CS 225 Spring 2019 :: TA Lecture Notes
3/25 Hashing Analysis

By Wenjie

- ## Collision
  - ### Separate chaining

We could resolve collisions using separate chaining which is an instance of open hashing, which we use another linked list to save all collisions.

- Running time:
  - Insert: O(1)
  - Remove/find:
    - worst case: o(n)
    - SUHA: O(N/n) = load factor

(*Open hashing : data stored outside of the array

- ## Linear probing

Another way to handle collisions is to use closed hashing. When using closed hashing, we keep all our data inside of our hash table. We will cover two strategies for closed hashing:

Probing- based hashing

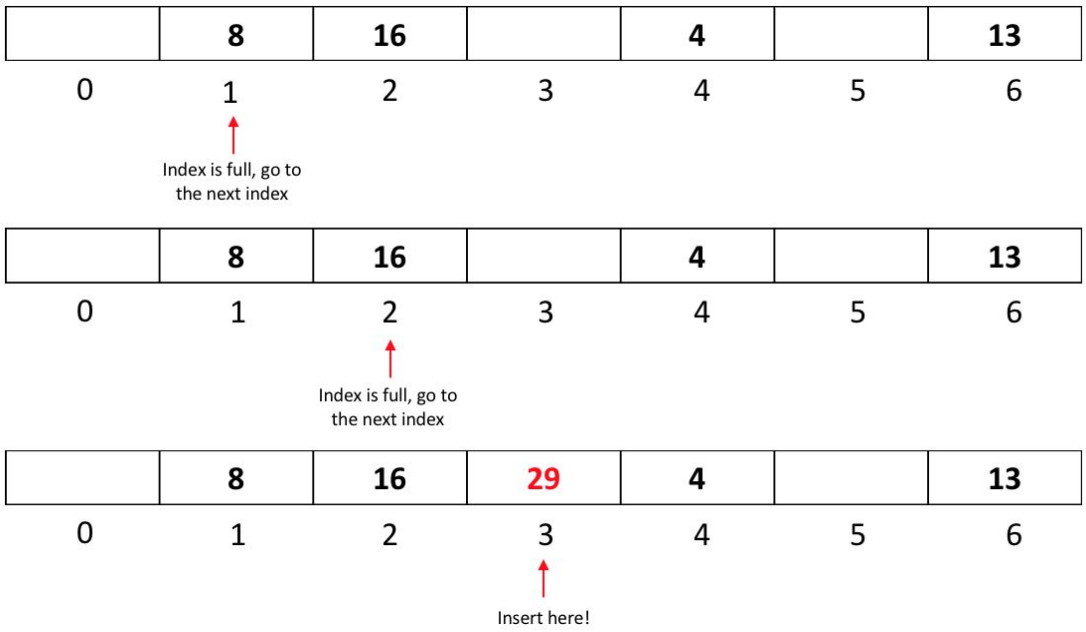
- Hash the key and try to insert. If the spot is filled, linearly loop down the array to find the next open spot to place the data. Assume we have the following hash table:

	8	16		4		13
0	1	2	3	4	5	6

Insert 29: 29 % 7 = 1

CS 225 Spring 2019 :: TA Lecture Notes
3/25 Hashing Analysis

By Wenjie



- We stop when we find an empty spot to insert our data.
  - In the worst case scenario, it will run in O(n) time.

- **Primary Clustering**
  - Primary cluster is the largest sequential block of data in a hash table.
  - It is the source of the worst case running time in linear probing.
  - It takes the largest chunk of space; therefore data points have the highest probability of landing exactly there, which makes the cluster even bigger.

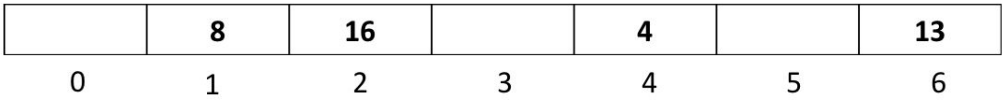
- **Double Hashing**
  - The idea behind double hashing is to add a second hash function, which is called a step function, where a good step function is:
    - Has to return values between 1 and (size - 1).
    - Needs to cover all elements of the array.
      - We need to make array size a prime or a relative prime (will not be divisible by the step size) to the range of the second hash function.

