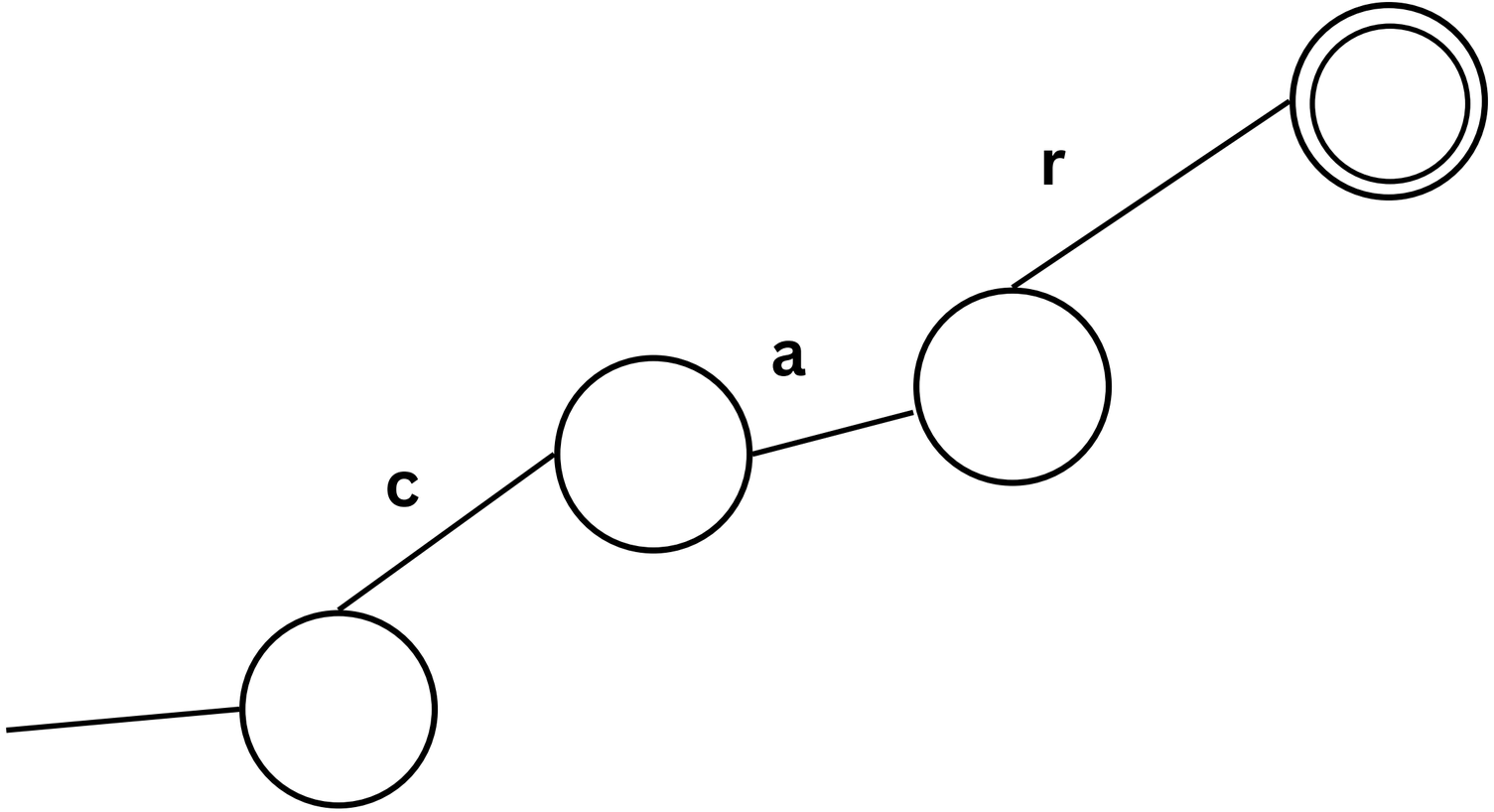
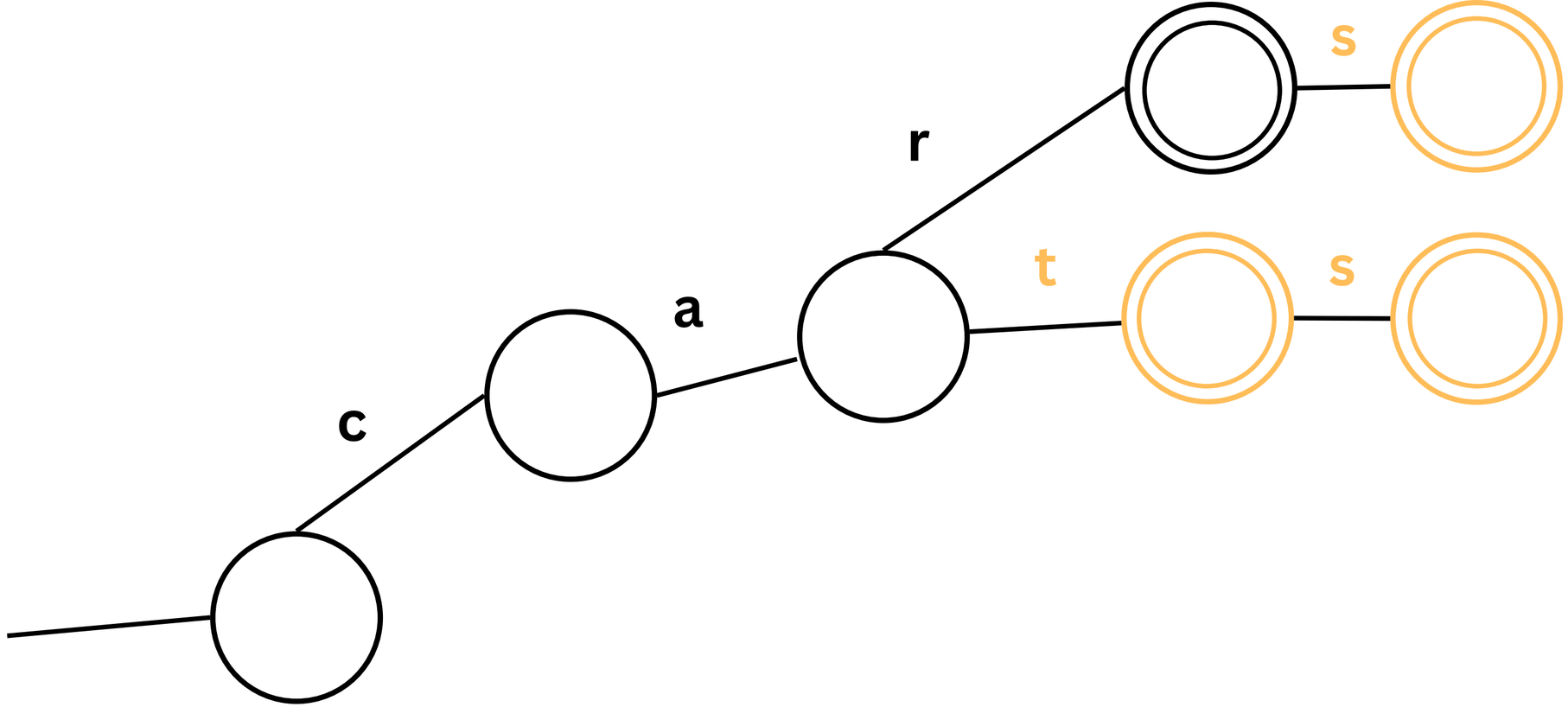
TRIE + *info on the board*

