DOR → Mid-sem

Roll No: → 2021PCS1017

Name → Ambuj Mishra

Que① → To use skip-list, we need to skip element if we want to use it "efficiently".

forexample →

Betty → [1] → [5] → [8] → [11] → [12] → [16]

Cake → [11].

→ This is an example which shows the inverted indexes of betty & cake. So, for following cases →

ⓐ. Betty AND cake → we can use skip-list as we only need to search & keep intersection pts.

ⓑ. Betty OR cake → Using skip-list will not be efficient. As we have to traverse through both lists anyway.

Que③ →

word① → [2, 4, 5, 11, 14, 17, 19, 23, 28, 31, 34, 67, 89, 91, 145, 167, 180];

word② → [67];

size (word1) = 17

size (word2) = 1

To find the intersection → with skip length = 3, comparisons are made as following →

Comparison No.↓	Comparison Detail↓	Comparison Result
1	→ 2 and 67	→ skip 3 values bcoz lesser no.
2	→ 11 and 67	→ " "

3	→ 19 and 67	→ "
4	→ 31 and 67	→ "
5	→ 89 and 67	→ Since value became greater. So, it's not skipped here but checked in this interval throughout
6	→ 34 and 67	→ Not matched
7	→ 67 and 67	→ Matched .

No. of comp.^{n} ⇒ 7.

Note → assuming that skip list is not made using Doubly LL. but only singly LL. and moving in forward dir.^{n} only. Otherwise, we can move back & reduce comp.^{n} as well

Que 2 → (Tang. or Trees) AND (Orange OR Manges) AND (fruits OR Veg.)

Note → idea: we'll do the processing in order of increasing frequencies starting from smaller set. Mainly becoz of the reason that T.C. of intersection or AND op.^{n} is ⇒ O(smaller set size)

estimated size of →

(i) → (Tang. or Trees) = 46k + 316k = (362k)

(ii) → (Orange OR Mangoes) = 107k + 271k = (378k)

(iii) → (fruits OR Veg.) = 213k + 87k = (300k) we need to calculate → (i) AND (ii) AND (iii).

Query processing order → {(iii). AND (i)} AND (ii).

→ We'll take intersection of smallest set (iii). with (i). & then intersect it with (ii).

Modifications → (i). We can use term-frequency as well. which will give directly idea about it's posting size

(ii). we ~~have assumed~~ have to assume that [X OR Y = size(X) + size(Y)]; although it's not always true. As 2 sets having 9 values & 990 values will have higher probability of resulting in larger union sets than 2 sets with 500 values, Although size(X) + size(Y) is smaller. (Depends upon similarity).

Que(5) → step(1) → Tokenization → And [Normalization] →

Doc(1) → ["Betty", "bought", "a", "butterscotchcake"]

Doc(2) → ["The", "cake", "was", "very", "bitter"]

Doc(3) → ["Betty", "returned", "the", "cake bcos", "the", cake, "was", "bitter"]

Doc(4) → ["Betty", "got", "a", "new", "cake", "andcontd", "the", "party"]

{ Note → using Normalization we Removed symbols. (', -, ') & replaced with NULL } step ② → stop-words →

["a", "The", "was", "very", "and", "new"] will be removed from all the documents. if they are present.

step ③ → Stemming →

{ "Cakecos" → "cake"
"Bitter" → "Bitt"
"Returned" → "Return"
"andcontd" → "and cont"
"Got" → "Go" } → These changes are made throughout the documents

After that posting list →

"Betty" → ① → ② → ③
"Bought" → ①
"Butterscotch cake" → ①
"cake" → ② → ③ → ④
"Bitt" → ② → ③
"return" → ③
"Go" → ④
"and cont" → ④
"party" → ④

Dictionary | posting.

→ Now sort, on the basis of Dictionary key
⇓
That is final result of postings.
(sorry, No time).