CS 225 Spring 2019 :: TA Lecture Notes 3/25 Hashing Analysis

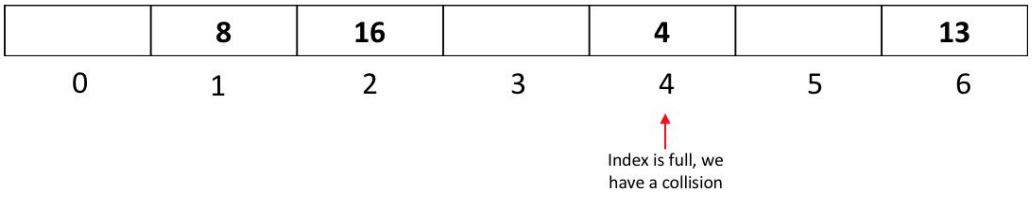
By Wenjie

- Ex: if the size was 6 and the step was 2, we would visit 0, 2, 4 in a cycle.

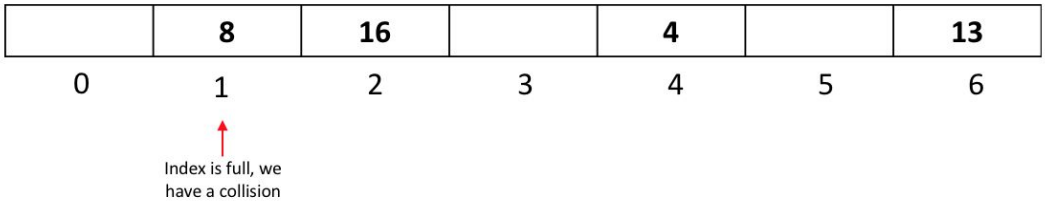
Double hashing example:



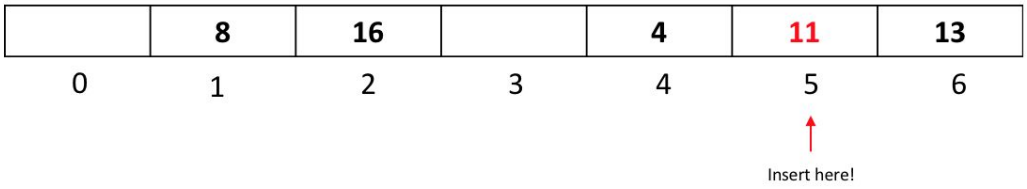
Insert 11: 11 % 7 = 4 → we have a collision.



Let step function be: 5 - (k % 5) → 5 - (11 % 5) = 5 - 1 = 4; the step is 4.
We move by 4:



Index 1 is still full, so we move another 4 steps and since index 5 is free, we add:



- **Running Time:**
  - Separate chaining
    - Successful $1 + \alpha/2$
    - Unsuccessful $1 + \alpha$

CS 225 Spring 2019 :: TA Lecture Notes 3/25 Hashing Analysis

By Wenjie

    - Linear probing:
        - Successful ½(1+1/(1-α))
        - Unsuccessful ½(1+1/(1-α))^2
    - Double hashing:
        - Successful 1/α * iN(1/(1-α))
        - Unsuccessful 1/(1-α)

- **Which collision policy is better?**
    - If data consists of large records, we should chose open hashing (ie. separate chaining) because we can copy only pointers and we do not have to copy the actual data.
    - When we do not care about structure speed and our data is easy to copy, we want to use closed hashing
    - In conclusion, hash tables are very fast and efficient for lookups. However, if we want to search for nearest neighbour (nn), hash tables are not that useful (it would take O(n) to find nn).
- **What hash table replaces?**
    - BST/AVL as underlying data structure in dictionaries
- **What constraints exist on hashtable but not for bst/avl?**
    - Hash can do exact find
- **Why we still care about bst?**
    - BST good for nearest neighbour and range finding

Summary of running time

	Insert	Find	Space
Hash	SUHA O(1) Worst O(1) for sep chaining O(n) for closed hashing	SUHA O(1) Worst O(n)	About $\frac{n}{\alpha} \approx 2n$ O(n)
AVL	lg(n)	lg(n)	O(m)
Linked list	O(1)	O(n)	O(n)