= HIGHEST SCORING MOVE !



Insertion

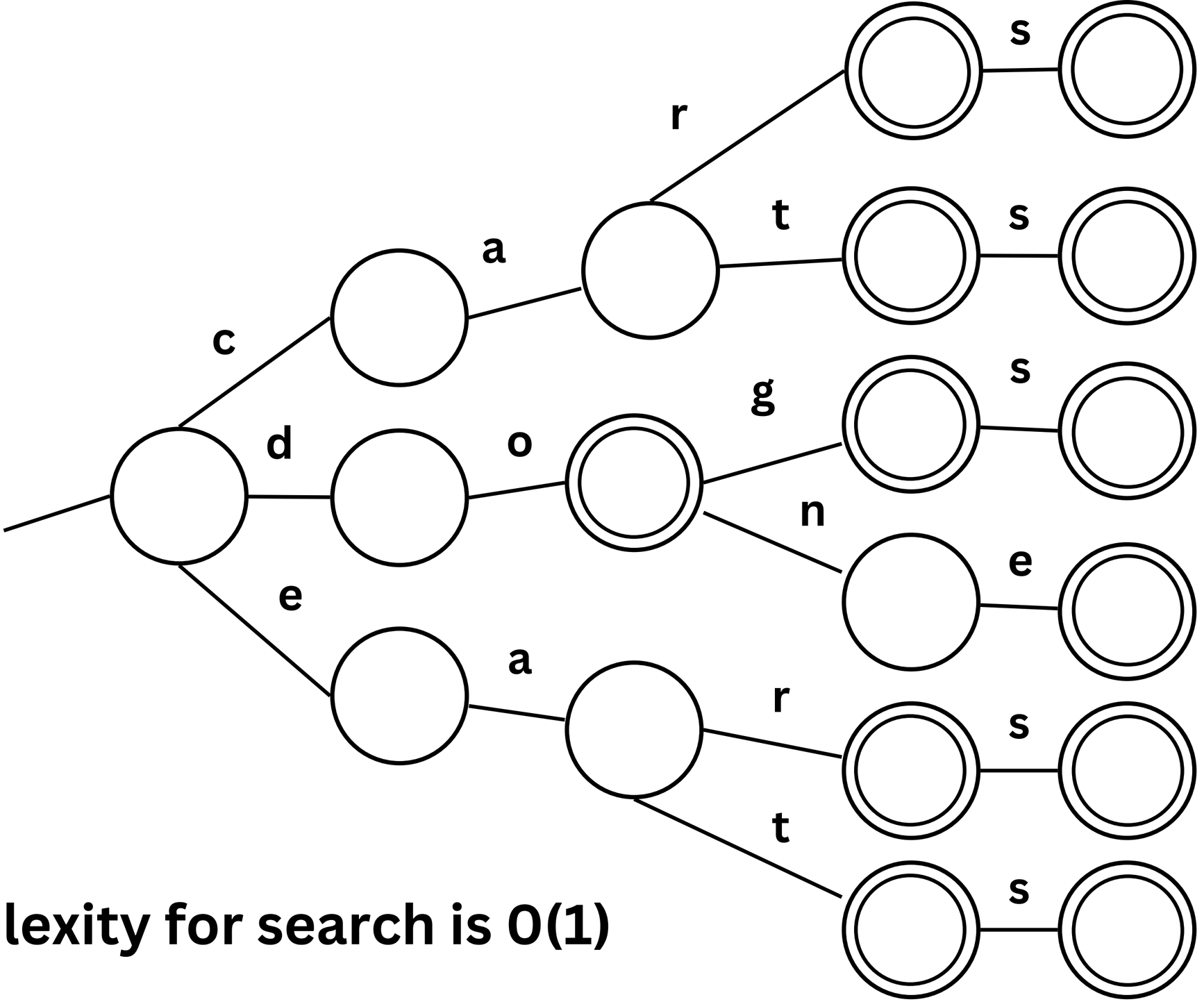


Complexity is for insertion O(1)

