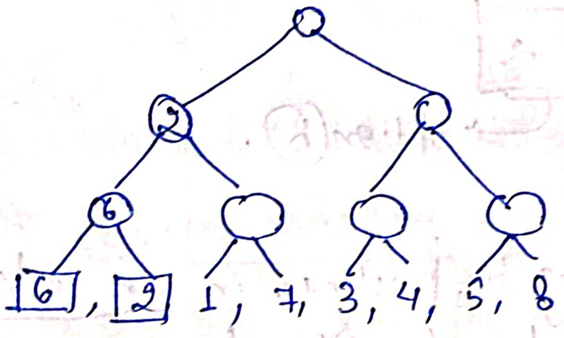
$\longrightarrow$ AMBUJ MISHRA
$\longrightarrow$ 2021PCS1017

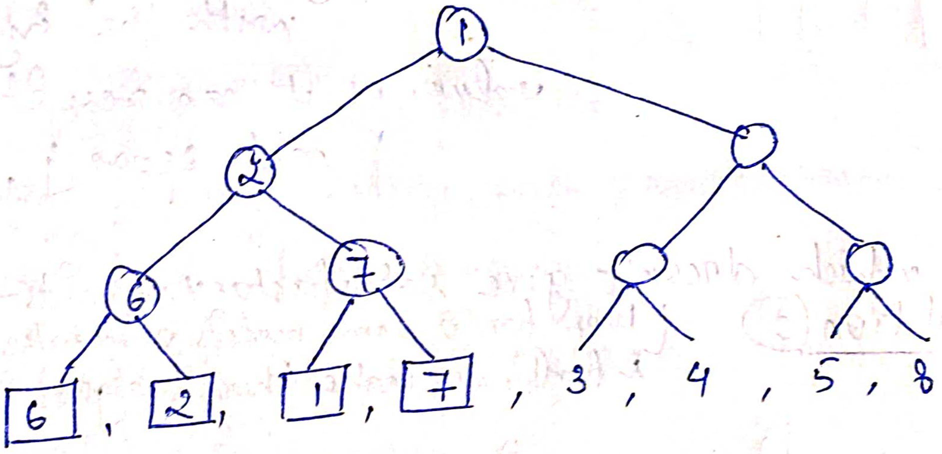
Que ② $\rightarrow$ Minimum loser tree $\rightarrow$

