

Talks next Thu, Tue, Thu
May 4, 6, 8

15-20 min each
Schedule on web

via Zoom
in person

Ultra-wide word RAM [2015]

memory cells = w -bit words usual operations in $O(1)$ time

Plus ultra-words with w^2 bits each
(w words)

small number

usual ops in $O(1)$ time

plus scattered read/write

real world: u words
have 1000s of bits

$O(1)$ time

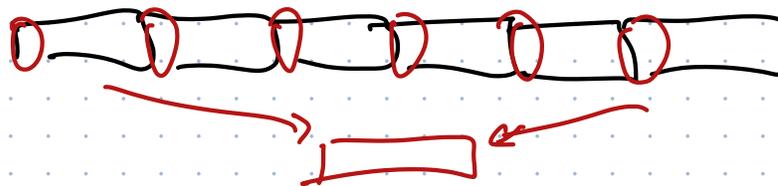
read: given u word $A = [A_0, A_1, \dots, A_{w-1}]$
get $Z = [\text{Mem}[A_0], \text{Mem}[A_1], \dots]$

write: given A, D
write $D[i]$ into $\text{Mem}[A[i]]$ for all i

Useful operations:

compress: Given X , compute word x
s.t. $x_i = \text{left bit of } X_i$

Pack Sign Bits



componentwise arithmetic

$$X \oplus Y = [X_i + Y_i \bmod Z^w]_i$$

$$X \ominus Y$$

$$X \ominus Y = [[X_i < Y_i]]_i$$

(?) $X \otimes Y = Z$

where $Z_{2i} = X_{2i} \cdot Y_{2i} \bmod Z^w$
 $Z_{2i+1} = X_{2i} \cdot Y_{2i} \text{ div } Z^w$
($\gg w$)

assume $X_i = Y_i = 0$ if i is odd

Bille Gørtz Stordalen '24

ordered dict with $O(1)$ time pred, succ, insert, delete
sum, exp

w-parallel hashing

Maintain a set S of n w -bit words s.t.

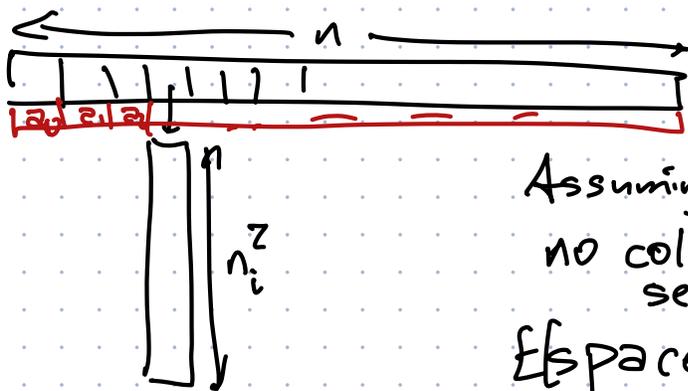
- insert(x)
- delete(x)
- pMember(X)

return word $\mathcal{I} = [x_i \in S]_i$

$O(n+w)$ space $O(1)$ query time $O(1)$ am. ex. update time

It's a hash table.

Two-level "perfect" hashing [FKS '84]



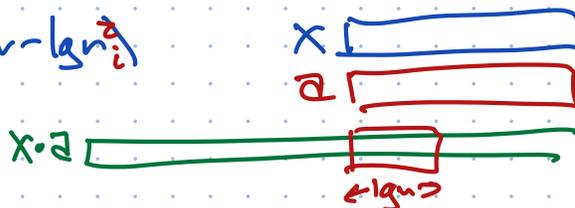
Assuming ^{near} 2-uniform hashing ✓
 no collisions in secondary tables
 $\mathbb{E}[\text{space}] = O(n)$

multiply-shift hashing

$$h(x) = (a \cdot x \bmod 2^w) \gg (w - \lg n)$$

$$h_i(x) = (a_i \cdot x \bmod 2^w) \gg (w - \lg n_i)$$

random odd words



Given X :