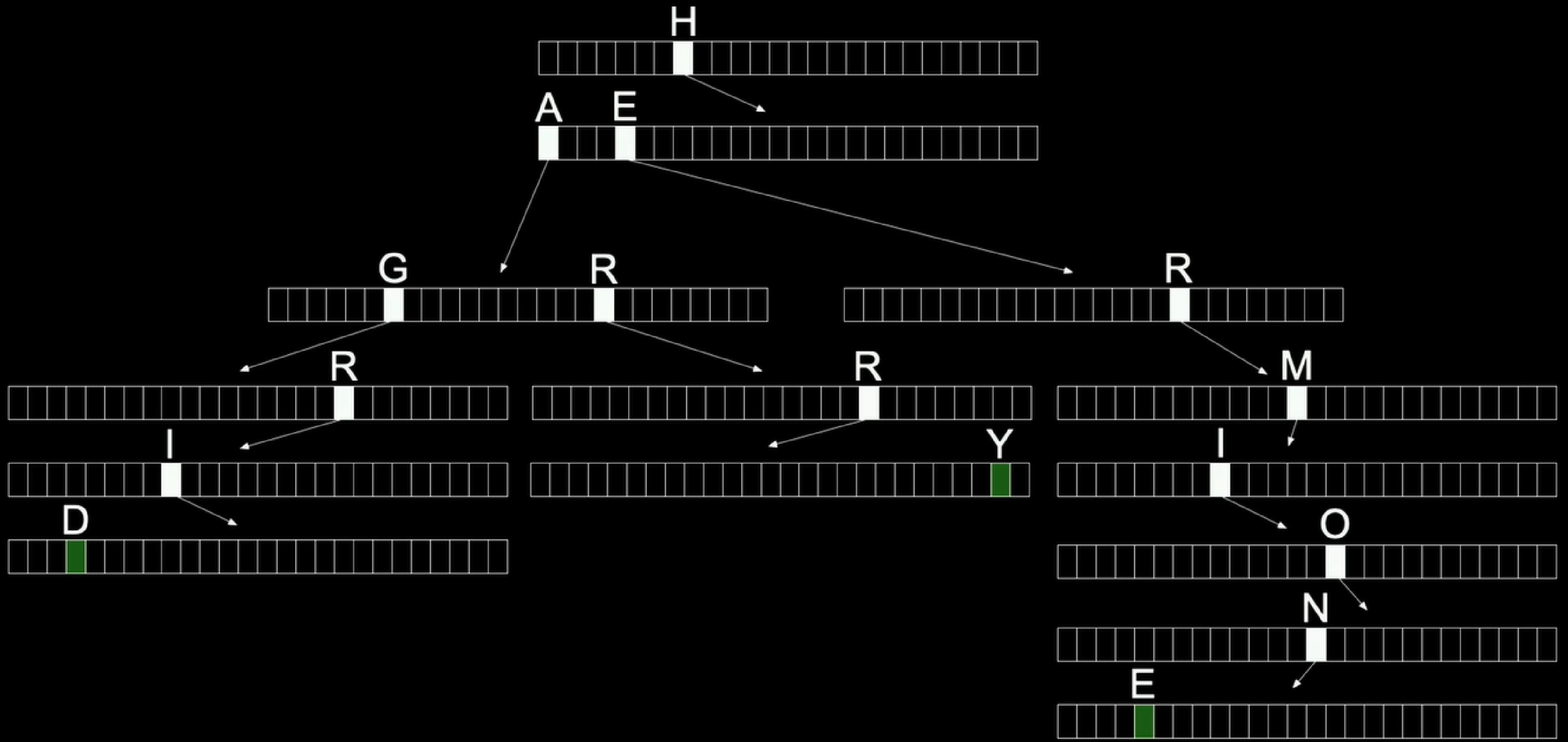
Inserting:
cars
cats

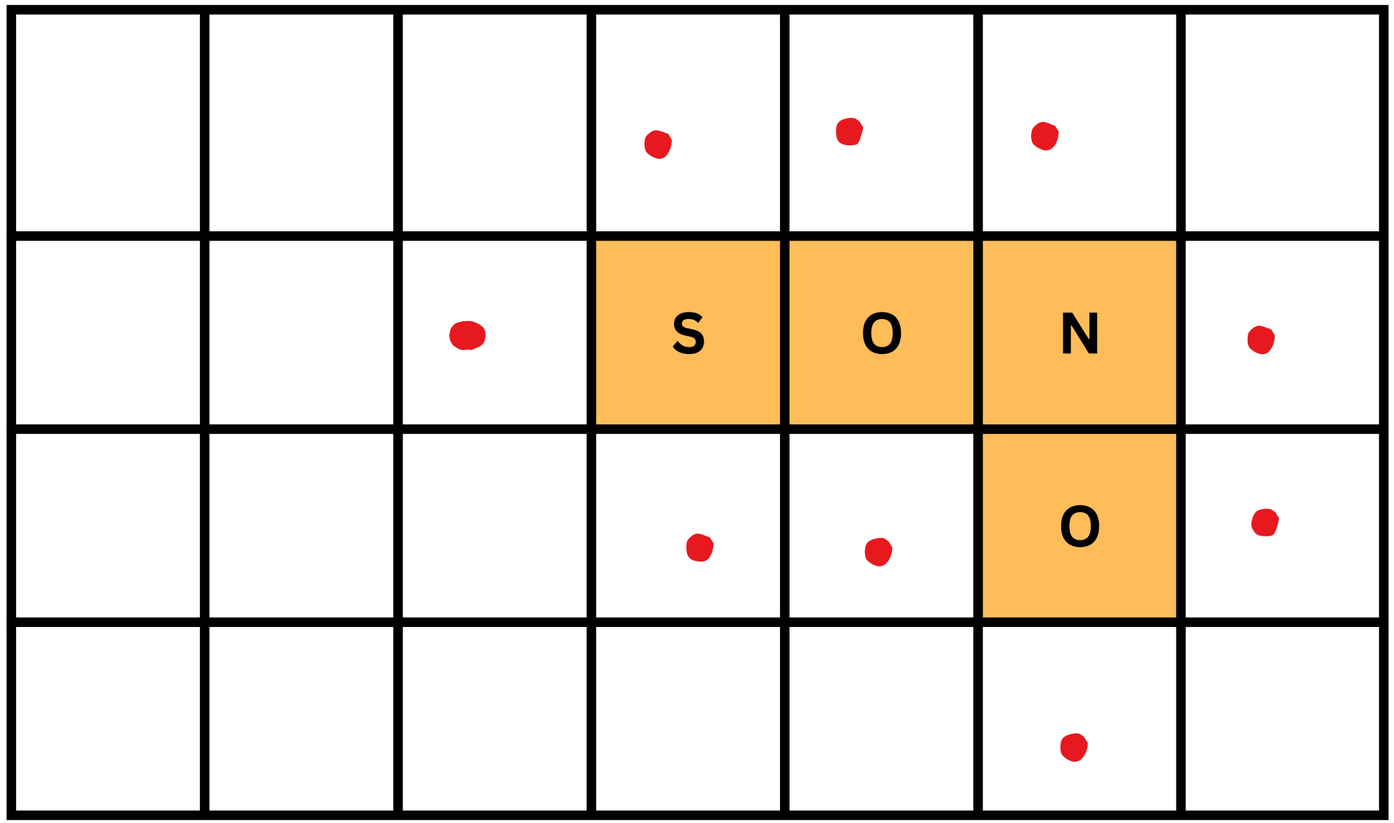
Sample words:

car
cars
cat
cats
do
dog
dogs
done
ear
ears
eat
eats

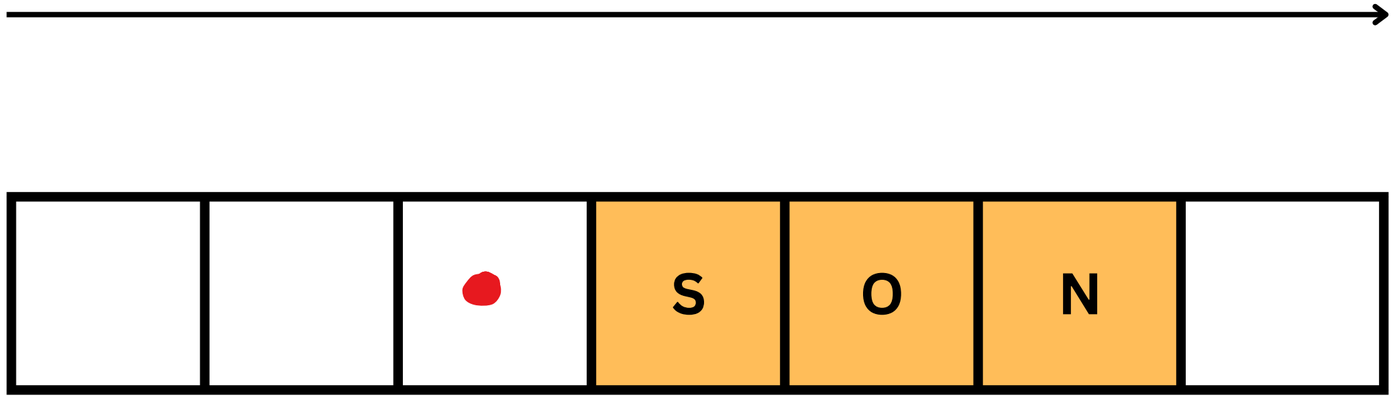


Complexity for search is 0(1)

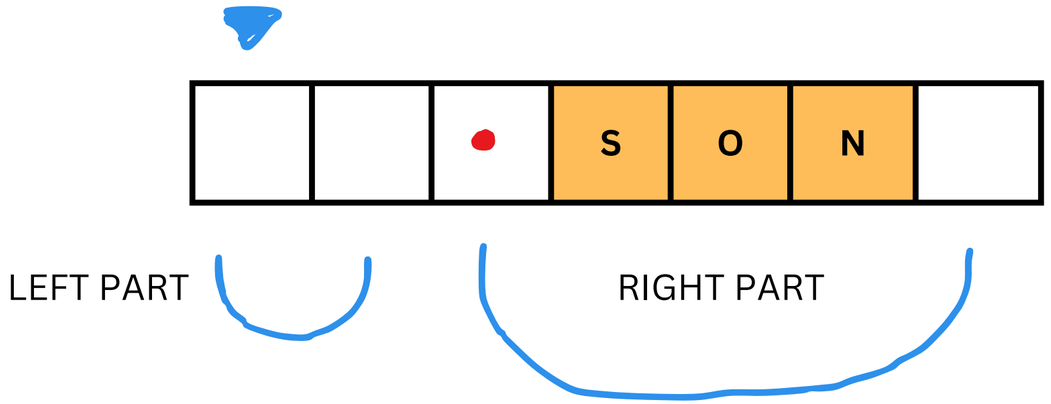

H
A E
G R
R
R R
M
I
Y
I
D
O
N
E


S
O
N
O

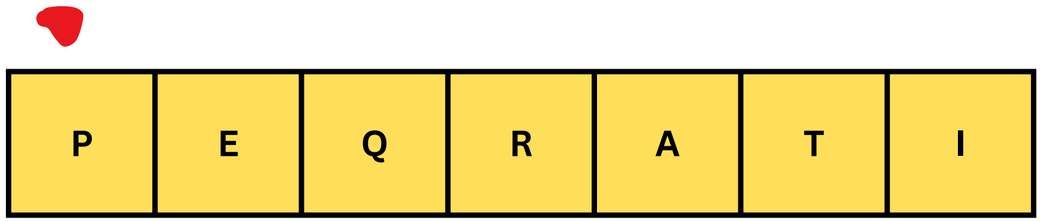
[*]	[ ]	[B, D, G, H, J, L, M, N, O, P, S, T, W, Y]	[A, B, D, E, F, H, J, K, L, M, N, P, T, Y, Z]	[A, E, I, S, U]	[*]
[ ]	C 3	O 1	A 1	T 1	[ ]
[*]	[H]	[D, E, F, H, I, K, M, N, P, R, S, W, X, Y]	[A, B, D, E, G, H, I, L, M, N, R, S, T, W, X, Y]	[A, E, I, O]	[*]