ⓐ $\rightarrow$ 6, 2, 1, 7, 3, 4, 5, 8

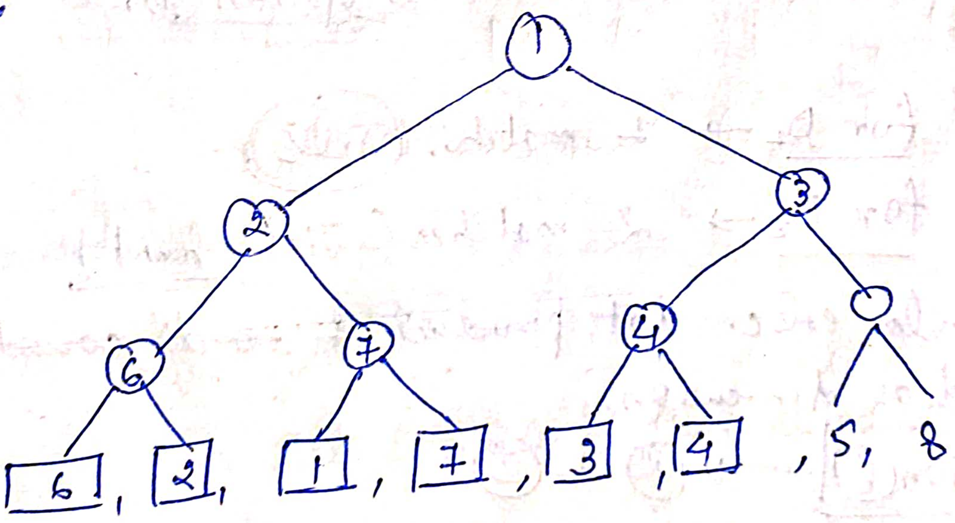
step ① $\rightarrow$



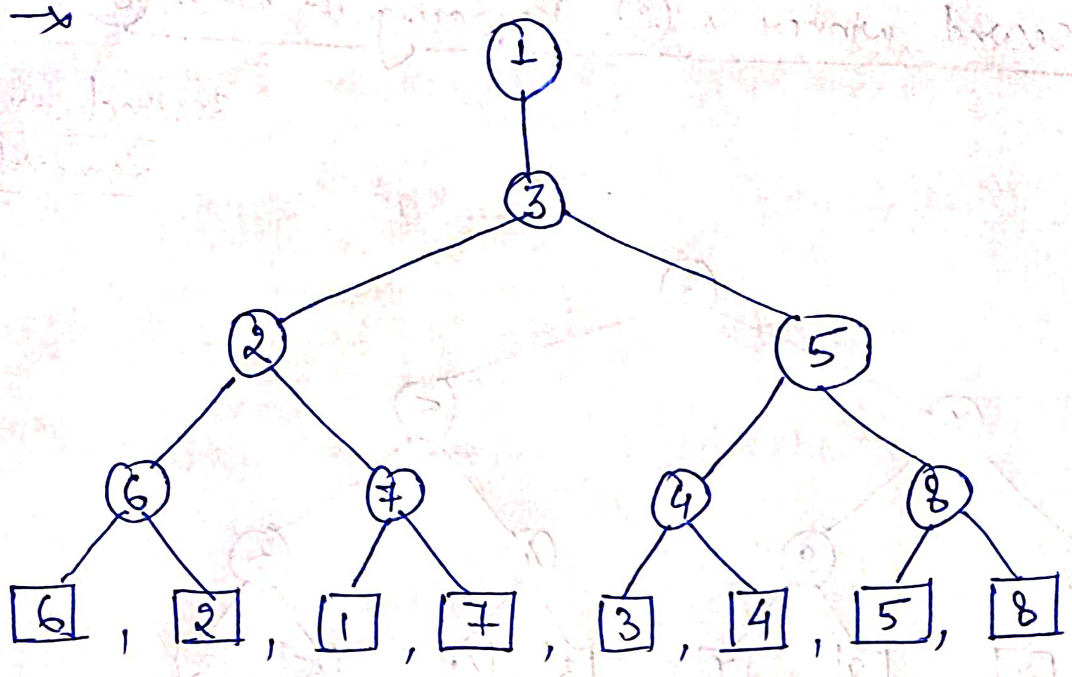
step ② $\rightarrow$



step ③ $\rightarrow$



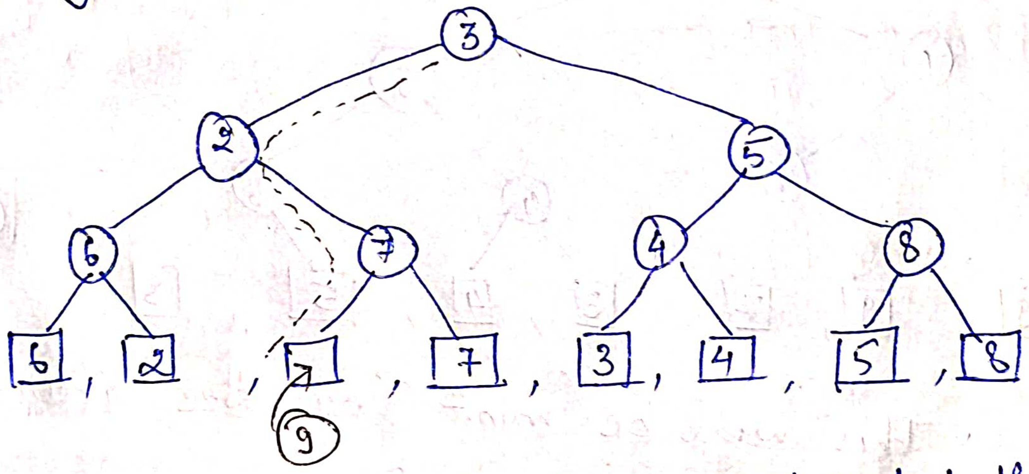
step ④ →



→ This is final min.ⁿ loser tree.

