Hashing & Dictionaries

Dictionary

1. Search → Static
2. Insert
3. Delete

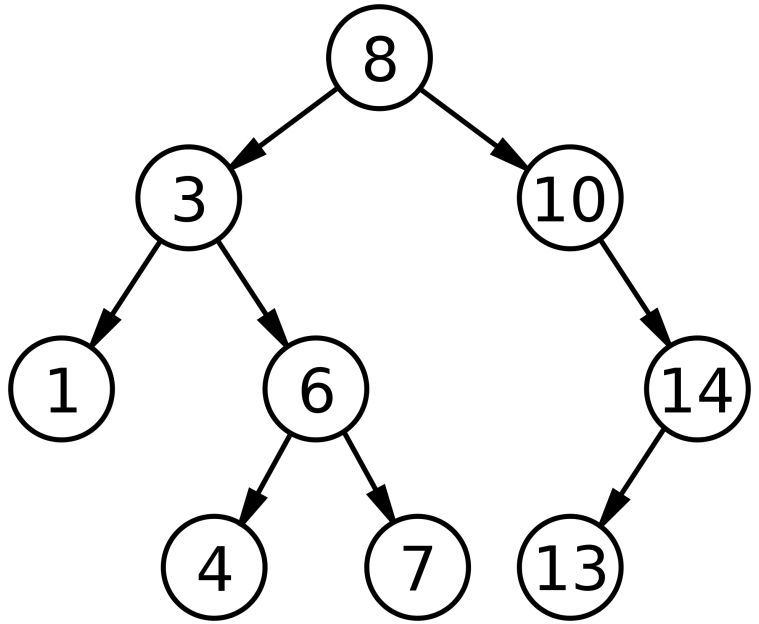
} Dynamic

Dictionary

1. Search → Static
2. Insert
3. Delete

Dynamic

Depth



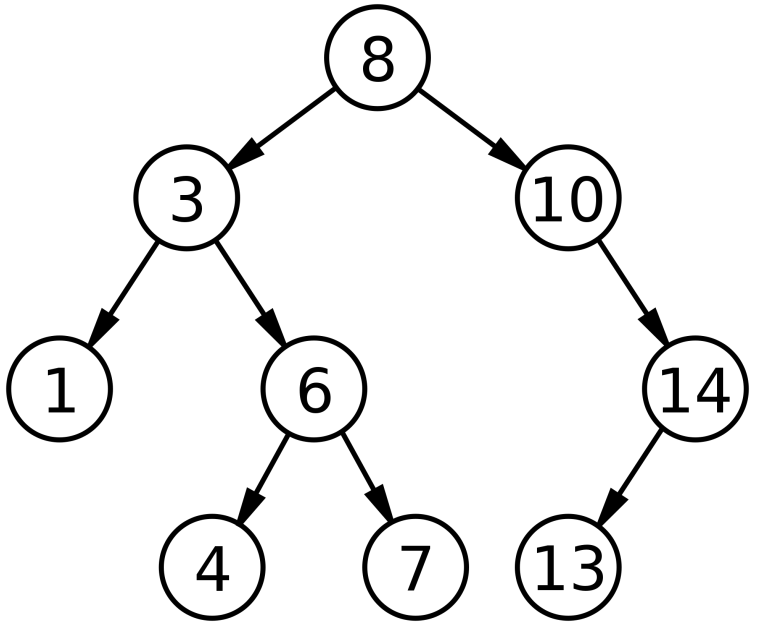
Binary Search Tree

Dictionary

1. Search → Static
2. Insert
3. Delete

Dynamic

