g-1) \,..\, i \cdot g] \}$

$S_= : \{ s : s[i \cdot (g-1) \,..\, i \cdot g] = p[i \cdot (g-1) \,..\, i \cdot g] \}$

$S_> : \{ s : s[i \cdot (g-1) \,..\, i \cdot g] > p[i \cdot (g-1) \,..\, i \cdot g] \}$

MkQS($S_<$, i, j, g)

MkQS($S_=$, i+g, j, g)

MkQS($S_>$, i, j, g)

}

Standard $O(d + n \log n)$

In questo caso abbiamo che nel caso = Vengono confrontati blocchi di g lettere ogni volta

Complexity:

S { $S_<, S_>$ → non avanza g però si dimezza $S_=$ e $S_>$

S = costo S

Tempo: $S + g \log n$

ESERCIZIO 5