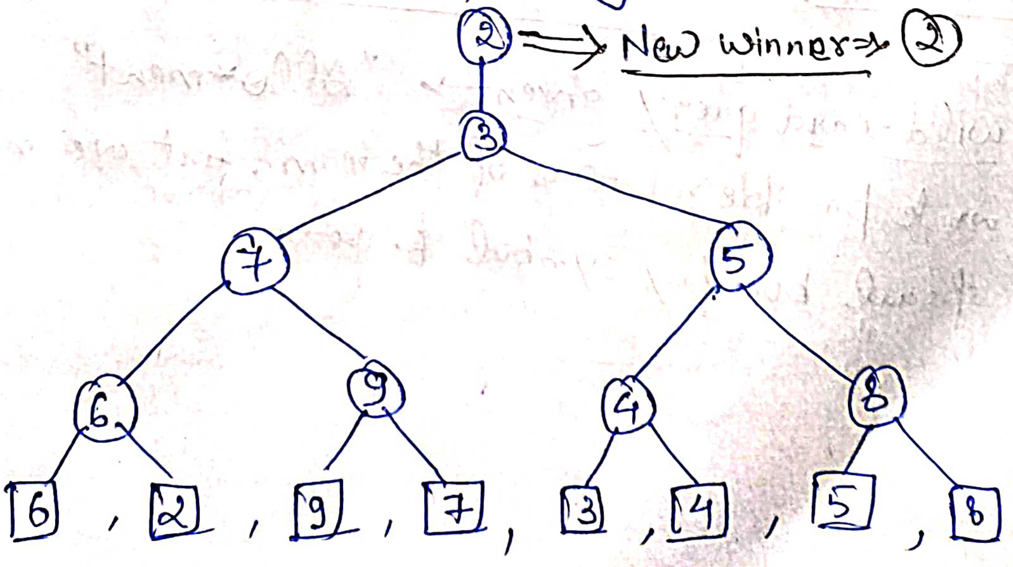
ⓑ → first winner ⇒ ①, so, replacing '1' with '9' & then regeneration →

first winner Removal.



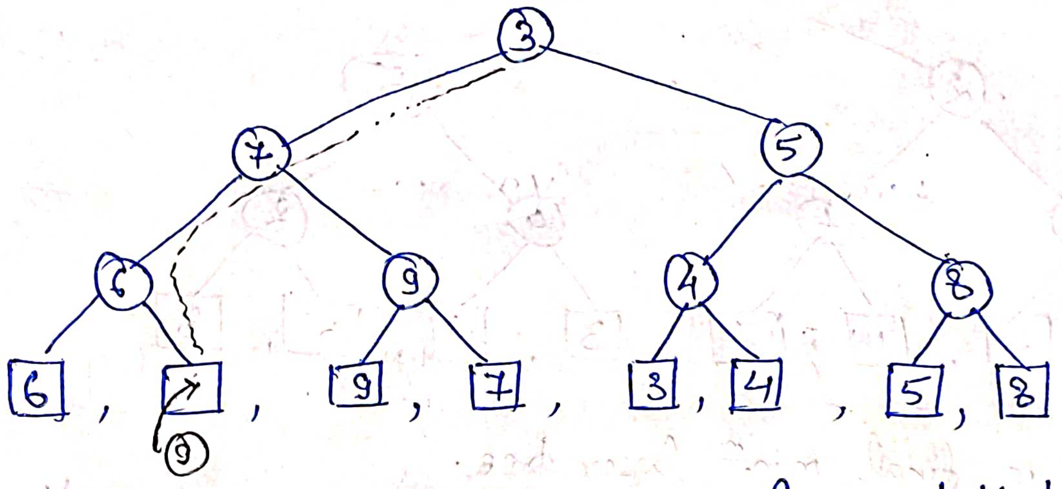
→ values may change in dotted path, so →