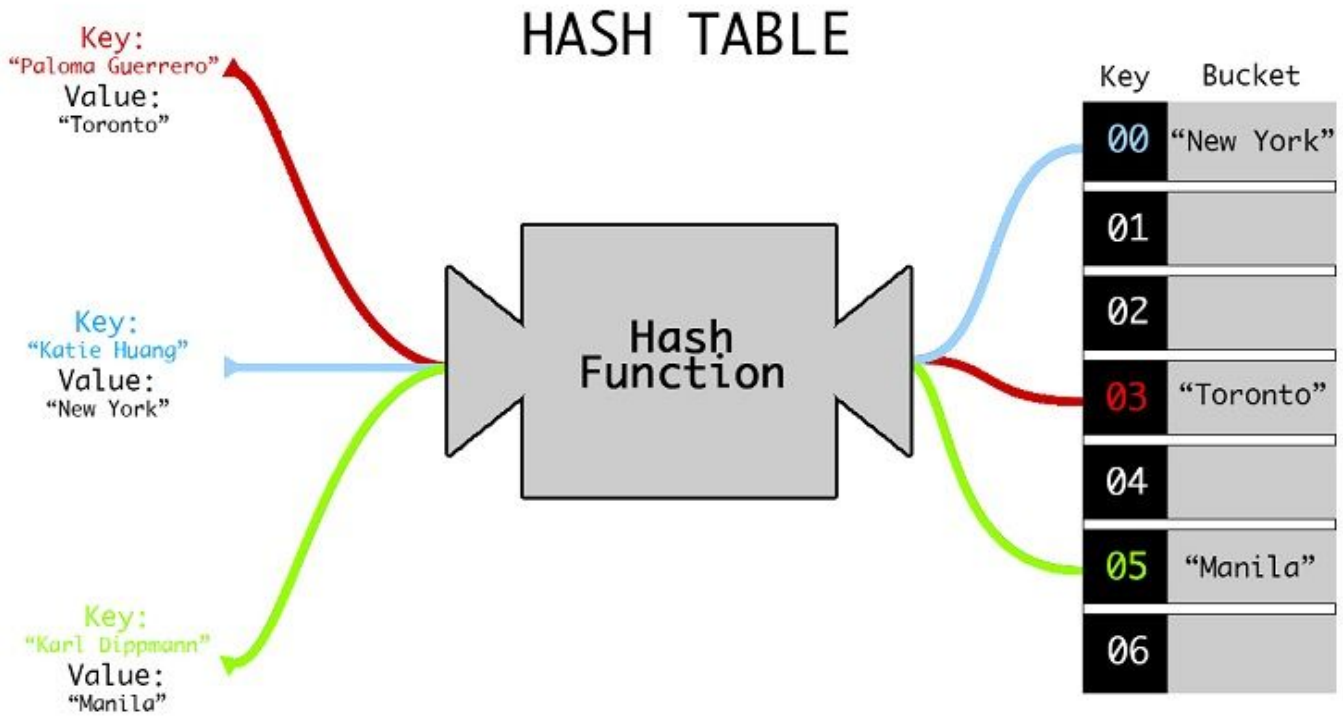
Depth



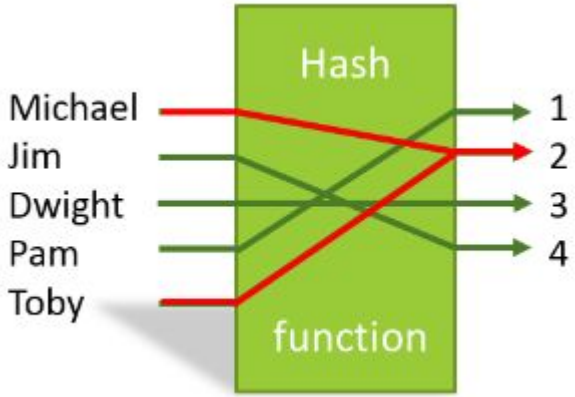
Binary Search Tree

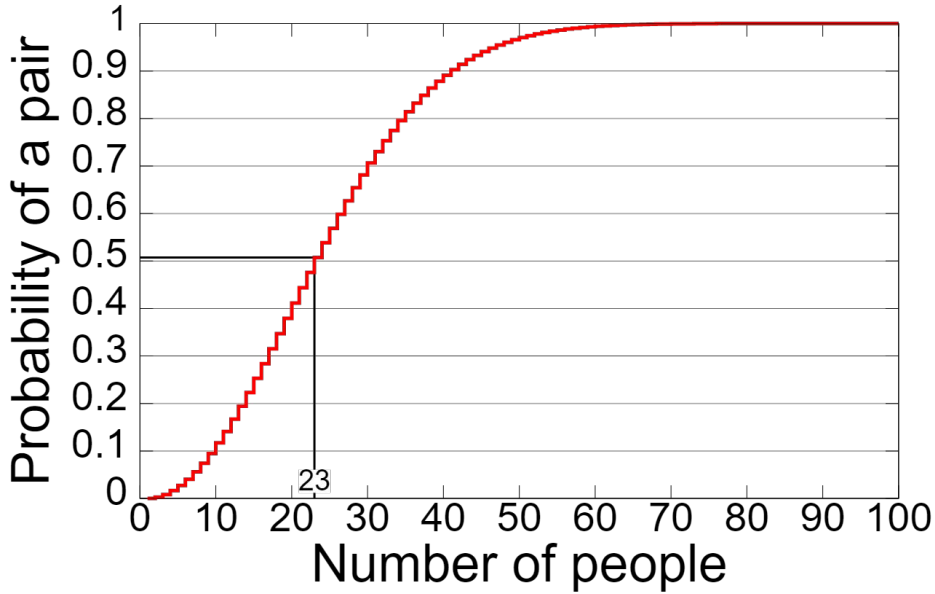
Can we do better?

Hashing


HASH TABLE
Key:
"Paloma Guerrero"
Value:
"Toronto"
Key:
"Katie Huang"
Value:
"New York"
Key:
"Karl Dippmann"
Value:
"Manila"
Hash Function
Key
Bucket
00
"New York"
01
02
03
"Toronto"
