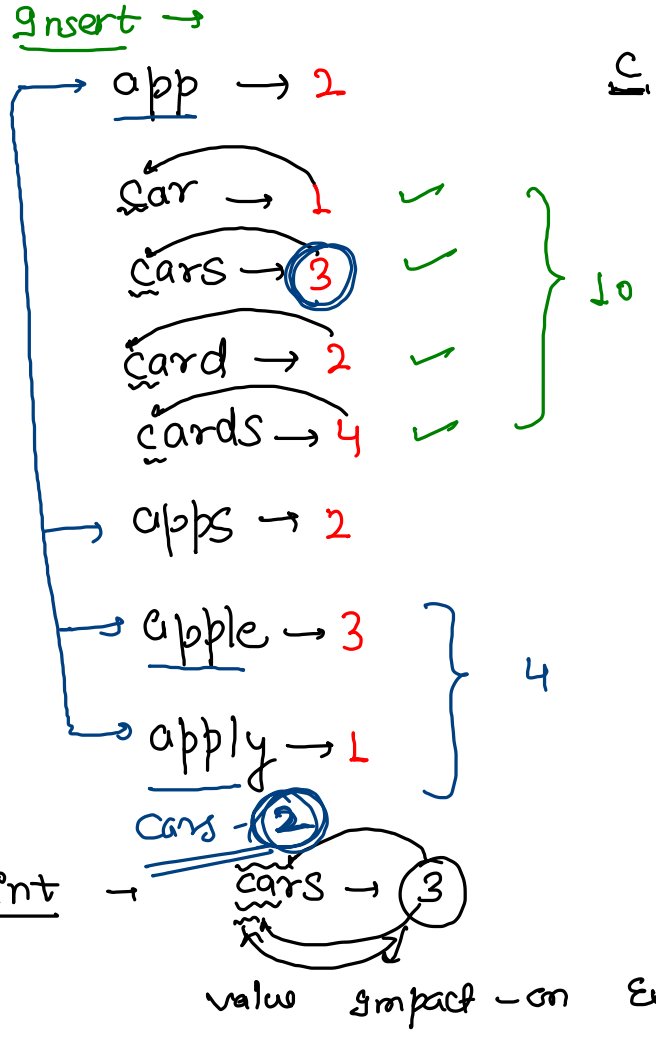
Map Sum Pair



prefix is unique →

car → 1
cars → 3

① Sum ("car") → 10

find keys with prefix "car" and return sum of values.

② sum ("appl") → 4

③ sum ("app") → 8

* Brute force ?

car - 1

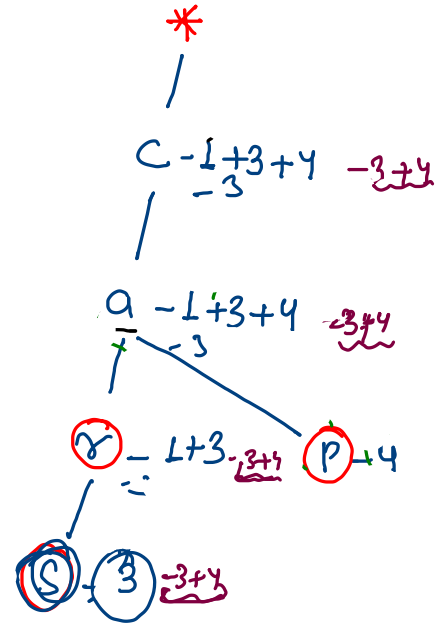
cars - 3

cap - 4

sum("ca")

sum("cap")

cars - 4



HashMap

String vs Integer.

cars → 3

oval = map.get("cars") (3)

gmpact = nval - oval

oval = map.getORDefaut("cars", 0);

gmpact : 4 - 3 = (1)