② ⟹ New winner ⇒ ②

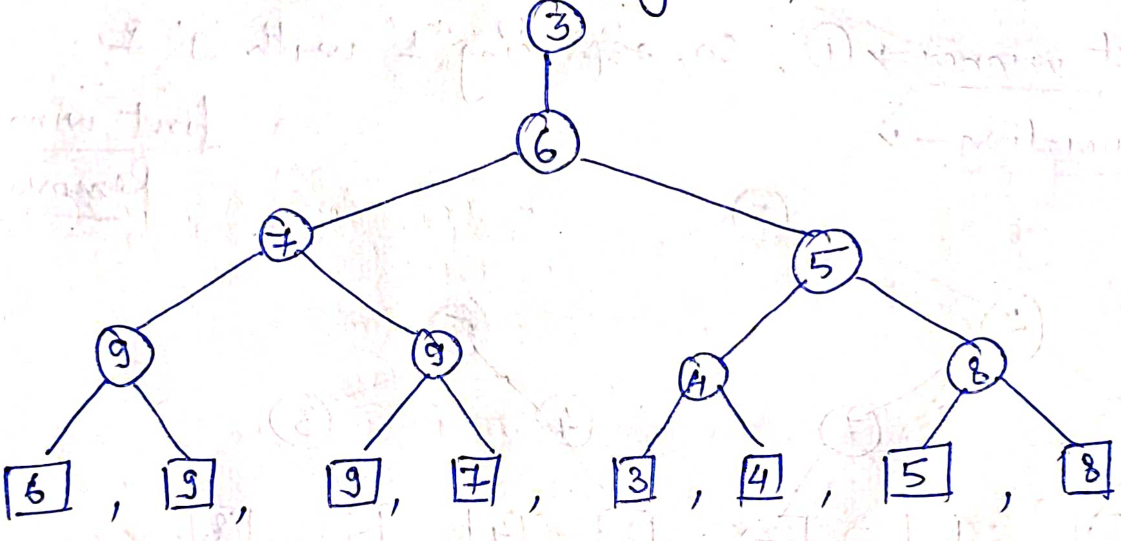


Since second winner is ②, Replacing it with ⑨ →

Second winner Removal



→ values will change only in dotted path.



→ This will be min.ᵐ loser tree after 2 winner Removals.

Que.⑦ → wild-card query given → "Allo*ment"

→ To write possible rotations of the term, first we need to use a special boundary symbol $. so →

⇒ Allo * ment$ ⇒ possible all rotations.