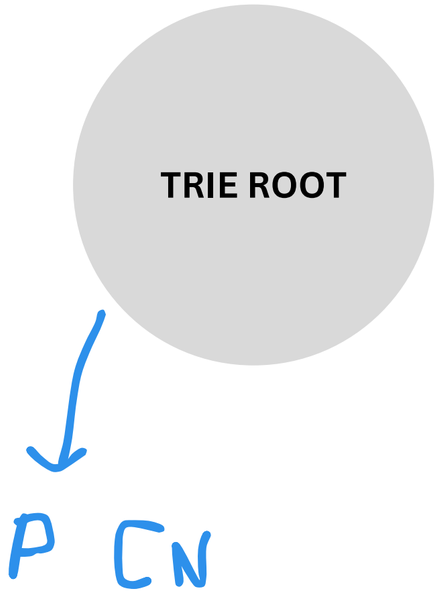

S
O
N

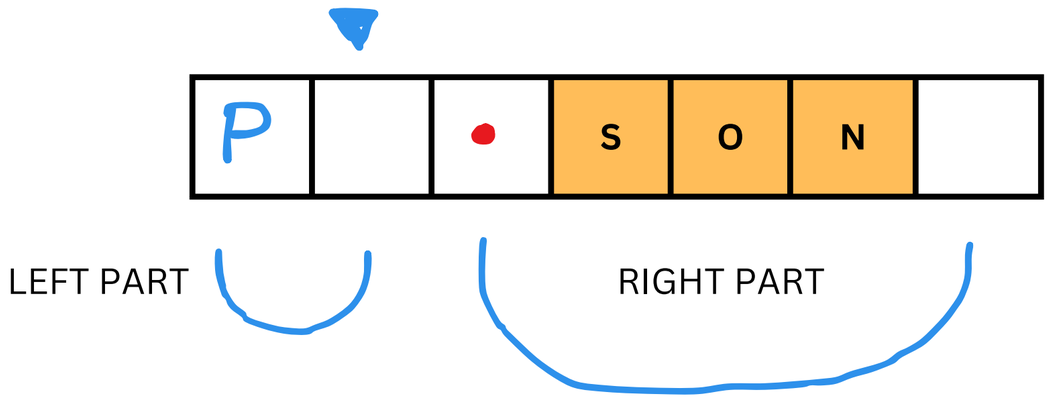
TRIE ROOT

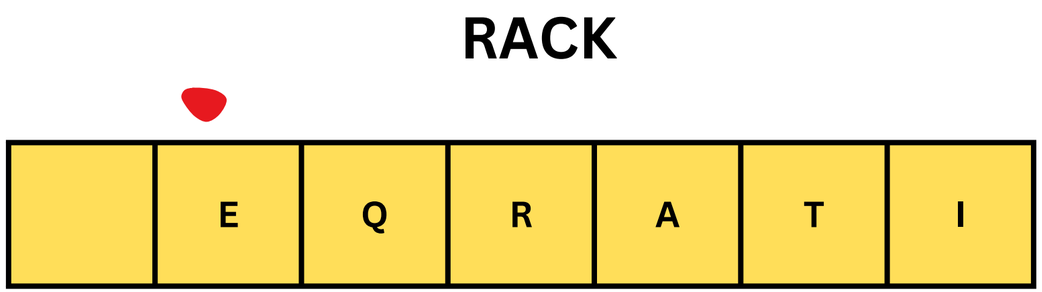


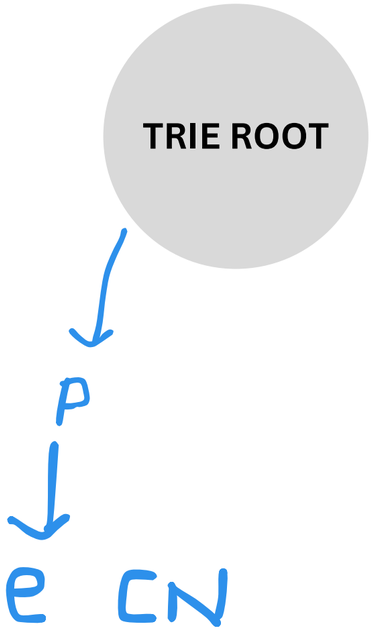
RACK

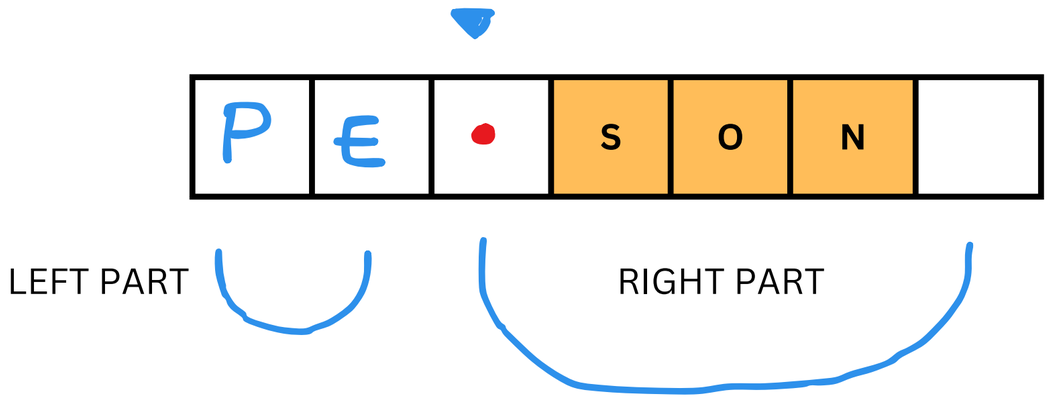


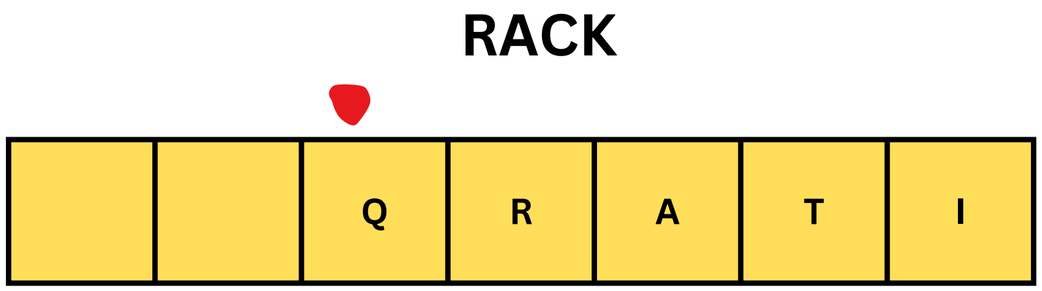

TRIE ROOT
P
N

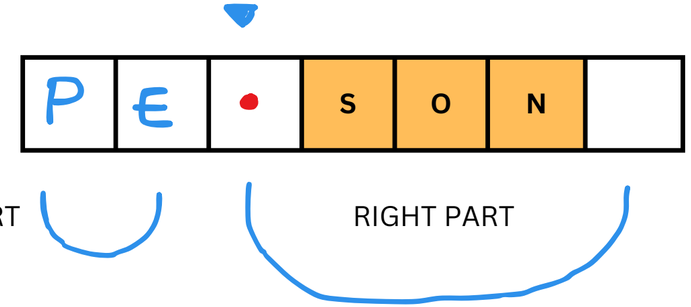

P
S
O
N
LEFT PART
RIGHT PART

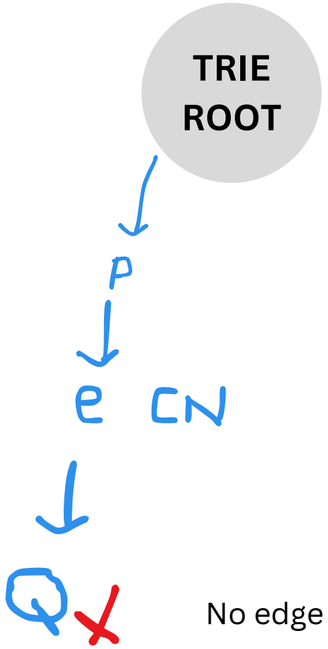

RACK
E
Q
R
A
T
I


TRIE ROOT
P
e
CN


P
E
S
O
N
LEFT PART
RIGHT PART

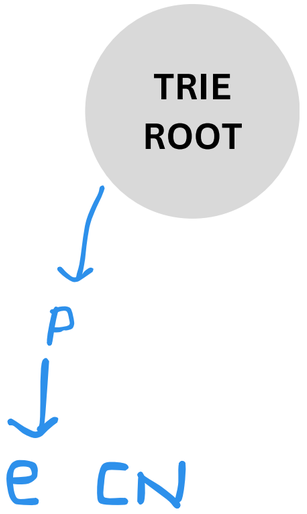