* Catch gmpact from Hashmap.
* Add value in Hashmap
* Insert → Add gmpact in Trie.
  sum → find val at last index

cot - 1

ca - 2

cat - 3

→ [sum("ca") → 5

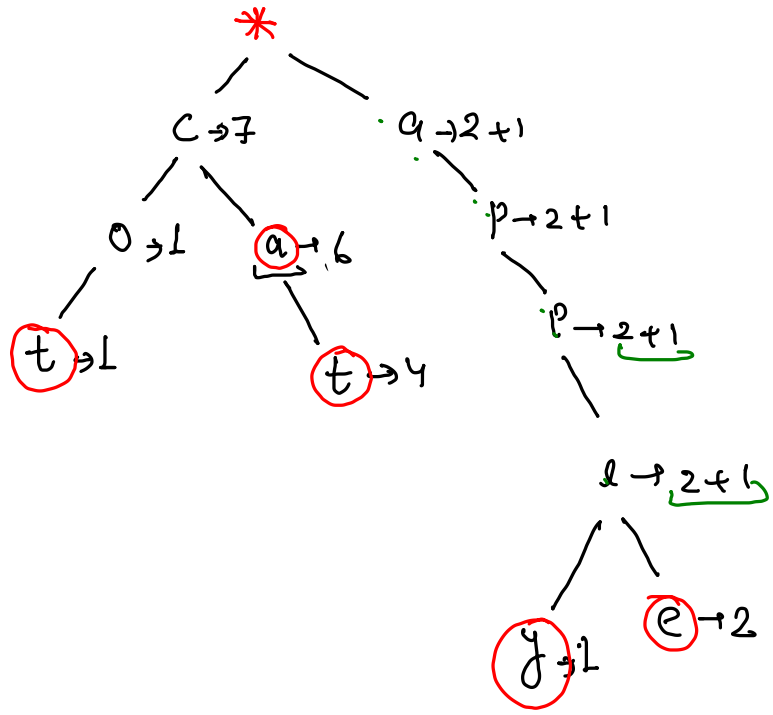
cat - 4

apple - 2

apply - 1

→ sum("appl") → 3

→ sum("app") → 3



oval = ~~0~~ ~~0~~ ~~0~~ ~~0~~ ~~0~~ 0

nval = ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~2~~ 1

impact = ~~1~~ ~~2~~ ~~3~~ ~~1~~ ~~2~~ 1

H.Map

String vs. Integer

cot → 1

ca → 2

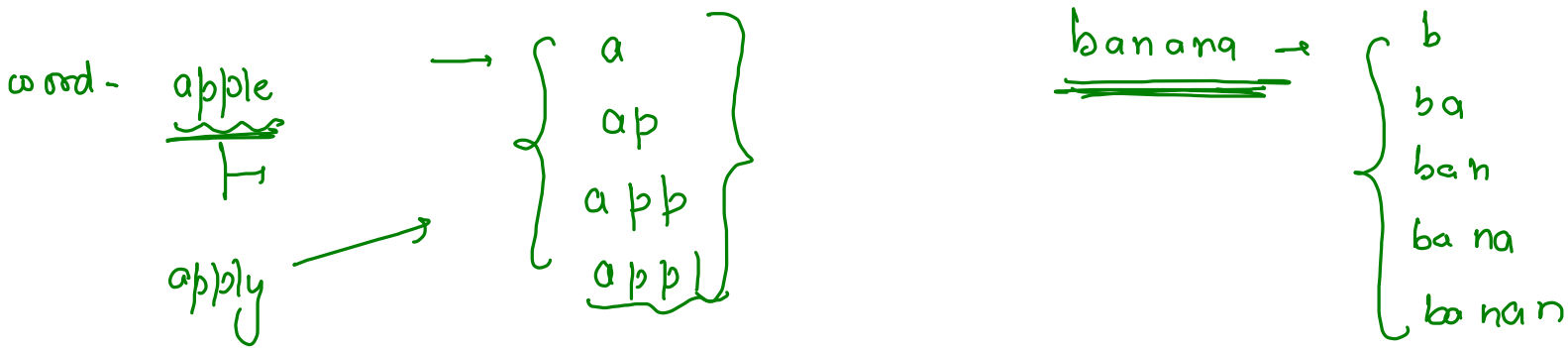
cat → ~~3~~ 4

apple → 2

apply → 1

Longest Word in Dictionary

words = ["a","banana","app","appl","ap","apply","apple"]