⇒ llo*ment$A , lo*ment$Al , o*ment$All ,

ment$Allo, $\boxed{\text{ment\$Allo}}$, ent$Allo*m, nt$Allo*me,

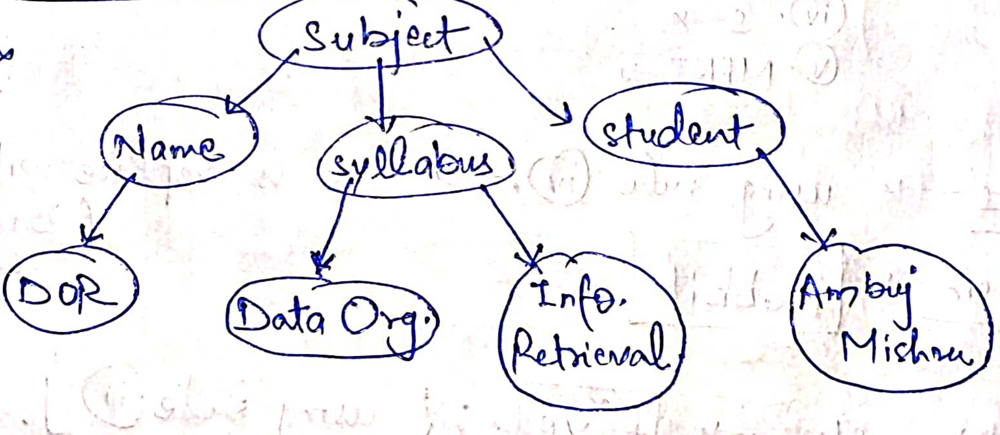
t$Allo*men, $Allo*ment, Allo*ment$

→ To find the fittest term, we'll take the rotation, in which * is at last. So → $\boxed{\text{ment\$Allo*}}$

(b) → Keys to look one → from this we can conclude that we need terms, which start with $\boxed{\text{\$Allo}}$ and that ends with $\boxed{\text{ment\$}}$. We can create n-gram to find that.

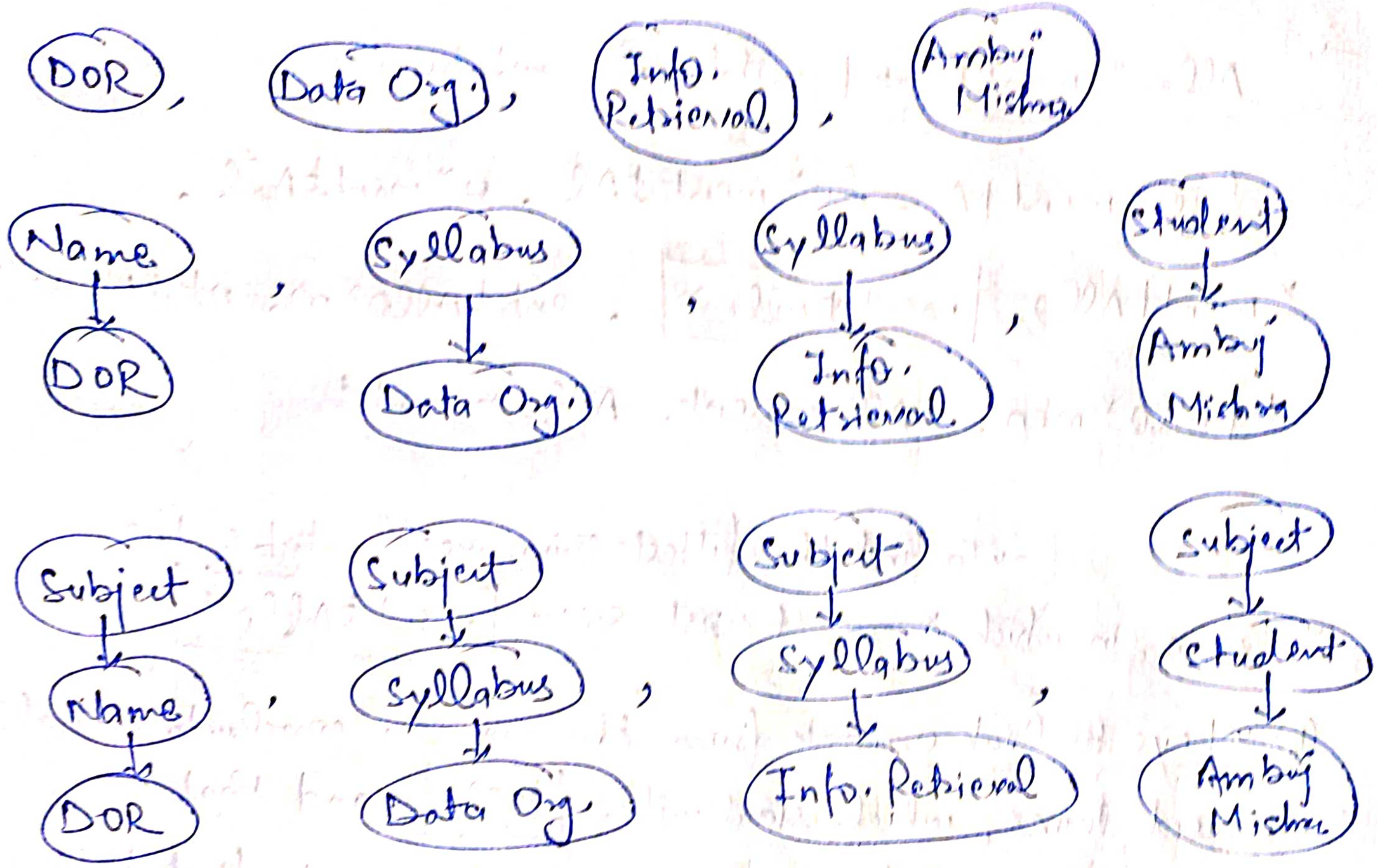
(c). Different Trigram → $Al, All, llo, men, ent, nt$