04
05
"Manila"
06

Collisions





For any hash function
there are bad key sets!

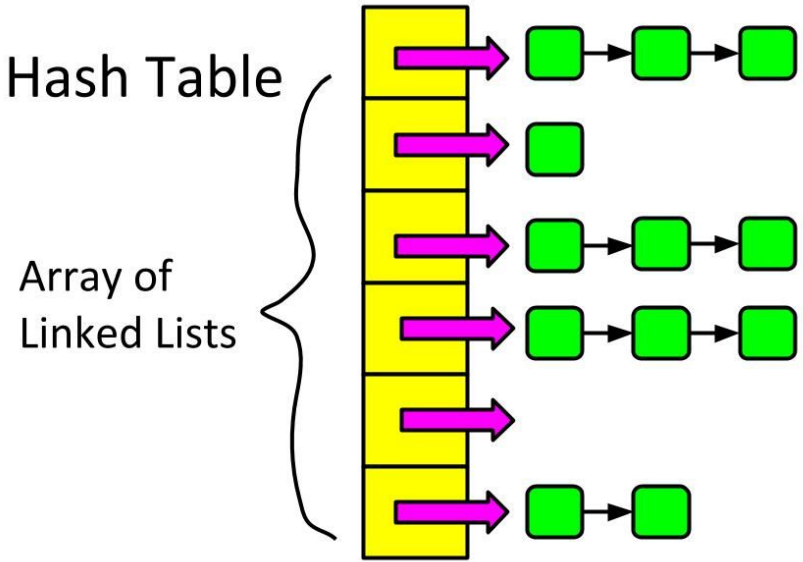
Universal Hashing

- Universal Hash Family: the hashes are distributed well regardless the key set.

- Universal Hashing: We simply choose a prime number m and uniformly at random some factors $a_0, \ldots, a_r$.