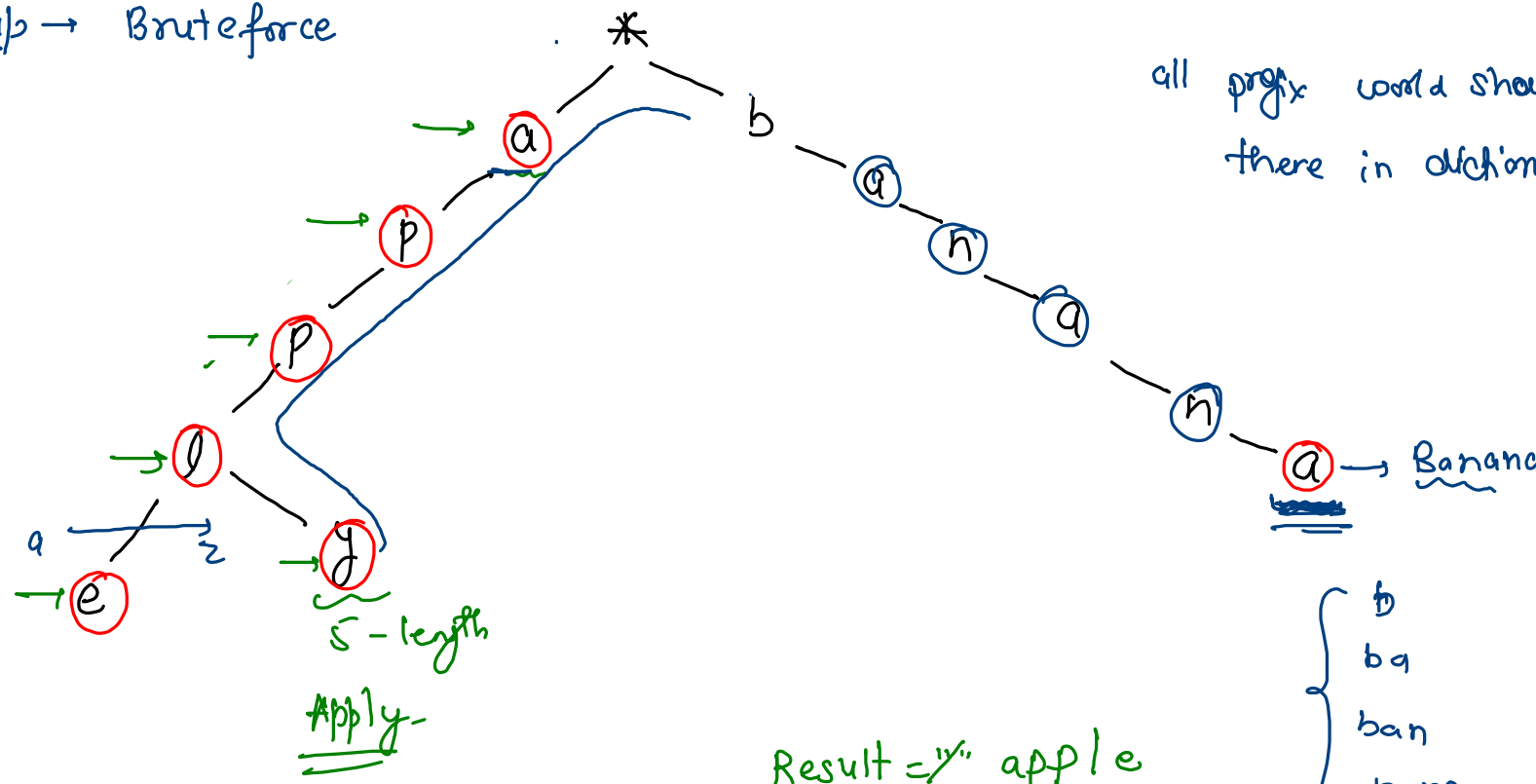


apple is smaller in terms of lexicographical order.