Que 4 →



So, structural terms →

check Next page.



Que. 6 → Rules → (i). ION → E
(ii). ING → E
(iii). SS → SS
(iv). S →
(v). MENT →

(a). →

(i). Rabbits → using rule (iv). → 's' is replace with blank. (basically Removed)

stem → $\boxed{\text{Rabbit}}$

(ii) Flying → stem → $\boxed{\text{flye}}$; { using Rule (ii) }.

(iii). Cement → stem → $\boxed{\text{ce}}$; { using Rule (v) }.

(iv). Ration → stem → $\boxed{\text{Rate}}$; { using Rule (i) } .

(b) → These stemming rules are case-specific. They'll generate meaningful words few types of words like in case of "Rabbits" but not meaningful words in case of "Cement".

(i) → Meaningful
(ii) → Not meaningful (or may be semi-meaningful depending upon how we use it further).
(iii) → Not meaningful
(iv) → Not meaningful (context is changed completely).

→ We can say, in general that these rule may ~~work~~ work if these suffixes are added extra on the words. Like if plurals are given. But wouldn't ~~work~~, if these are part of word meaning. 'ING' suffix case wouldn't work if it is not added by removing 'e'.

forexample → "Caution" will become → "Caute" {which is meaningless}.

(c). Modification → Using "lemmatization" would be an appropriate solⁿ: But, if we want to change these rules to improve accuracy, then → we need to try to convert all possible form in one common "stem" form. So →

ING → {"ing" can be completely removed}

MENT → MENT {leave it as it is}.

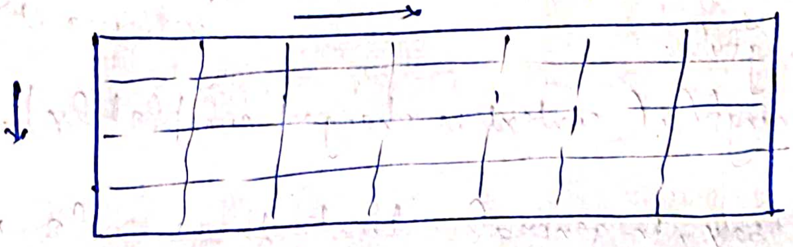
→ We can also put condⁿ to have a chek on which vowel can occur in case of "ING".

Que ① → No. of Documents → 1 M.

Vocabulary → 1000 terms.

So → Term Document incidence matrix →

Document id vectors corresponding to terms.



Issues → ①. any term will be present in only a limited no. of Documents. So, matrix will be a sparse matrix. It means that space will be wasted.

②. → entries in matrix ⇒ $10^3 \times 10^6 \rightarrow \boxed{10^9}$ ∴ so, size of will be very large of matrix. and memory can get full.

Alternative → Instead of term-document incidence matrix, inverted indexing method should be used.

Que ② → RDBMS can't be used for structured and unstructured IR because →

(i). Users generally don't know about the structure of the database system.

(ii). Using RDBMS would result is larger no. of returned results which will unnecessarily increase the processing time

(iii) Users may not know how to write queries in RDBMS query languages. To search, users should have a proper knowledge about query languages.

