Comp 352 – Summer 2023

Hash Table

Outline

- Hash Table
  - Definition
  - Hash function
  - Collision
- Exercises

HASH TABLE(1)

The ideal hash table(hash map) data structure is merely **an array of some fixed size**, containing the keys. Besides, A hash table uses a **hash function** to compute an index into an array of buckets or slots, from which the desired value can be found.

It is often used to the situation that we need to fetch the desired key-value frequently.

Basic methods:

- get(k)
- put(k, v) -- add an entry with key k and value v
- remove(k) -- remove the key-value entry e.
- size()

Hash code and hash function

- Function to map the key to an entry in the array of bucket to store the pair (key/value). A hash function is usually specified as the composition of two functions:

  Hash code:

  $\boldsymbol{h}_1$: keys $\rightarrow$ integers

  Compression function:

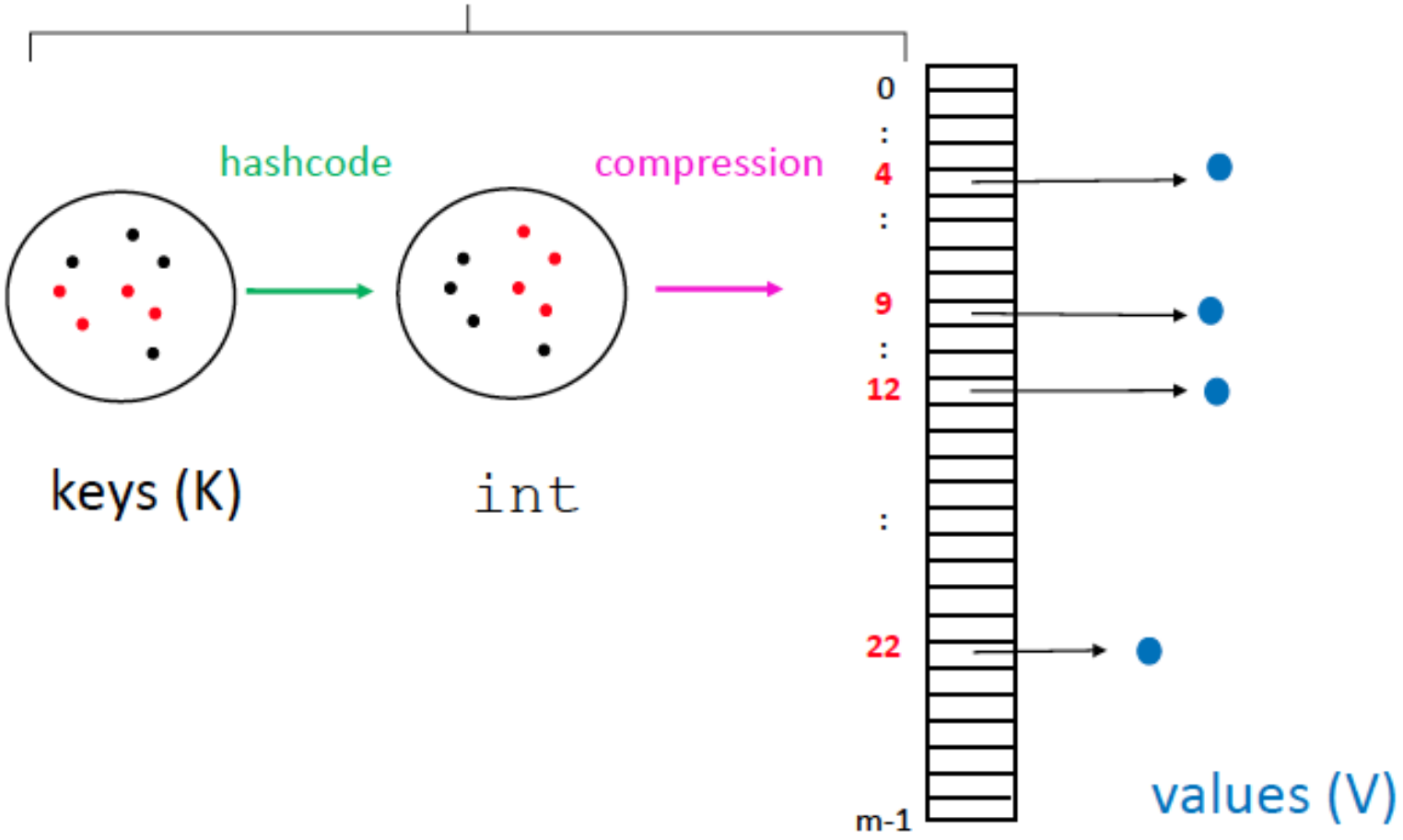
  $\boldsymbol{h}_2$: integers $\rightarrow$ [0, $\boldsymbol{N}$ – 1], N generally a prime number