⟶ apple is answer.

HashMap → Bruteforce

words =
[
"a",
"banana",
"app",
"appl",
"ap",
"apply",
"apple"
]



all prefix word should be there in dictionary.

b
ba
ban
bana
banan

How to make sure that we are selecting smallest lexicographical word?

→ Child. Search from a to z

Result = "apple"

length = 0 1 2 3 4 5

smallest lexicographical word will select first

words =
[
"a",
"banana",
"app",
"appl",
"ap",
"apply",
"apple"
]

Bruteforce :→

① add words in map.

② Select word and check all possible prefix in map.

③ If all prefix are available then maximise length and manage Result.

④ otherwise move forward at next word.

word → ① app

- a ✓
- ap ✓
- app ✓

⑪ appl

- a ✓
- ap ✓
- app ✓
- appl ✓

String res = ~~"" app appl~~ apple.

int length = ~~0 3 4~~ 5

word →

a

ap

app

apple

c

cat

b

ba

bat

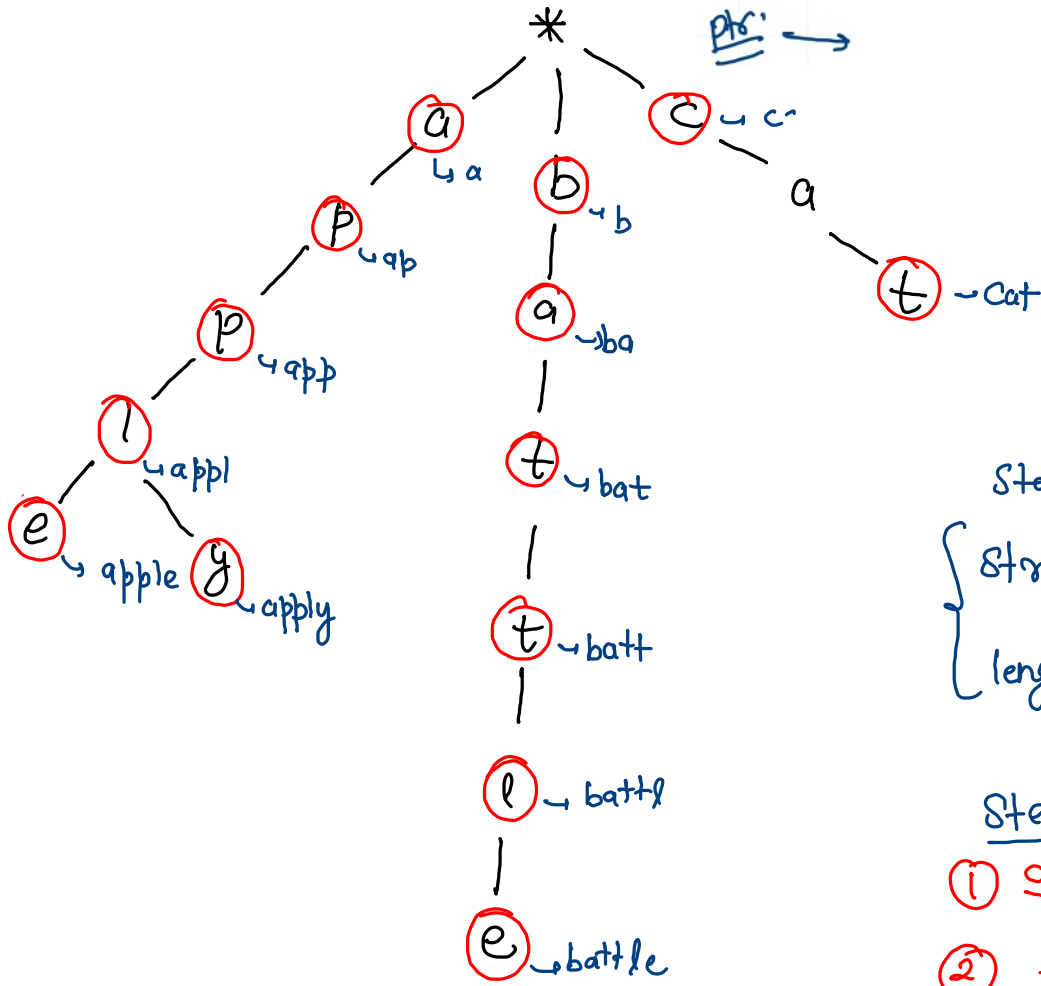
batt

battl

battle

appl

apply.



Static variable →

String req = battle

length = 6

Steps:

① Insert word in Trie

② find longest word.

Stream of Character

words →

"cd", "f", "kl"