Compute $\mathcal{I} = [h(X_k)]_k = [a \cdot X_k \bmod 2^w \gg (w - \lg n)]_k$

\uparrow
 $A \otimes X$

Compute $\mathcal{J} = [h_{\mathcal{I}_k}(X_k)]_k$

$A' \otimes X \otimes M$

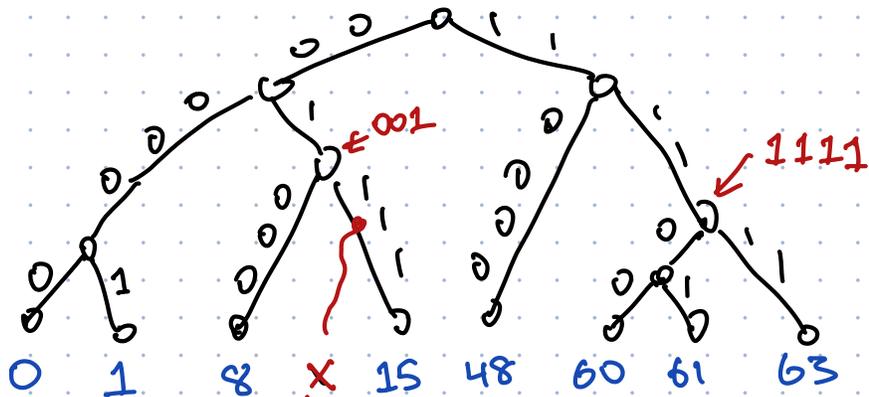
\nwarrow scattered read \uparrow shifts

Return $\mathcal{I} = [\mathcal{I}_k]_k$ $\mathcal{J} = [\mathcal{J}_k]_k$ $Q = [\mathcal{I}_k = x[k]]_k$

extra-fast trie

Assume n $(w-1)$ -bit words

High-level design: compressed trie



$str(v)$ = label of path from root to v

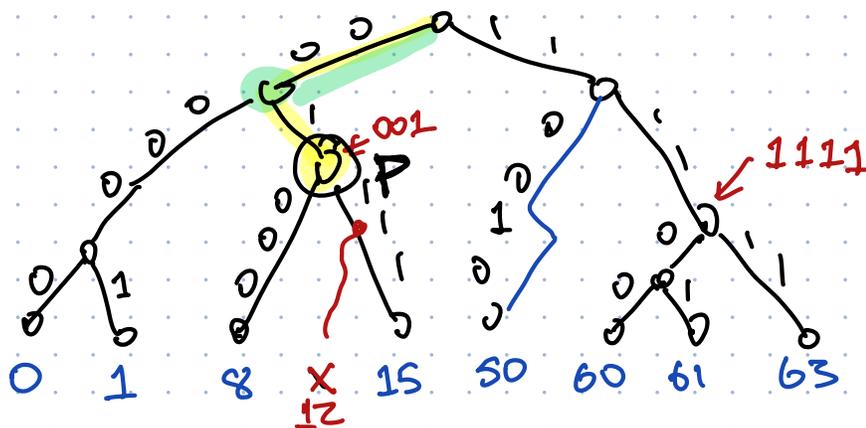
$min(v)$ = min leaf below v
 $max(v)$ = max

(keys in S)

- ① Doubly-linked list of leaves in sorted order
- ② w -parallel hash table for $\{str(v) \mid v \in T\} \leftarrow$ keys
 $(min(v), max(v)) \leftarrow$ data

Pred(x):

- ① Build an w -word containing all prefixes of x



- ② Lookup prefixes in hashtable, compact \rightarrow $\boxed{000110}$
 $msb \rightarrow z$

- ③ $pred(x)$ is either $p.left.max$ or $p.right.min.prev.$
or $p.left.max.prev$

Dynamic RMQ

$O(\log \log \log n)$ per query / update