TRIES

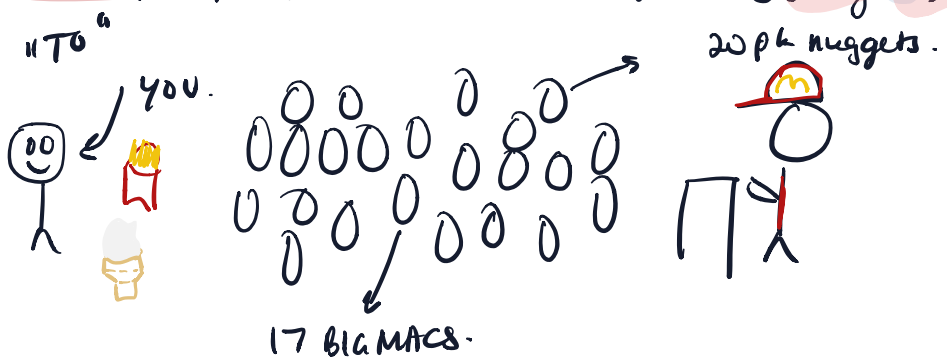
PROBLEM: Sets of strings are important to keep track of, and membership checks should be cheap!

↳ Dictionaries (171 476 ENGLISH words!)

↳ DNA analysis (A, T, C, G order of billions!)

OR SOLUTION Have a balanced binary tree store all the words! Insert, lookup are in $O(\log n)$ where n are the number of words in your set.

↳ **PROBLEM?** What's undesirable with $O(\log N)$



GOAL A structure that can add and search for membership in order of the LENGTH OF THE INPUT STRING.

Terminology: KEY - word we want to search for in our set.

