SC1007 Tutorial 5
Hash Table

Dr Liu Siyuan

Email: sylliu@ntu.edu.sg

Office: N4-2C-72a

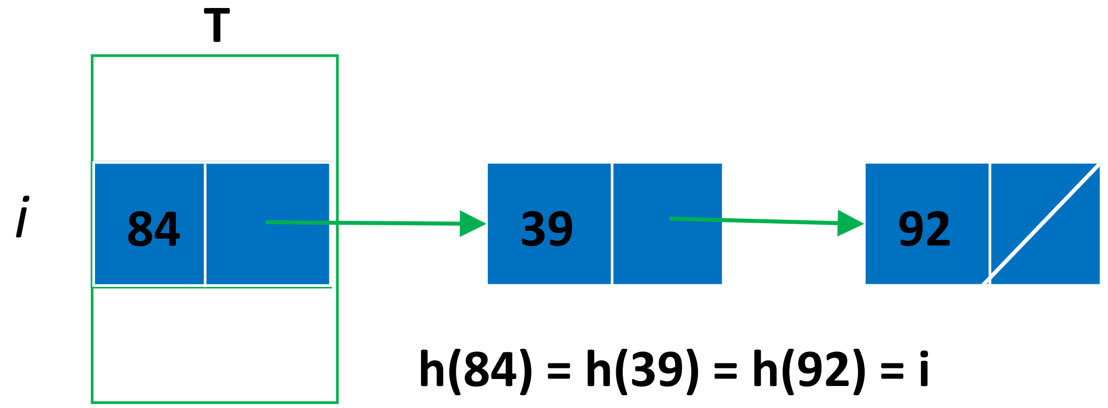
Hashing

- To reduce the key space to a reasonable size
- Hashing is the process of using a hash function to map data of arbitrary size to fixed-size values of keys

  hash function: {all possible keys} → {0, 1, 2, …, h-1}
- Each key is mapped to a unique index (**hash value/code/address**)
- Search time remains O(1) on the average
- The array is called a **hash table**
- Each entry in the hash table is called a **hash slot**
- When multiple keys are mapped to the same hash value, a **collision** occurs
- If there are n records stored in a hash table with h slots, its **load factor** is $\boldsymbol{\alpha} = \frac{n}{h}$

Closed Addressing: Separate Chaining

- When multiple keys are hashed into the same slot index, these keys are inserted into to singly-linked list, which is known as a chain