An example of compression function:

h= integers mod N

"hash values"

hash function : keys → {0, …, m-1}

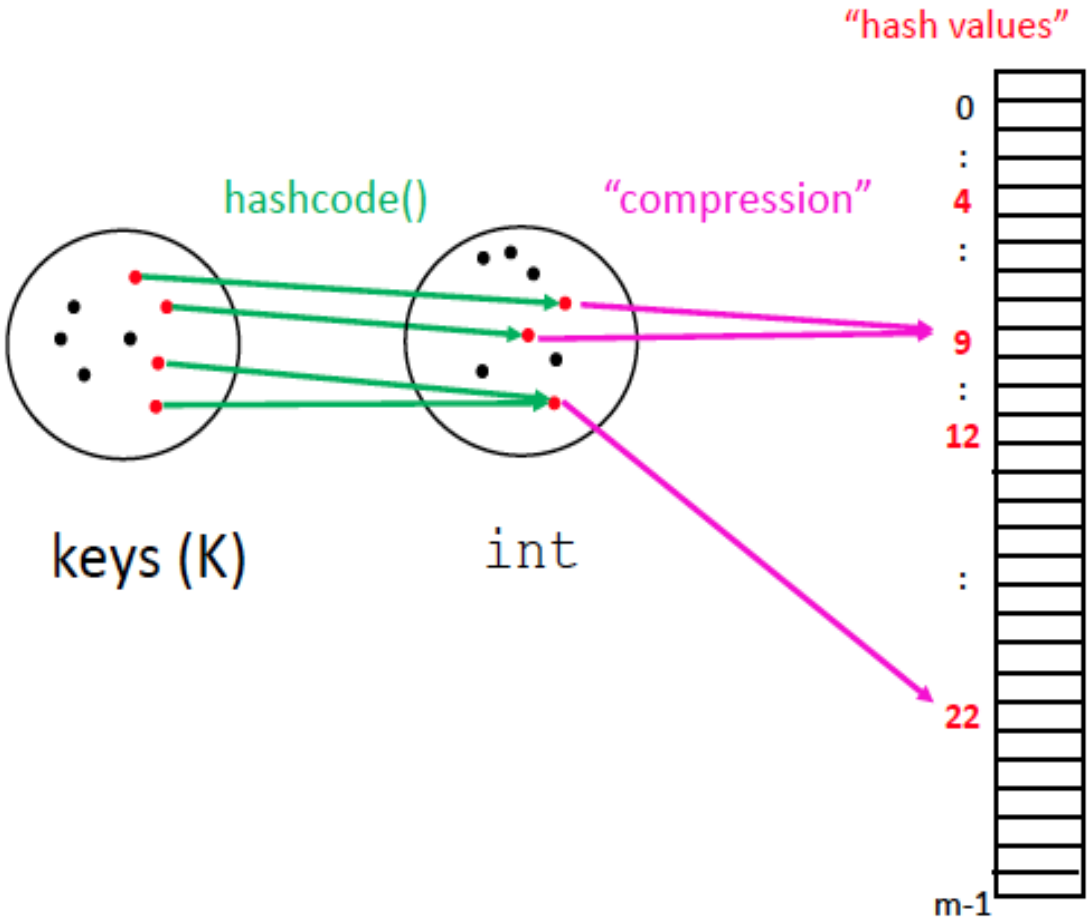


HASH FUNCTIONS -- COLLISIONS

- One issue with hash tables is how well the Hash Function behaves. That is to say, how well the keys map to integers.
- When two keys share the same hash value (result of the hash function), we get a collision.
- A good hash function minimizes collisions under most conditions.
- The way a hash table implementation handles collisions has an impact on the running time complexity of functions relying on the hash table.