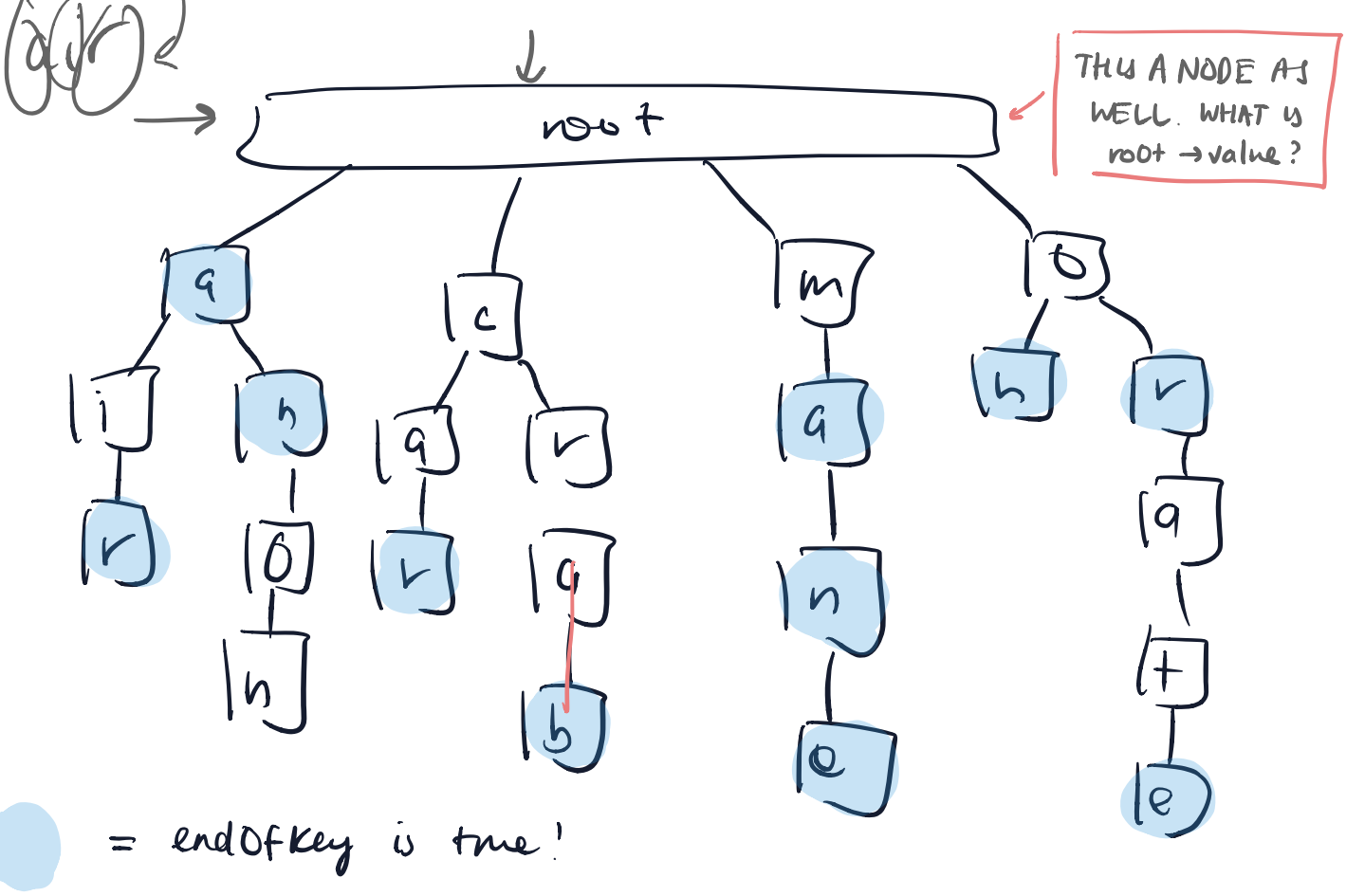
IsMember, Insert, Delete : $O(|K|)$

DESIGN

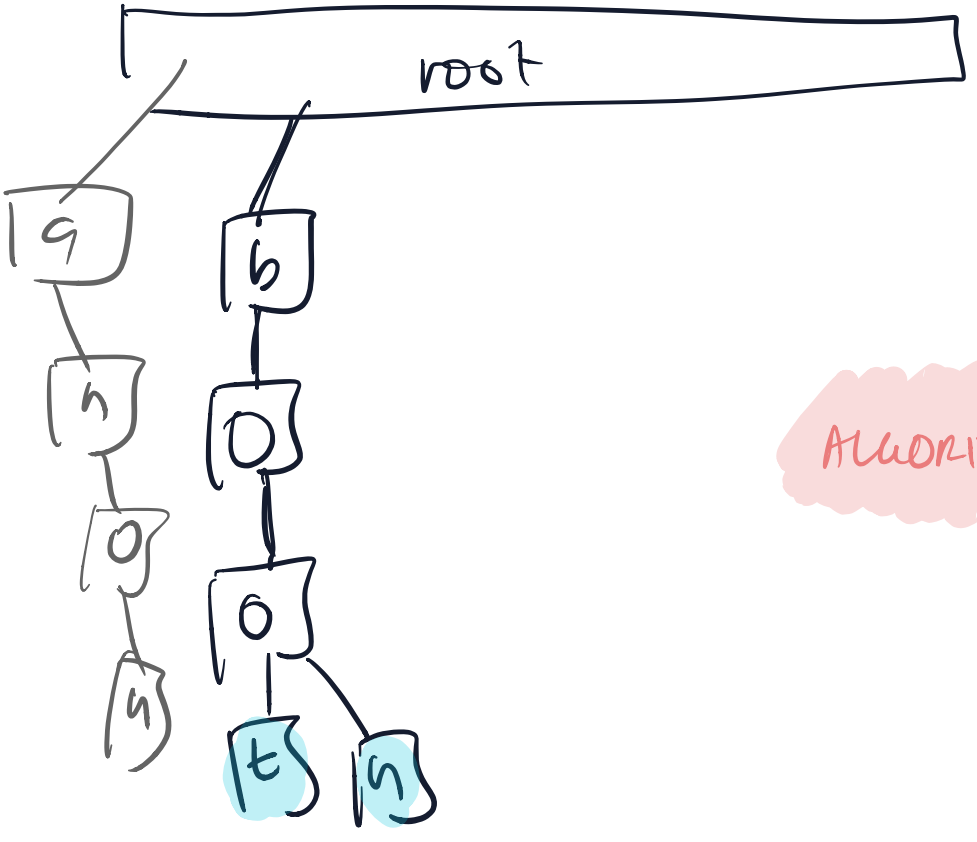
struct TrieNode *

```
bool   endOfKey
char   value
struct TrieNode children[26]
```

TrieNode * — english alphabet, lower case.
what if not a static alphabet?