(iv). RBBMS system is an unranked system. So, in case of multiple results, it wouldn't be able to present most relatabale and highest ranked results.

(v). On unstructured data, applying RDBMS is not good as the ~~result~~ query can be wrong because of unstructured info while trying to access RDBMS.

Que (3) → $\boxed{\text{false}}$. ⇒ Because if we don't applying stemming on query then stem of query term and stem of it's correct index term (if it exists) may be different.

for example → while indexing, suppose we have converted word "flying" into "fly" & then saved it in index, then if "flying" is present in query, then before searching, we need to convert this query word also in it's stem form "fly".

⇒ Que (5) → In map-reduce algo., parser is used in map phase to distribute data & inverter are used in reduce phase to collect the data and bind it together.

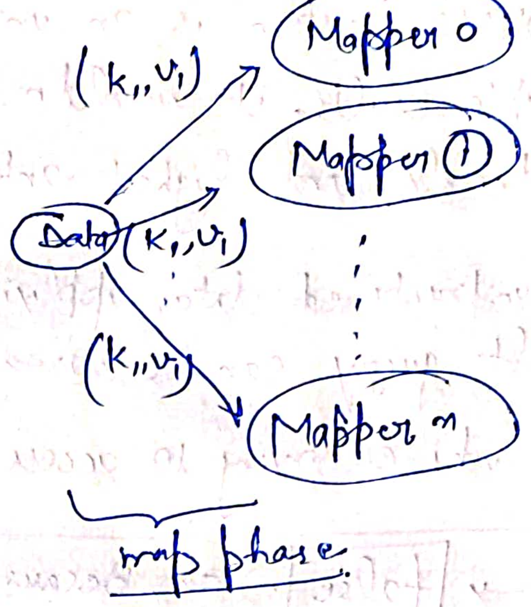
structure → {term → Doc. id, term freq.}

Pseudocode → first input all data.

(a) Map() fn → ⇒ Take terms & T.f as input.

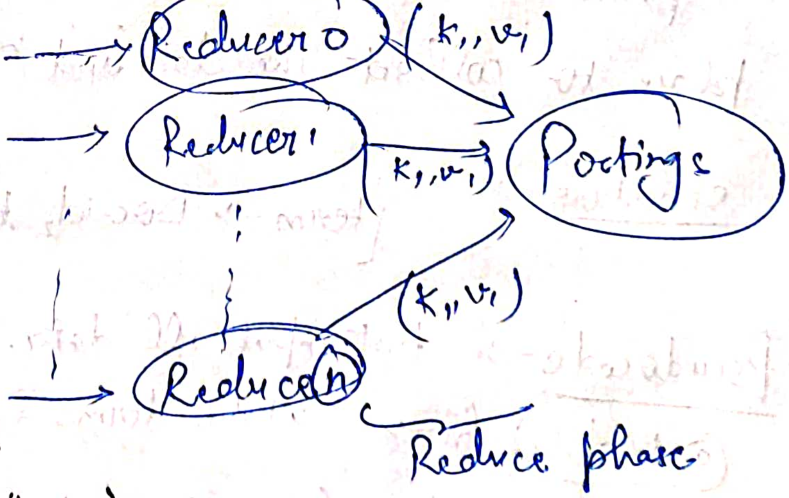
```
input( list (k,v))        ; { list of all terms}.
Partionar( list (k,v))    ; { Partitioners are used to distribute data}.
```

→ partitioner takes keys & distribute to Mappers.
{ every 'K'th term is provided / distributed to 'n' mappers }.



pseudocode for Reduce() → Combiner or Reducers are used to collect data.

```
→ Take input from all mappers.
→ for {k:all term}:
      Take kth term and it's partions.
      Combiner (k, list(v));
→ Output (k, list(v));
```



for example →

("Hello", 1)
("Hello", 1)
("Hello", 1)

⇓ input taken from map phase

→ combiner/Reducer → ("Hello", 3)