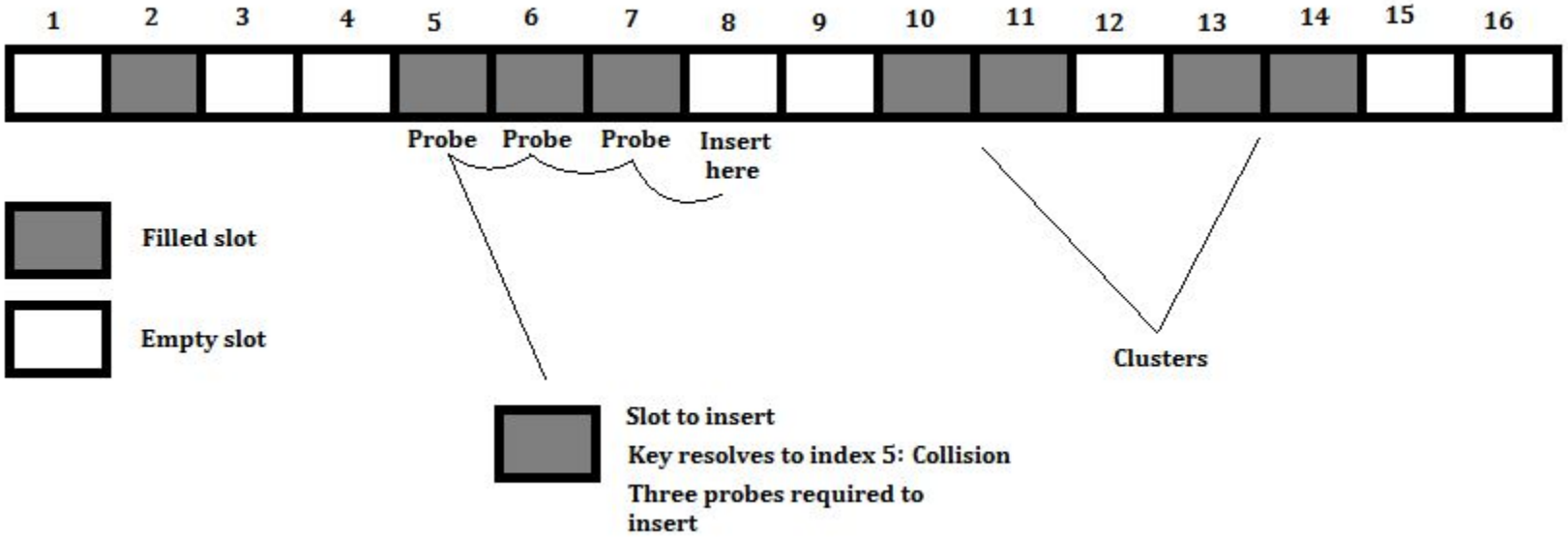
Static Hashing

- Space - Collisions trade-off
- Perfect Static Hashing



Hashing with probing

- Linear probing (primary clustering)
- Quadratic Probing (secondary clustering)
- Double Hashing

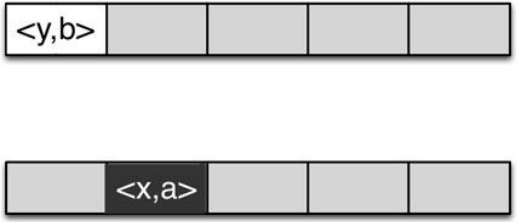


Cuckoo Hashing

Insertion when one of the two buckets is empty

Step 1: Both buckets for <x,a> are tested, the one in T2 is empty.

Step 2: <x,a> is stored in the empty bucket in T2.



Insertion when the two buckets already contain entries

Step 1: Here <y,b> will be withdrawn from T1 so that <x,a> can be stored.

Step 2: After <x,a> has been stored in T1, <y,b> needs to be moved to T2. The bucket in T2 may already contain an entry, if so this entry will need to be moved.