INSERT - children[0] boot, boon. cajon.



ALGORITHM?

Q: When wouldn't you use a TRIE as a structure for lookups?

PATTERN MATCHING

Problem: Given Text T and pattern P, find the index i s.t. $T[i..|P|] = P$

NAIVE APPROACH? $T =$ "a b a e d e f g h" $P =$ a a b

Complexity? $O(|T||P|)$ P

Space? $O(1)$

T → quadratic in

'a'	'b'
4	3

STRAIGHT FROM THE SLIDES:

① "Looking Glass Heuristic": Compare P with subsequence of T moving backwards

② "Character Jump Heuristic": When a mismatch occurs at $T[i] = c$

HOW ON EARTH IS THIS CORRECT?

EXAMPLE

D = "K A P P A"

T = "O M E G A L U L P H A N K S" pls...

K A P P A

HOW DO WE KNOW WHETHER $T[i] \in P$? HOW MUCH DO WE SHIFT?

Observation: If $T[i] \notin P$, what's the maximum we can shift?

[If $T[i] \in P$, then we want to match up our $T[i]$ with a $c \in P[1...j]$, this might give me a match!

BAD CHARACTER → "A pattern P can be shifted so the right most symbol of x in P is aligned with $x[i]$".

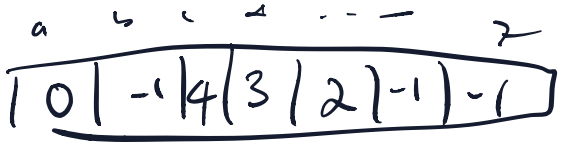
i.e. T ..... x.....

P ....x..x.....

T ..... x.....
P ...c x..x c.....

The mechanism to retrieve this shift is by precomputing a "FUNCTION"

SHIFT :: ALPHABET → ℤ



apple.

Algorithm:

① Produce array of |ALPHABET| size.

② Initialise all elements to −1.

③ For each char c in P, set f[c] = index of c in P.

Why? Update our head of T to be $i := i + |P| - \min(j, 1 + \text{SHIFT}(T[i]))$

Recall: c ∉ P, then move |P|

Otherwise move to next alignment where T[i] is over same char in P.

Examples: SHIFT for P = xywapaswyy

T: A B C A C B B D A B C A D D

P: A B C A D

a a a a a c

c a a a a a

WORSTCASE? T := "aaaaaaaaaaaaa"
P := "baaa"

Complexity? O(?)

Space ? O(?)

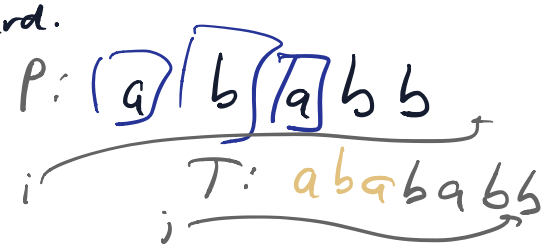
EXAMPLE
HERE U A SIMPLE EXAMPLE

CONSIDER

P := "ababb" and T := "bababbabbq"

① Can walk through our text naively look at a character and if it be beginning of our pattern, move both forward.

② what happens if we don't match?