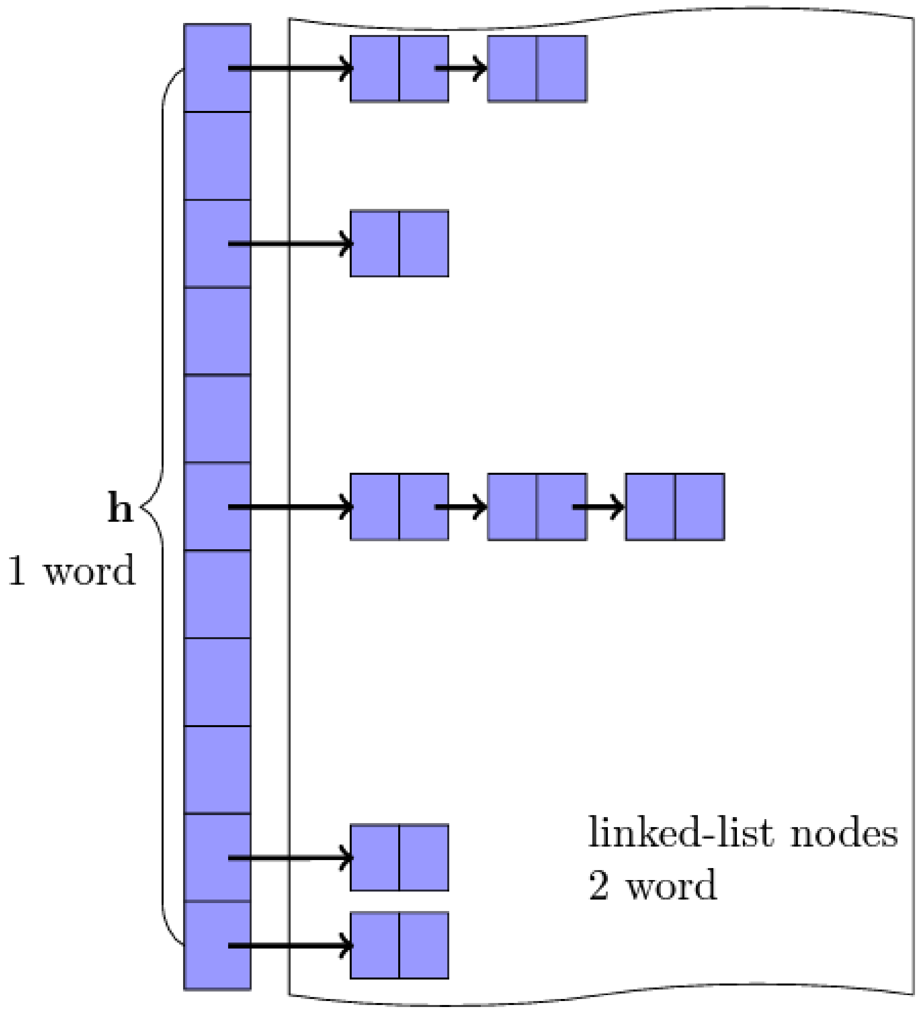


- During searching, the searched key with hash address i is compared with keys in linked list H[i] sequentially
- In closed address hashing, there will be α number of keys in each linked list on average.

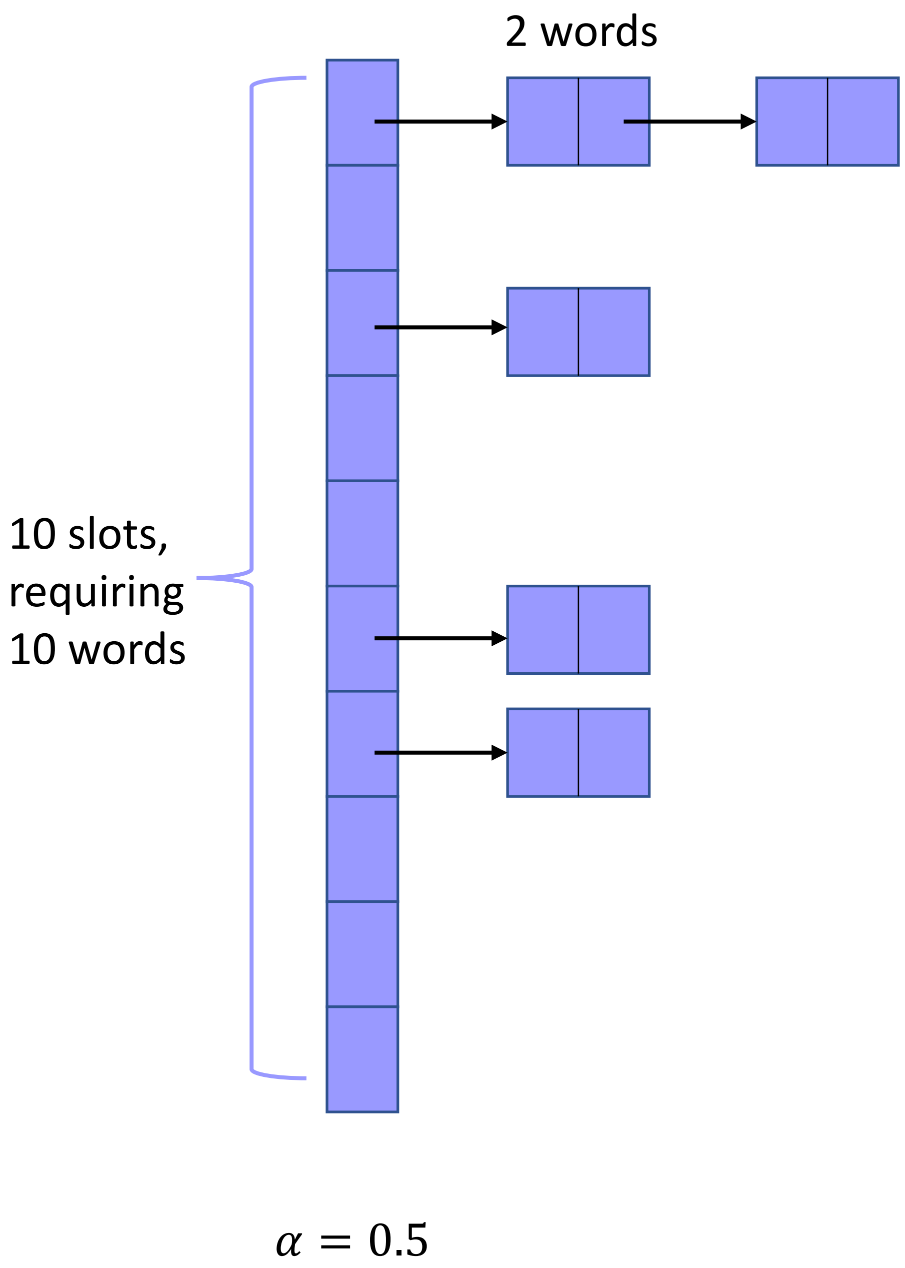
Open Addressing

- When collision occurs, probe is required for the alternate slot
  - Linear Probing
    - $H(k,i) = (H'(k)+i) \bmod h$, $i = 0, 1, 2, \ldots, h-1$
  - Quadratic Probing
    - $H(k,i) = (H'(k)+c_1 i+c_2 i^2) \bmod h$, $i = 0, 1, 2, \ldots, h-1$
  - Double Hashing
    - $H(k,i) = (H_1(k)+iH_2(k)) \bmod h$, $i = 0, 1, 2, \ldots, h-1$