cd
f de
kl

→ HashMap

Brute force →

HashMap

↳ check all possible combinations.

a ✓
b ✓
c ✓
d ✓
e ✓
f ✓
g ✓
h ✓
i ✓
j ✓
k ✓
l ✓

a	b	c	d	e	f	g	h	i	j	k	l
False	False	False	True	False True	True	F	F	F	F	F	True

a	b	c	d	e	f	...	l
	ab	bc	cd	de	ef		kl
		abc	cde	cde	def		jkl
				bcde	cdef		ijkl
				abcde	⋮		⋮
							abc---jkl

How to Implement Stream of character using TRIE ⟶

words → cat, Rat, Pot, mate, in, no, amt

cam.

c	a	m	t	c	a	t	m	a	t	n	o
F	F	F	T	F	F	T	F	F	T	F	T

Suffix

c	a	m	t	c
	ca	am	mt	tc
		cam	amt	mtc
			camt	amtc
				camtc

Add coord in Reverse order

Suffix

m
am
cam

⟶ Suffix → convert into → prefix

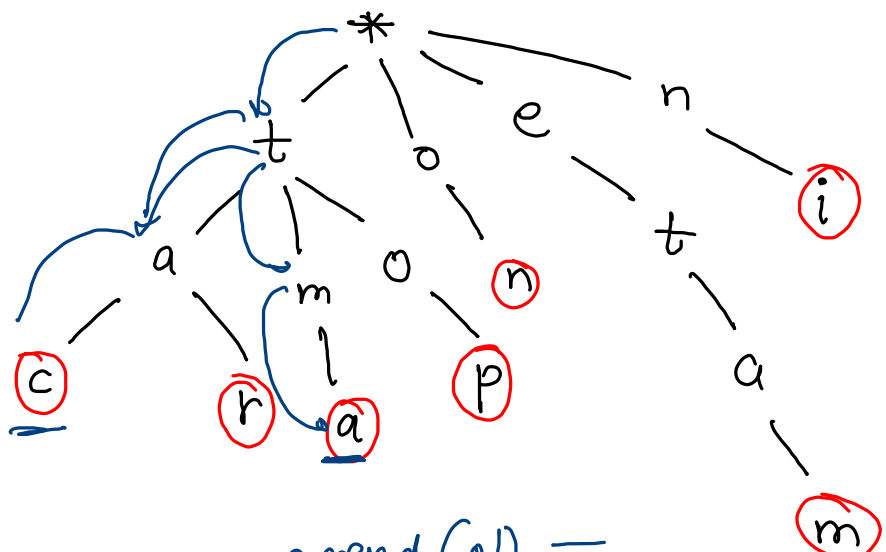
cat, Rat, Pot, mate, in, no, amt

logic ??

c a m t c a t m a t n o]

F f F T F F T F f F F T

~~cat~~
~~Rat~~
~~Pot~~
amt



① How to get previous character?

②

StrBuilder ——→ append(ol)
getChar(ol)

ArrayList<Character> ——→ addLast - O(1)
getIndex() O(1)

```
fun() {
    a = 15

}

main() {
    a = 10
    syso(a);
    fun()
    syso(a);
}
```

print 1 → 10

print 2 → 15