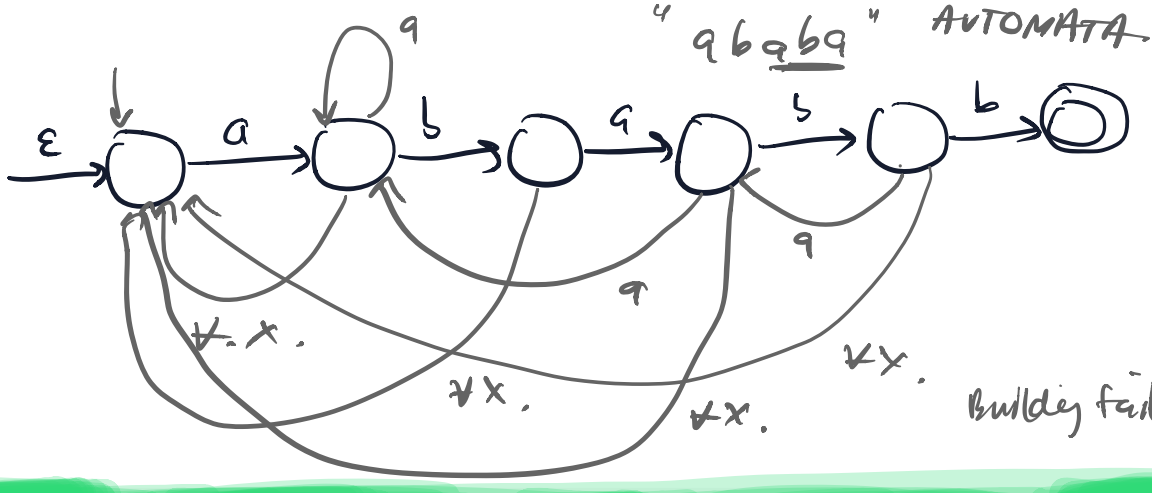


START THE P HEAD AGAIN?

DO CLEVER THINGS? THE "GENIUS" STEP.

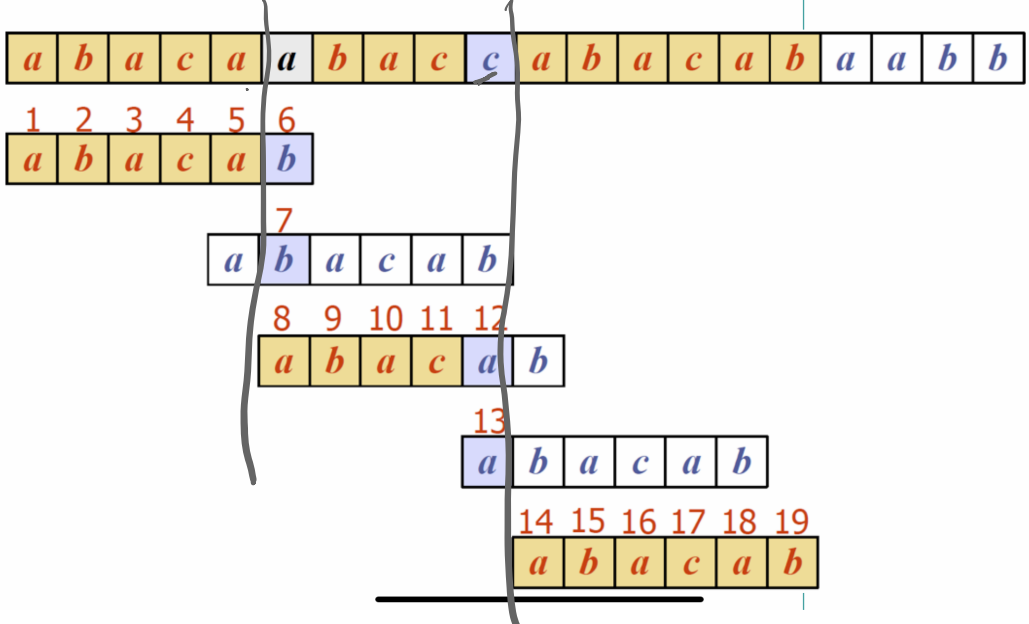
THE CLEVER THING: FAILURE RESOLUTION.

FINITE AUTOMATA



Building failure: suffix matching prefix of match

j	0	1	2	3	4	5
P_j	a	b	a	c	a	b
$F(j)$	0	0	1	0	1	2



COMPRESSION

We represent our integers in the base 2 number system. Something like:

3214567890 ... → ton of bits to rep. $\log(n)$

vs.

16 → 10000 (base 2)

What if we have a ton of our BIG! number in a file, and only one 16? Wasteful to rep. the BIG Number in base 2 if used a ton!

a, b, c, d, e, ...

shorter bit representation.

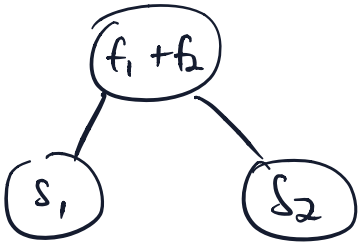
"how much wood can a woodchuck chuck if a woodchuck could chuck wood?"

GOAL: high frequency symbols represented with small number of bits / space.

① Calculate frequency of all the symbols in your collection.