Q1

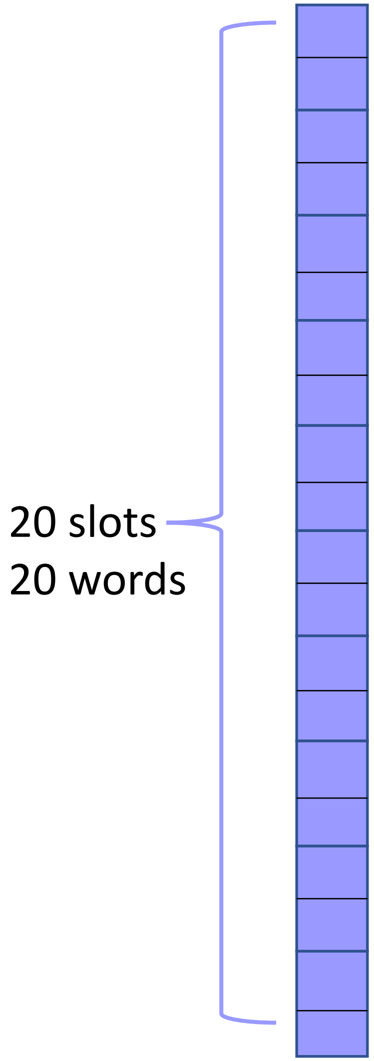


- The type of a hash table *H* under closed addressing is an array of list references, and under open addressing is an array of keys. Assume a key requires one “word” of memory and a linked list node requires two words, one for the key and one for a list reference.
- Consider each of these load factors for closed addressing: 0.5, 1.0, 2.0. Estimate the total space requirement, including space for lists, under closed addressing
- Assuming that the same amount of space is used for an open addressing hash table, what are the corresponding load factors under open addressing?



- Let *h* be hash table size. There are *h* slots.
- Load factor $\alpha = \frac{n}{h}$

1. When $\alpha = 0.5$, under closed addressing
   - *n*=0.5*h*, meaning there are 0.5*h* keys, which are 0.5*h* nodes.
   - Each node require 2 words.
   - Total space is $2\mathrm{n} + h = 2 \times 0.5h + h = 2h$.
2. When *α*=1
   - There are *h* nodes
   - Total space: $h \times 2 + h = 3h$.
3. When *α=2*
   - There are 2*h* nodes
   - Total space: $2h \times 2 + h = 5h$.



- Assuming that the same amount of space is used for an open addressing hash table, what are the corresponding load factors under open addressing?

1. When there are 0.5*h* keys, and given 2*h* space, the corresponding load factor under open addressing is $\alpha = \frac{0.5h}{2h}$=0.25
2. When there are 1*h* keys, and given 3*h* space,
$$\alpha = \frac{h}{3h} = 0.33$$
3. When there are 2*h* keys, and given 5*h* space,
$$\alpha = \frac{2h}{5h} = 0.4$$

Q2

- Consider a hash table of size *n* using open address hashing and linear probing. Suppose that the hash table has a load factor of 0.5, describe with a diagram of the hash table, the best-case and the worst-case scenarios for the key distribution in the table.
- For each of the two scenarios, compute the average-case time complexity in terms of the number of key comparisons when inserting a new key. You may assume equal probability for the new key to be hashed into each of the *n* slots. [Note: Checking if a slot is empty is not a key comparison.]