RACK
Q
R
A
T
I





No edge Q comes out of node “E”

put “Q” back to rack and stay on the current node

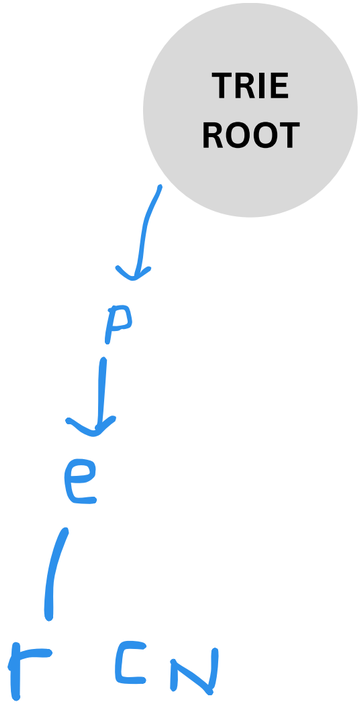


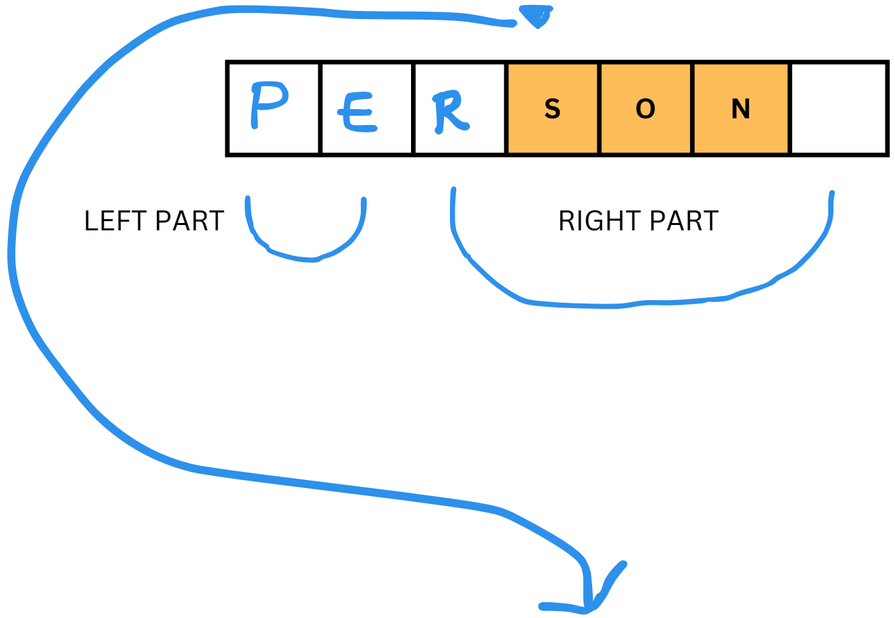
RACK

			R	A	T	I

RACK

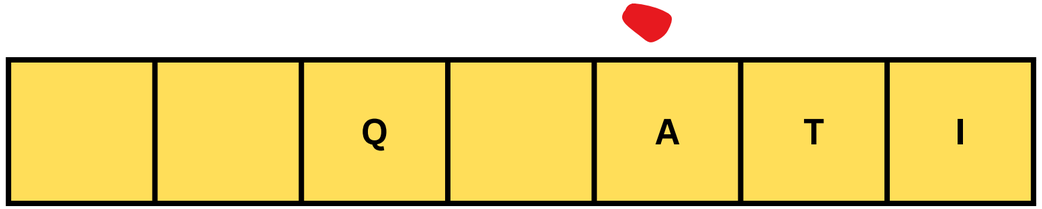
		Q	R	A	T	I

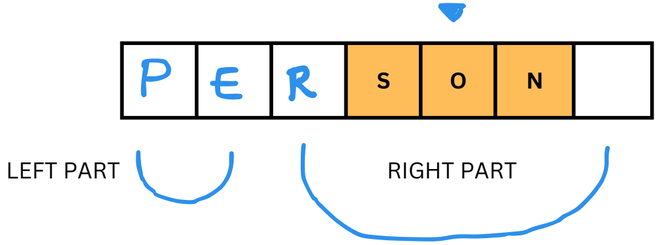


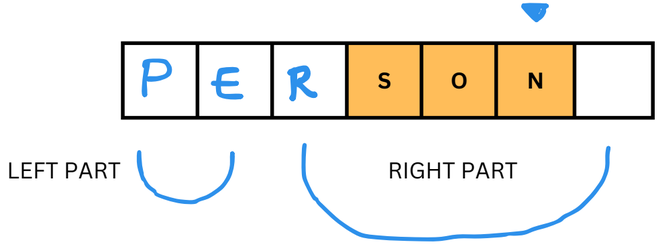


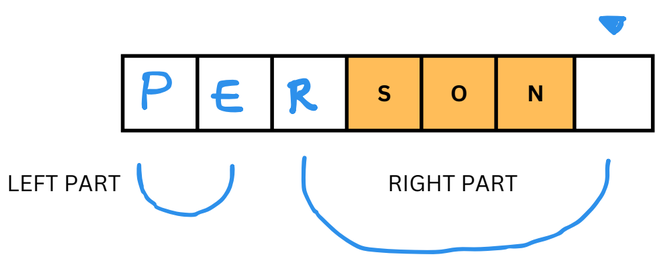
since this cell is occupied we will ignore the tile on rack and search the edge of “*Current node*” with tile on the cell