② Make the array of frequencies a MIN heap (by frequency)

③ Extract the two min elements (freq, symbol) make tree:

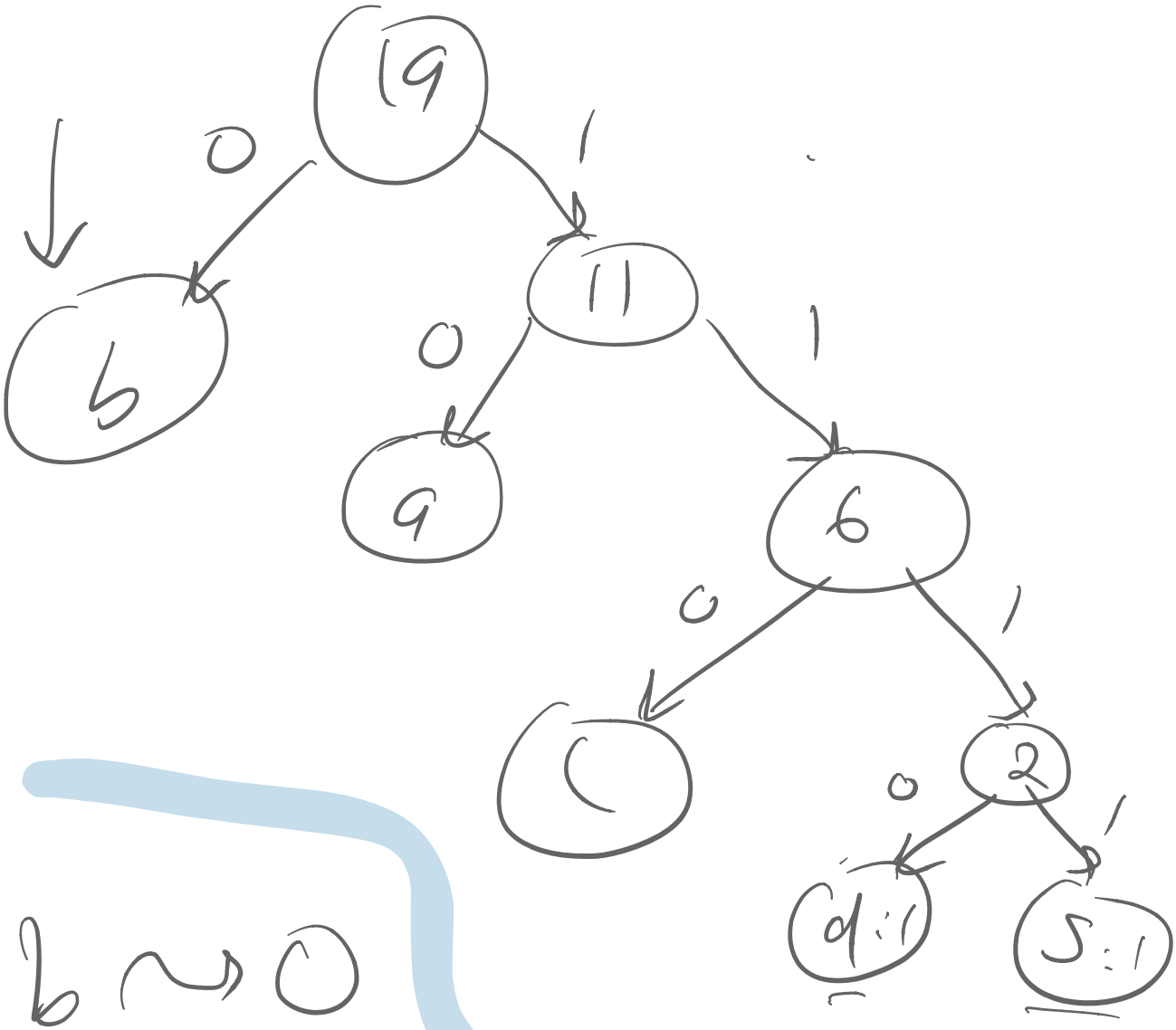


put back into PQ.

④ Repeat until we have no more nodes!

EXAMPLE!

a b a b b c d c b a a b c a c b b b s



b ~ 0

a ~ 10

c ~ 110

d ~ 1110

s ~ 1111

0

bascb.