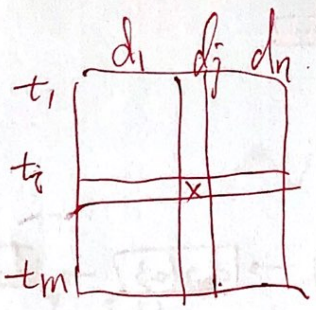
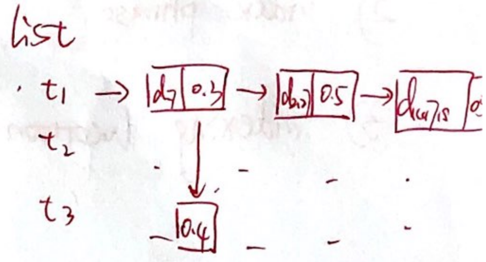


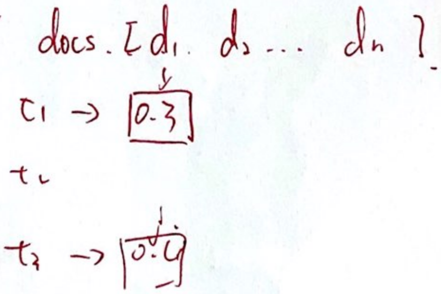
efficiency ?

- space
- query pro

$q = t_1, t_2, t_3 \ldots$



or.



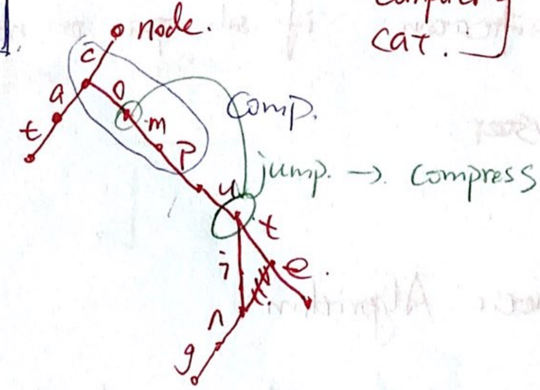
$\sum_{t \in q \cap d}$

start from term of query.

find (t, List)

① sorted list - $O(\log_2 N)$.

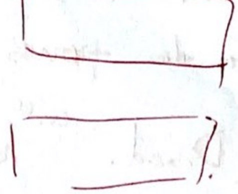
Index ? - { hashing. B⁺tree. trie : }



Compute
computing
cat.

→ parallezation : $\vec{d}$ $\vec{q}$

$(t_1, t_2, t_3 \ldots\ldots)$



→ " t1 t2 "

" t1 NEAR t2 "

within 10 terms ....

1) run time.

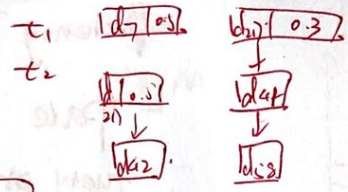
- look up t1
- look up t2
- intersection
- check position ☆

ad: not need to change index.

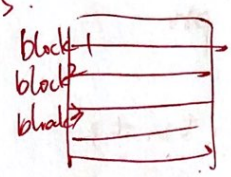
disad: very common or very less docs will cost a lot to find out.

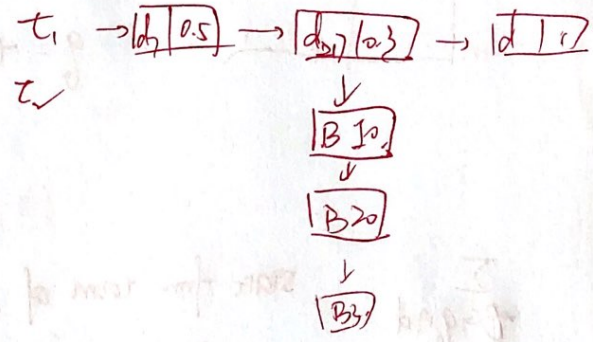
3rd-Nov IR-1

2). index phrase.



3) indexing locations.





Or can also store the GAP between the index locations so that we can save more space

IR: (Learning process).

classification: if sth span or not / ≠ relevant or not.

cluster:

Genetic Algorithm:

Start with a random set of sols.

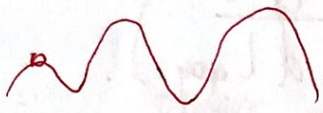
Calculate the fitness

- Disad.: will have convergence on a certain weight.

mutation: mutate operators.

tf x idf.

term weighting scheme.



T = {df, cf, N, V, C, 1, 10, 0.5}.

df: doc freque cf: cross-frequency values of df, cf

F: { }

factor operators.

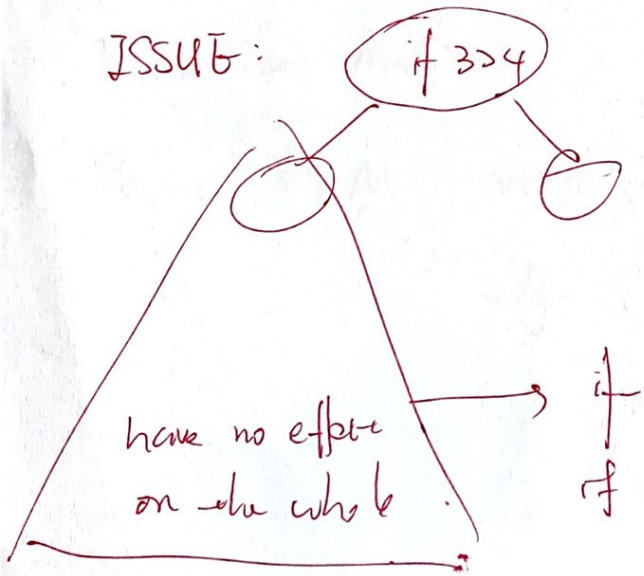
There is no bias. ⇒

N: num of doc. cf: collection frequency

df: doc frequency.

term density $w_3 :=$ ↓

ISSUE:



if there are too large: cut the subtree

if the fitness is the same:

keep the shortest subtree

BM 25

Pivot normalisation.

$$\sqrt{\left(\frac{dl}{dl_{avg}}\right)}.$$

↘

There is no turning involved.