n=7

$H'(k)$ = k mod n

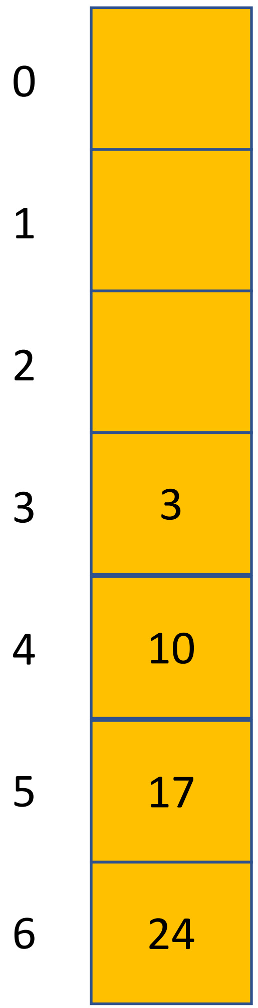
0	0
1	
2	2
3	
4	4
5	
6	6

- Linear Probing: probe the next slot when there is a collision
  - $H(k,i)$ = $(k+i)$ mod n, where $i\in[0, n\text{-}1]$, if H and H' are the same hash function
- There are n slots, α=0.5, there are $n/2$ keys.
- Best case scenario:
  - The n/2 keys are hashed and distributed evenly into the n slots
- Assume that equal probability for a key to be hashed into each of the n slots, the average-case time complexity

$$= \frac{1}{n}\left(\sum_{i=1}^{\frac{n}{2}} 0\right) + \frac{1}{n}\left(\sum_{i=1}^{\frac{n}{2}} 1\right) = \frac{1}{n} \times \frac{n}{2} = 0.5 = \Theta(1)$$

n=7

$H'(k)$ = k mod n



- Linear Probing: probe the next slot when there is a collision
  - $H(k,i)$ = $(k+i)$ mod n, where $i \in [0, n-1]$, if H and H' is the same hash function
- There are n slots, α=0.5, there are $n/2$ keys.
- Worse case scenario:
  - The n/2 keys are hashed in consecutive slots. Each key always has to rehash and visit every key in the table. The ith key is hashed and rehash i times to get the slot.
- Assume that equal probability for a key to be hashed into each of the n slots, average-time-complexity

$$= \frac{1}{n}\left(\sum_{i=1}^{\frac{n}{2}} 0\right) + \frac{1}{n}\left(\sum_{i=1}^{\frac{n}{2}} i\right) = \frac{1}{n} \times \frac{\frac{n}{2}\times\left(1+\frac{n}{2}\right)}{2}$$

$$= \frac{n}{8} + \frac{1}{4} = \Theta(n)$$

Q3

- Consider a hash function h(k)= k mod m, where m = 2^p -1 and k is a character string interpreted in radix 2^p . Show that if string x can be derived from string y by permuting its characters, then x and y hash to the same value.
- A character string interpreted in radix 2^p: Each character in the string is assigned a numeric value. The string is then interpreted as a number in base 2^p. For example, suppose p = 3, so that base 2^p = 8 is used, and the character values are assigned as follows: $\boldsymbol{A} = \mathbf{1}$, $\boldsymbol{B} = \mathbf{2}$, $\boldsymbol{C} = \mathbf{3}$
- Then, the string "CAB" is interpreted as: k = 3 × 8^2 + 1 × 8^1 + 2 × 8^0 = 202
- Similarly, the string "BCA" is interpreted as: k = 2 × 8^2 + 3 × 8^1 + 1 × 8^0 = 153
- We can see that $h(202) = 202\ mod\ (2^3 - 1) = 6$, $h(153) = 153\ mod\ (2^3 - 1) = 6$

Q3

- For a string $S=C_1C_2C_3 \ldots C_n$, where each C_i is a character with a numerical value a_i, its numerical representation in radix 2^p is:
  - $k = a_1 \times (2^p)^{n-1} + a_2 \times (2^p)^{n-2} + \cdots + a_n \times (2^p)^0$
- Let x and y be two strings that are permutations of each other. This means they contain the same characters, just arranged differently.
- The numerical representations of x and y are:
  - $k_x = a_1 \times (2^p)^{n-1} + a_2 \times (2^p)^{n-2} + \cdots + a_n \times (2^p)^0$
  - $k_y = b_1 \times (2^p)^{n-1} + b_2 \times (2^p)^{n-2} + \cdots + b_n \times (2^p)^0$

Q3

- $(2^p)^i \ mod \ (2^p - 1) = 1$, for any integer i.
- Applying Modulo $2^p - 1$
  - $k_x \ mod \ (2^p - 1)$ = $a_1 + a_2 + \cdots a_n$
  - $k_y mod \ (2^p - 1) = \ b_1 + b_2 + \cdots b_n$
- As $a_1 + a_2 + \cdots a_n = b_1 + b_2 + \cdots b_n$
- $k_x \ mod \ (2^p - 1)$ = $k_y \ mod \ (2^p - 1)$
- $h(k_x) = h(k_y)$

Q3

- This property can lead to hash collisions in applications where order matters.
- Password Hashing “1234” and “4321” should produce different hashes, but under this scheme, they do not.
- Database Indexing “ABCD” and “BCDA” mapping to the same hash could cause incorrect lookups.
- Cryptographic Signatures: Digital signatures should change if even a small character swap happens.