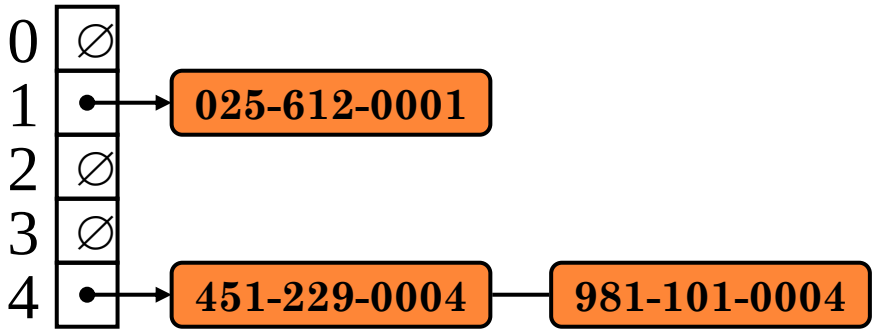
Collision

when two or more keys k map to the same hash value.



Collision handling: Open Hashing, Separate chaining

The first strategy, commonly known as either open hashing, or **separate chaining**, is to keep a list of all elements that hash to the same value.



Collision handling: Closed addressing

An alternative method to resolving collisions. The colliding item is placed in **a different entry of the table** instead of a linked list.

We have different way to deal with collisions.

- Linear Probing
- Quadratic Probing
- Double Hashing

The new hash function is:
hi(X) = (Hash(X) + F(i)) mod N

LINEAR PROBING

The idea of linear probing is to handle collisions by placing the colliding item in the **next (circularly) available table cell**.

On insert(x), compute f(x) mod N, if the cell is full, find another by sequentially searching for the next available slot

- Go to h(x)+1, h(x)+2 etc..

On find(x), compute f(x) mod N, if the cell doesn't match, look elsewhere.

In linear probing we have
hi(x) = (h(x) + i) mod N (i= 0, 1, 2, …)

if A[h(k)modN] is already occupied, then try next A[(h(k)+1)modN], A[(h(k)+2)modN], etc.

QUADRATIC PROBING

- Resolve collisions by examining certain cells (1,4,9,...) away from the original probe point
- In quadratic probing we have

hi(k) = (h(k) + i^2) mod N, i=1,2,3,.....

- **if A[h(k)modN] is already occupied, then try next A[(h(k)+1^2)modN], A[(h(k)+2^2)modN], etc.**

DOUBLE HASHING

uses a secondary hash function h'(k) and places the colliding item in the first available cell using the formula

hi(X) = (h(x) + i*Hash2(x)) mod N

if A[h(k)modN] is already occupied, then try next A[(h(k)+1*Hash2(x))mod N], A[(h(k)+2*Hash2(x))mod N], etc.

E.g. Hash2(X) = R – (X mod R)

R is a prime smaller than N

MCQ 1

What is a hash table?

a. A structure that maps values to keys
b. A structure that maps keys to values
c. A structure used to implement a stack and queue
d. Another name for binary search tree

MCQ2

What can be the technique to avoid collision?

a. Make the hash function to appear random
b. Use chaining method
c. Use uniform hashing
d. All of the mentioned

MCQ3

In a simple uniform hashing what is the search complexity?

a. O(n)
b. O(logn)
c. O(nlogn)
d. O(1)
e. None of the above

MCQ 4

Hash table question multiple choice

Consider the following hashtable:

Index	0	1	2	3	4	5	6	7	8
key			20	11	4		51	6	

It was created with the hash function H(x) = x % 9 and uses a linear collision scheme.

Which of the following was the order in which the elements were inserted to produce the table above?

MCQ 5

a. All of the insert orders listed here will produce the hashtable shown above.
b. None of the insert orders listed here will produce the hashtable shown above.
c. The insert order: 20, 4, 51, 6, 11 will produce the hashtable shown above.
d. The insert order: 20, 11, 51, 6, 4 will produce the hashtable shown above.
e. The insert order: 51, 6, 20, 11, 4 will produce the hashtable shown above.

MCQ 6

Consider the following hashtable:

Index	0	1	2	3	4	5	6	7	8	9
key	81	10	63			14	42			

It was created with the hash function H(x) = x % 9.

Which of the following statements is true about this hashtable?

MCQ 6

a. None of the statements listed here is correct
b. All of the statements listed here is correct
c. The hashtable uses a quadratic collision scheme to handle collisions
d. The hashtable uses separate chaining to handle collisions
e. The load factor of the table (that is nb_elment_in_arr / size_of_array) is 77.77%

Exercise

- Assume an 11 entry hash table
- Use the hash function
  - h(k) = 2k + 5 mod 11
- Insert the keys (k)
  - 12, 44, 13, 88, 23, 94, 11, 39, 20, 16, 5
- Draw the contents of hash table given that for collisions:
  - Chaining is used
  - Linear Probing is used
  - Double Hashing is used with h'(k) = 7 - (k mod 7)