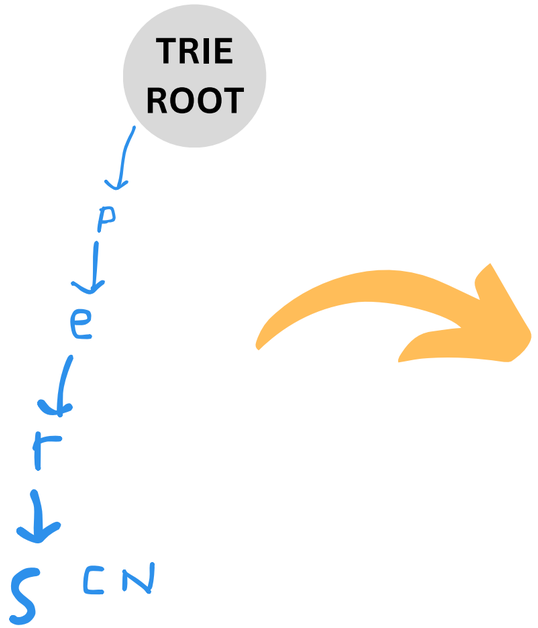
RACK

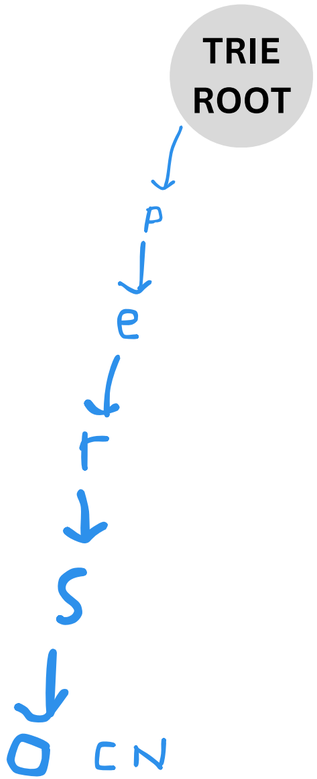


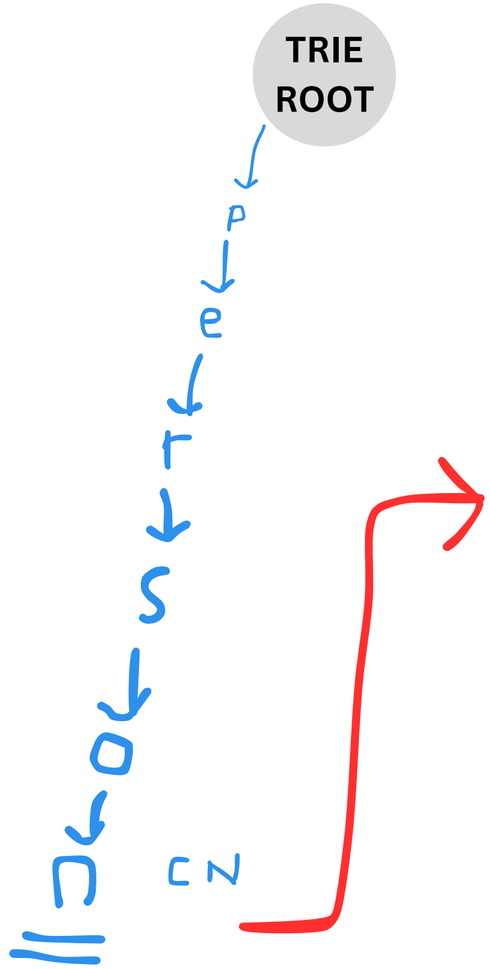












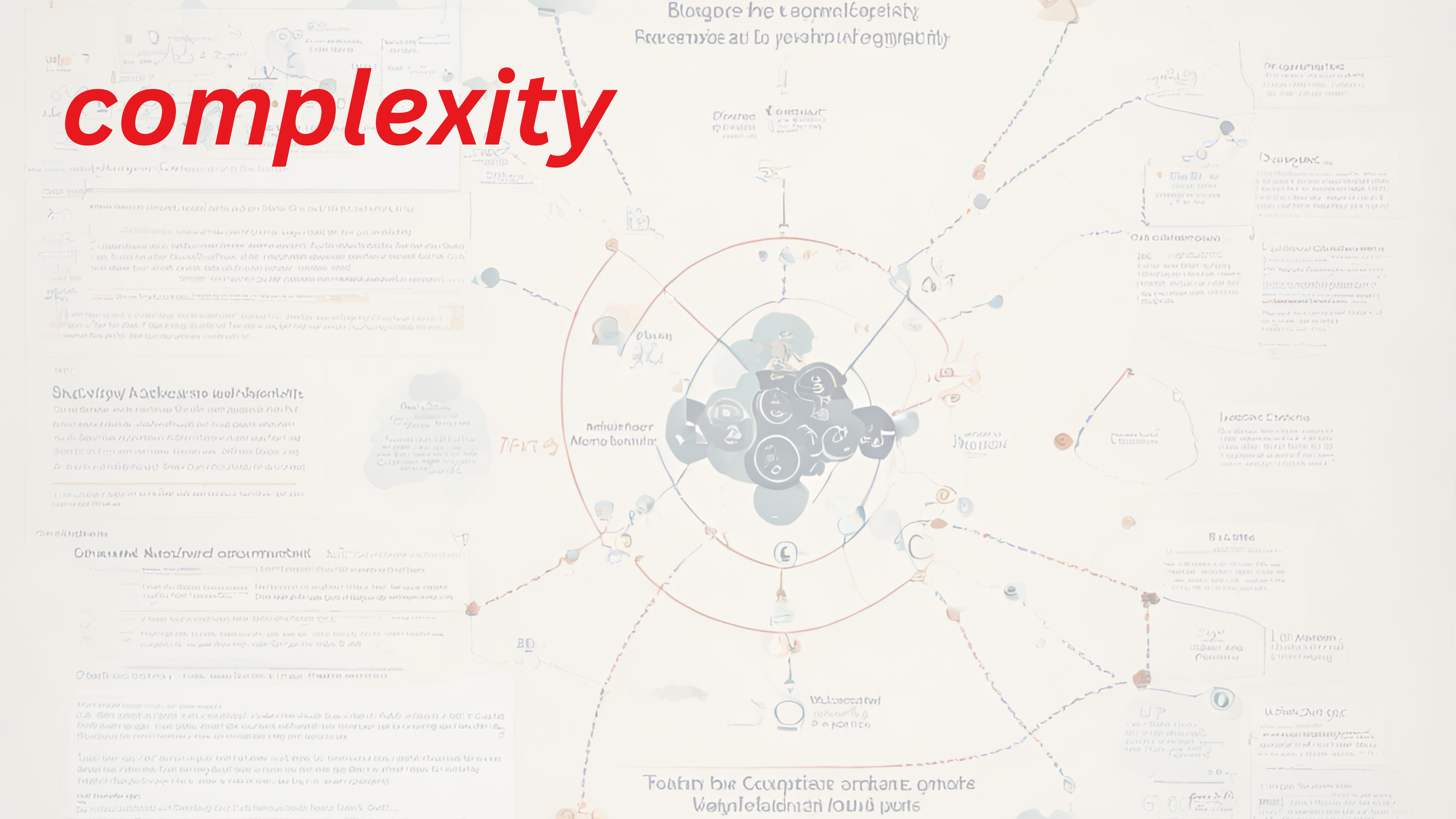
REACHED A VALID WORD ON TRIE, ADD IT TO THE LIST OF VALID WORDS

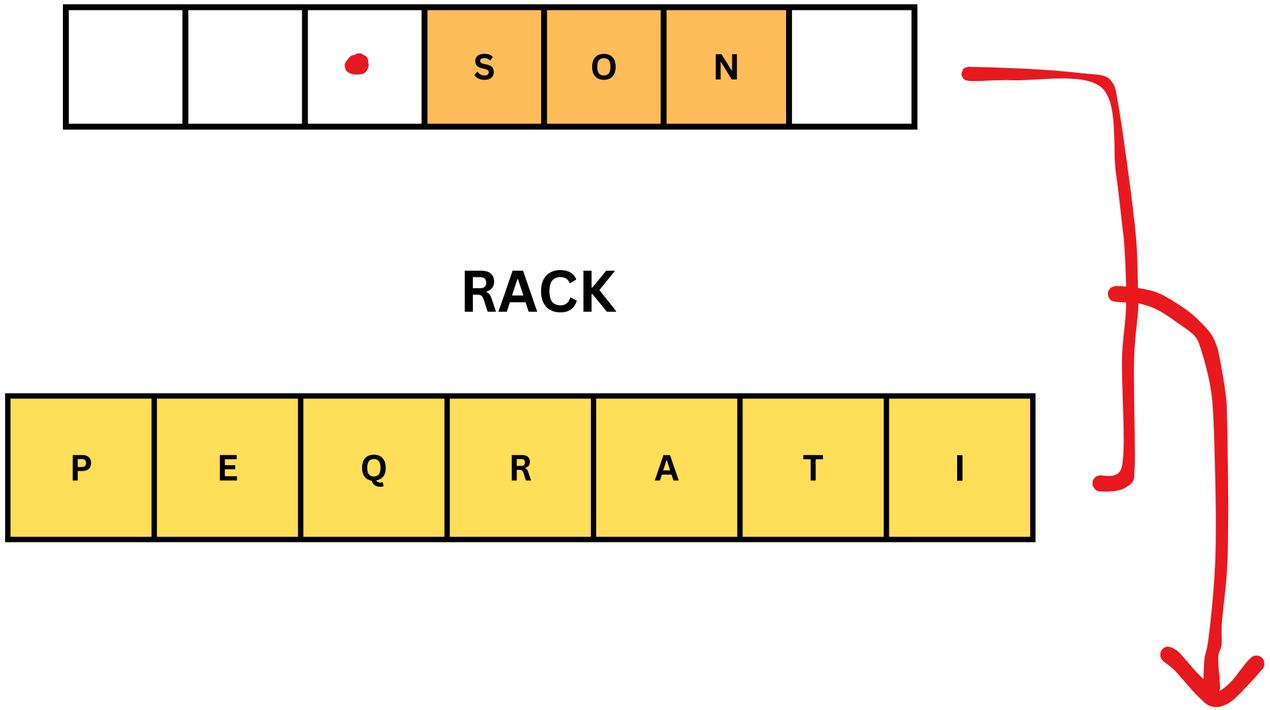
This was just one possible word, since the algorithm is defined recursively it will make all possible words from that particular anchor point.

And if we call this recursive function from every anchor point it will generate all possible words that can be made on the current state of the game

To pick the highest scoring move we calculate the score of each possible move and pick the highest of the lot...

