TABLE DOUBLING, KARP-RABIN

- division method:
  $h(k) = k \bmod m$
- multiplication method:
  $h(k) = \lfloor (a \cdot k) \bmod 2^w \rfloor >> 2^{r-w}$
  where $m = 2^r$

How to choose m?

We want $m = \Theta(n)$

$\Rightarrow \alpha = \Theta(1)$

Idea: start small: $m = 8$
grow / shrink as necessary

If $n > m$: grow table

(how much bigger)

$m' = m + 1$:

what is the cost of n inserts

$\Theta(1+2+3+\ldots+n) =$
$= \Theta(n^2)$

GROW TABLE: $m \to m'$
- make table of size m'
- build new hash f'
- rehash:
  for item in T:
  T'.insert(item)

$\Theta(n+m+m')$

$m' = 2 \cdot m$:

cost of n inserts

$\Theta(1+2+4+8+\ldots+n) = \Theta(n)$

TABLE DOUBLING

Amortization:

- operation takes "T(n) amortized" if k operations take $\leq k \cdot T(n)$ time
- think of meaning "
  ~"T(n) on average", where average over all operations

Table doubling:

k inserts
take $\Theta(k)$ time
$\Rightarrow \Theta(1)$ amortized/insert

Deletion:

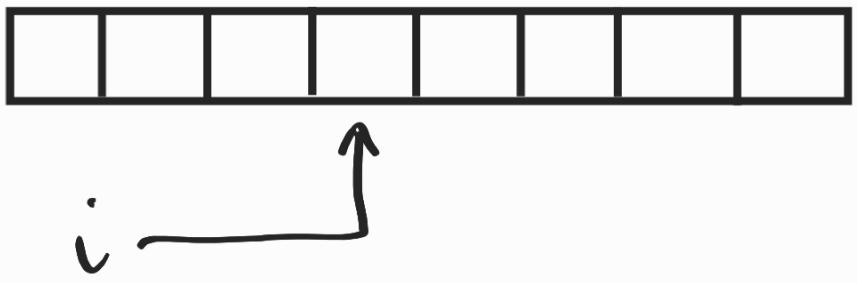
① if $m = \frac{n}{2}$ then shrink -> $m/2$

slow: $2^k \xleftrightarrow[\text{delete}]{k \text{ insert}} 2^k + 1 \Rightarrow \Theta(n)$ per operation

:(

② if $m = \frac{n}{4}$ then shrink -> $m/2$

amortized time $\rightarrow \Theta(1)$



$n \leq m \leq 4n$

list.append $\rightarrow \Theta(1)$
list.pop $\rightarrow \Theta(1)$ amortized
} last element

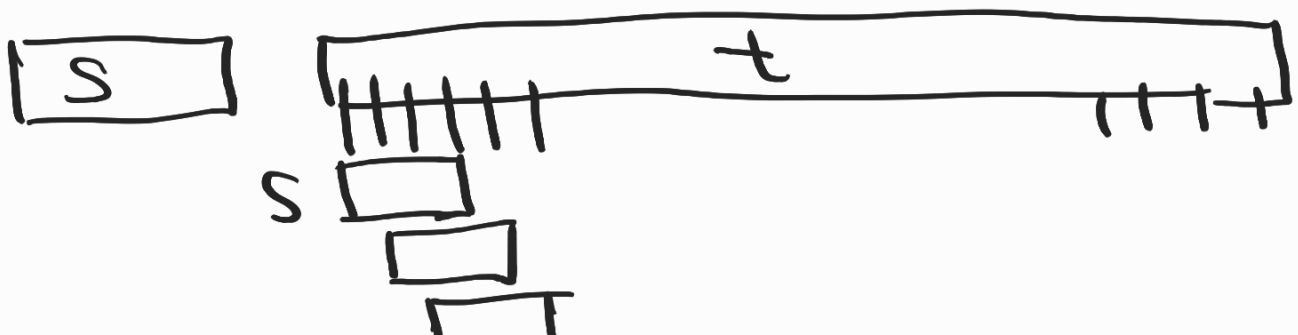
STRING MATCHING:

given two strings s & t does s occur as a substring of t?

s = '6.006' t = INBOX

SIMPLE ALGORITHM:

```
any(s == t[i : i+len(s)]
    for i in range(len(t) - len(s)))
```



time: $\Theta(|s| \cdot (|t| - |s|))$

$=$

$\Theta(|s| \cdot |t|)$

take this time to linear time

$\Theta(|s| + |t|)$

Rolling hash ADT: { r maintains a string x

- r.append(c): add.char(c) to end of x
- r.skip(c): delete first char of x (assuming it is c)
- r(): hash value of $x = h(x)$

KARP-RABIN algorithm

```
for c in S: rs.append(c)
for c in t[:len(s)]:
        rt.append(c)
```

if hs() == ht():

for i in range(len(s), len(t)):
    ht.skip(t[i - len(s)])
    ht.append(t[i])
    if rs() = rt():
        check whether
            s == t[i - len(s) + 1; i + 1]
        if equal:
            found match
        else:
            happens with probability ≤ 1/|s|

O(s)

=> O(l) expected time

O(|s| + |t| + #match · |s|)
expected time

- division method:

$h(k) = k \bmod m$

random prime $\geq |s|$