Table Doubling, Karp-Rabin

- How to choose m for hashing?

  $\alpha = n/m \qquad time = \Theta(1+\alpha)$

  $-$ what $m = \Theta(n) \longrightarrow \alpha = \Theta(1)$

  - Idea:
    - start small: $m = 8$
    - grow/shrink as necessary
      - if $n > m$, grow table

        [Grow table]: $m \to m'$
        - make table of size m'
        - build new hash h' func
        - rehash:

          for item in T:

            T'. insert(item)

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

- Table Doubling
  - if $m' = m+1$:

    cost of n inserts $= \Theta(1+2+3+\cdots+n) = \Theta(n^2)$
  - If $m' = 2m$

    cost of n inserts $= \Theta(1+2+4+8+\cdots+n) = \Theta(n)$

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 insertions take $\Theta(k)$ time

↳ $\Theta(1)$ amortized insert

Also: k insert & deletes take $O(k)$ time.

Deletion

① if $\frac{m}{2} \geq n$ then shrink $\rightarrow m/2$

slow if go $2^k \underset{\text{Delete}}{\overset{\text{Insert}}{\rightleftarrows}} 2^k+1$

↳ $\Theta(n)$ per op. :(

② if $\frac{m}{4} \geq n$, then shrink $\rightarrow \frac{m}{2} \rightarrow n \leq m \leq 4n$

amortized time $\Theta(1)$

Python list

list.append()
list.pop()

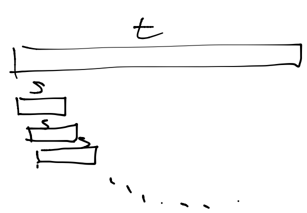
} $\Theta(1)$ amortized

String matching

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

∘ Simple Algorithm

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



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

goal: use hashing to get above done in linear time.

∘ Rolling hash ADT (Karp−Rabin algorithm)

− given a rolling hash value r, r. append(c). r matains a string x, r.append(c) add c to the end of x.

− r.skip(c): delete first char of x (assuming it is c)

− r( ): give hash value of x = h(x)

for c in s: rs.append(c)   [rs: rolling hash of s]

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

[rt: rolling hash of first len(s) chars of t.]

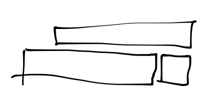
if rs() == rt(): ......

same

for i in range(len(s), len(t))
    rt.skip(t[i - len(s)])
    rt.append(t[i])
    if rs() == rt():
$O(s)$ ⟶ check whether $s == t[i - len(s) + 1 : i + 1]$
        if equal: found match
        else:
            (happens with probability $\leq \frac{1}{|s|}$

Total $O(|s| + |t| + \#\text{match} \cdot |s|)$ linear time

- Implement hash function
  - Division method
    $h(k) = k \bmod m$
    m: random prime $\geq |s|$
  - Treat x as multidigit number u in base a (alphabet size)
    - r.append(c)



$u \to u \cdot a + \text{ord}(c)$

    - r.skip(c) $u \to u - c \cdot a$

- $r \longrightarrow (r \cdot a + \mathrm{ord}(c)) \bmod m$