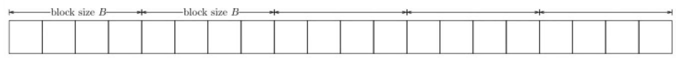
Hashing With Probing

Data Structures

- The dictionary problem: maintain a dynamic set S of ***keys*** (with associated data)
  - Insert(x) : add x to S
  - Delete(x) : delete x from S
  - Lookup(x) : is $x \in S$? (Return ptr. to associated data if yes.)
- **Hashing w/ Chaining**
  - $O(1)$ time ***in expectation***, with simple hash functions.
- **Cuckoo Hashing**
  - $O(1)$ ***worst case*** per delete, lookup. Requires somewhat stronger hash functions.
- **Hashing w/ *Linear Probing***
  - $O(1)$ ***in expectation***, but much more ***cache-efficient***.

Cache-efficiency

- Cache misses (not memory accesses) is usually the best proxy for running time in practice.
- Data automatically moved from main memory to cache (L1/L2/L3) in contiguous blocks of size B.



Hashing with Linear Probing

- Array $A[0..N-1]$
- Hash function $h: [u] \to [N]$. **(Assume Ideal Hash Model...)**
- All array cells initially NULL.
- Insert(x) : Find first index i in the sequence $h(x), h(x)+1, h(x)+2, \ldots, h(x)+N-1$ (all mod N) such that $A[i] = $ NULL. Set $A[i] := x$.
- Lookup(x) : Find first index i in the sequence $h(x), h(x)+1, h(x)+2, \ldots, h(x)+N-1$ (all mod N) such that $A[i] = $ NULL. Return "true" if x appears in $A[h(x)..i-1]$ and "false" otherwise.

↳ ignore deletes

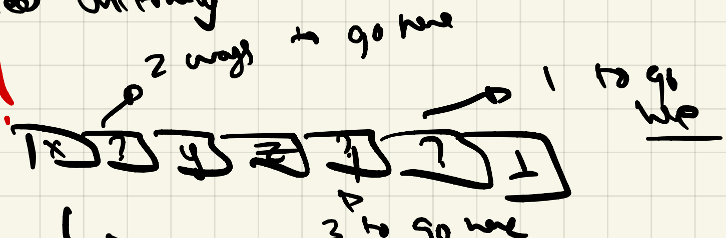
add a special icon @ which is null for insert but not lookup

(or rehash keys after but before next ⊥ — null)

- Assume at most n Inserts, $N = 2n$.

Can we assume that cells are occupied uniformly across $\binom{N}{n}$ possibilities? NO!

Co insertion skewed towards clump
↳ long clumps get longer
↳ merging clump is quite likely



↳ Clumps grow proportionally to their length

- A "run" J is a contiguous interval of keys. It exists if:
  (1) exactly $|J|$ keys are hashed to J.
  (2) Nothing is hashed to cells on either side of J.
  (3) At least j keys hashed to first j cells of J.
- Total insertion cost for a run J is at most $|J|^2$.

trying to bd time for all n inserts

how much did this contribute

↳ well at worst $\frac{|J|^2}{2}$

they all hashed to the same first value

↳ now, lets see prob of seeing this run.

↳ upper bd by this

Analysis of Linear Probing

- **Def.** $I_{i,k}$ is the indicator for the event that a run of length k starts at $A[i]$, i.e., $A[i..i+k-1]$ is a run.
  total insertion time $\leq \sum_{i=0}^{2n-1} \sum_{k=1}^{n} I_{i,k} \cdot k^2$
  $E[\text{total insertion time}] \leq \sum_{i=0}^{2n-1} \sum_{k=1}^{n} E[I_{i,k}] \cdot k^2$
- $p_k = E[I_{i,k}] = \Pr(J = A[i..i+k-1] \text{ is a run})$.
  $\leq$ (# ways to choose k keys)
  $\times \Pr(\text{those } k \text{ keys hashed to } J)$
  $\times \Pr(\text{remaining } n-k \text{ keys not hashed to } J)$
  $= \binom{n}{k} \cdot \left(\frac{k}{2n}\right)^k \left(\frac{2n-k}{2n}\right)^{n-k}$ → binomials

this is an upper bd as we dont consider not hashing to end pts

It was shown 5-wise enough

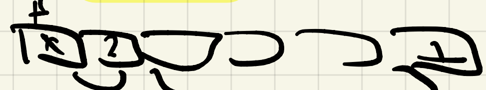
due to geom decay

assumes n-wise independence

↳ can show k-wise suff

& k = log(n)

this case



will merge J

- $\binom{n}{k} = \frac{n!}{k!(n-k)!} \leq \frac{n^n}{k^k (n-k)^{n-k}}$ (follows from Stirling's approx.)
- $p_k \leq \frac{n^n}{k^k (n-k)^{n-k}} \cdot \frac{k^k}{(2n)^k} \cdot \frac{(2n-k)^{n-k}}{(2n)^{n-k}}$
  $= \frac{n^n}{k^k (n-k)^{n-k}} \cdot \frac{k^k}{2^k n^k} \cdot \frac{(n-k/2)^{n-k}}{n^{n-k}}$
  $= \left(\frac{1}{2}\right)^k \cdot \left(1 + \frac{k/2}{n-k}\right)^{n-k}$
  $\leq \left(\frac{1}{2}\right)^k e^{\frac{k}{2}} = \left(\frac{\sqrt{e}}{2}\right)^k < 0.83^k$

↳ decays geom as $\frac{\sqrt{e}}{2} < 1$

- $E[\text{total insertion time}] \leq \sum_{i=0}^{2n-1} \sum_{k=1}^{n} E[I_{i,k}] \cdot k^2$
  $\leq \sum_{i=0}^{2n-1} \sum_{k=1}^{n} \left(\frac{\sqrt{e}}{2}\right)^k \cdot k^2$ → inserting n elts
  $= \sum_{i=0}^{2n-1} O(1) = O(n)$

=> ave insertion time cost