#ASLI ENGINEERING

Quadratic Probing in Hash Tables

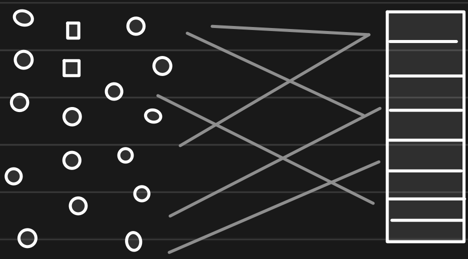
SWIPE

BY

ARPIT BHAYANI

Conflict Resolution with Quadratic Probing

Conflicts are inevitable!



With Open Addressing, we use a probing function to find the slot where the key should be placed

One such method is Quadratic Probing

Probing Function

Probing Function is defined as $p(k, i) = j$ ← index

key → k, attempt → i

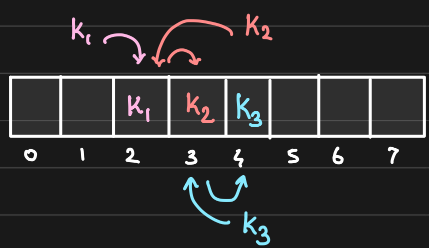
we use the probing function to find the first available slot

the same function is used during lookups

Challenges with Linear Probing

Linear probing suffers from cascading collisions

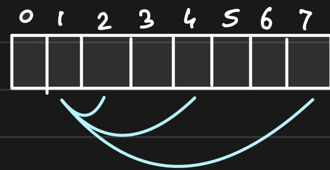
if k_1 hashes to 2
k_2 hashes to 2
k_3 hashes to 3



ARPIT BHAYANI

Quadratic Probing

Instead of placing the collided key in the neighbouring slot, quadratic probing adds successive value of an arbitrary quadratic polynomial.



$$p(k, i) = h(k) + i$$

Linear Probing

$$p(k, i) = h(k) + c_1 i + c_2 i^2$$

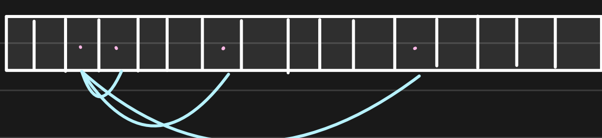
Quadratic Probing

Sample sequence:

$$h(k), \quad h(k) + 1^2, \quad h(k) + 2^2, \quad h(k) + 3^2, \ \ldots$$

How is it better than linear probing?

It reduces clustering and cascaded collisions as collided keys are placed further away from each other



ARPIT BHAYANI

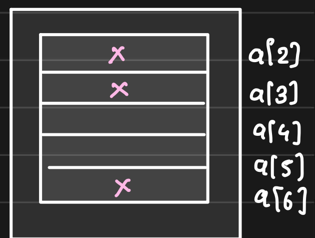
Properties of Quadratic Probing

1. It reduces clustered collisions by distributing it quadratically

   * It is not immune to it, but still it reduces it to a good extent

2. It has a good locality of reference but not as great as linear probing

   * leverages CPU cache well



Two slots are already cached on CPU

$h(k) = 2, \quad h(k) + 1^2 = 3, \quad h(k) + 2^2 = 6$

Unless there are large collisions on same key, at least a couple of subsequent slots will be in CPU cache.

ARPIT BHAYANI