complexity

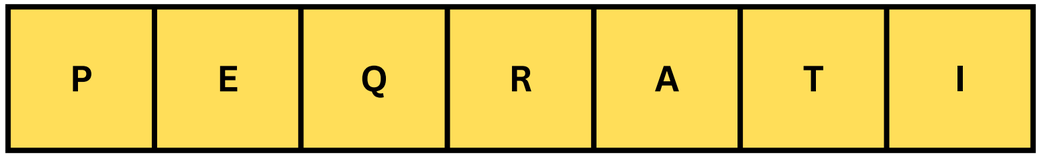




TRIE

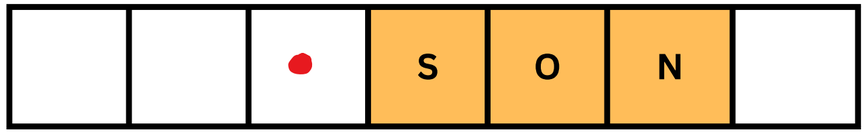
The search can only contain these letters so we don’t have to look for entire ***TRIE*** to look for a word

RACK



Total possible combinations of 7 letter rack is **13692**

But since the algorithm uses ***trie*** we never search all these possible ways because using ***trie*** we know which combinations to not look for



letters already on the board further reduces the search
for example in this case we will ignore any combination that does not contain ***“SON”***

The Article:

https://www.cs.cmu.edu/afs/cs/academic/class/15451-s06/www/lectures/scrabble.pdf