Hashing (cont.)

Instructor: Dr. Sunho Lim (Ph.D., Assistant Professor)

Lecture 20

sunho.lim@ttu.edu

Adapted partially from Data Structures and Algorithms in C++, Adam Drozdek, 4th Edition, Cengage Learning; and Algorithms and Data Structures, Douglas Wilhelm Harder, Mmath

Collisions

- Map two different keys to the same location
  - cannot store two records in the same location
  - solve the problem of collision → **collision resolution**
  - cannot guarantee to eliminate collisions
- Two most popular methods,
  - **open addressing**
  - **chaining**

Collisions (cont.): Open Addressing

- Open addressing (or closed hashing)
  - upon collision, compute new positions
- Two types of values in hash table
  - sentinel values (e.g., -1 → null): no data value in the location
  - data values
- **Probing**
  - process of examining memory locations in the hash table
  - linear probing, quadratic probing, double hashing, and rehashing

Open Addressing (cont.): Linear Probing

- **h(k, i) = [h'(k) + i] mod m**, where
  - m: size of the hash table
  - h'(k) = (k mod m)
  - i: the probe number varies from 0 to m – 1
- When inserting a key,
  - probe the location generated by h'(k) = k mod m
    - if free, store the value
  - if occupied, subsequently probe the locations generated by,
    - [h'(k) + 1] mod m, [h'(k) + 2] mod m, and so on
- For example, consider a hash table of size =10. Using linear probing, insert the keys 72, 27, 36, 24, 63, 81, 92, and 101 into the table.

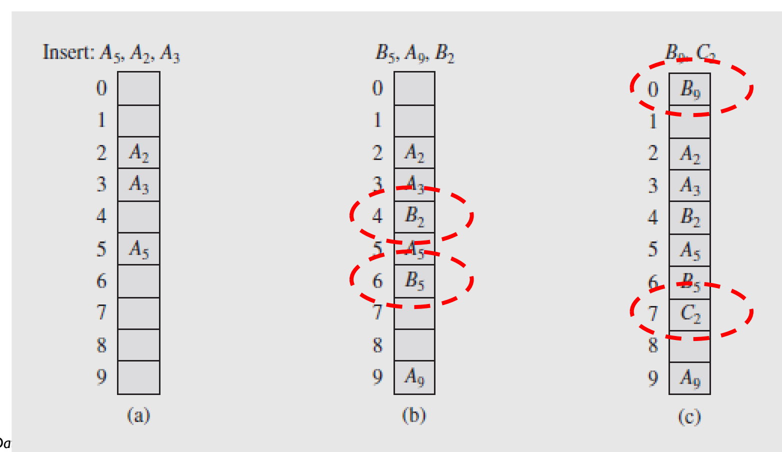
Open Addressing (cont.): Linear Probing (cont.)

- For example, consider a hash table of size =10. Using linear probing, insert the keys 72, 27, 36, 24, 63, 81, 92, and 101 into the table. (cont.)

0	1	2	3	4	5	6	7	8	9
–1	81	72	63	24	92	36	27	101	–1

Open Addressing (cont.): Linear Probing (cont.)

- Another example,



Open Addressing (cont.): Linear Probing (cont.)

- **Searching** a value using linear probing
  - **re-compute** the array index
  - **compare** the key stored at the location with the value to be searched
  - **same as for storing a value in a hash table**
  - If match?
    - search time = O(1)
  - If not,
    - begin a sequential search
- Search function
  - found the value
  - encounter a vacant location → indicating the value is not present
  - reach the end of the table → indicating the value is not present

Open Addressing (cont.): Quadratic Probing

- **h(k, i) = [h'(k) + c1 x i + c2 x i^2] mod m**, where
  - m: size of the hash table
  - h'(k) = (k mod m)
  - i: the probe number varies from 0 to m – 1
  - c1, c2: constants, such as c1 and c2 != 0
- When inserting a key,
  - probe the location generated by h'(k) = k mod m
    - if free, store the value
  - if occupied, subsequently probe the locations generated by,
    - h(k, 1), h(k, 2), h(k, 3), and so on

- For example, consider a hash table of size =10. Using quadratic probing, insert the keys 72, 27, 36, 24, 63, 81, and 101 into the table. Here c1 = 1 and c2 = 3.

Open Addressing (cont.): Quadratic Probing (cont.)

- For example, consider a hash table of size =10. Using quadratic probing, insert the keys 72, 27, 36, 24, 63, 81, and 101 into the table. Here c1 = 1 and c2 = 3. (cont.)

0	1	2	3	4	5	6	7	8	9
–1	81	72	63	24	101	36	27	–1	–1

Open Addressing (cont.): Quadratic Probing (cont.)

- Another example,

	Insert: A_5, A_2, A_3	B_5, A_9, B_2	B_9, C_2
0			B_9
1		B_2	B_2
2	A_2	A_2	A_2
3	A_3	A_3	A_3
4			
5	A_5	A_5	A_5
6		B_5	B_5
7			
8			C_2
9		A_9	A_9
	(a)	(b)	(c)

Open Addressing (cont.): Double Hashing

- **h(k, i) = [h1(k) + i x h2(k)] mod m**, where
  - m: size of the hash table
  - h1(k) and h2(k): two hash functions, such as
    - h1(k) = k mod m
    - h2(k) = k mod m', where m' < m, such as m' = m -1 or m - 2
- When inserting a key,
  - probe the location generated by h1(k) = k mod m
    - if free, store the value
  - if occupied, subsequently probe the locations generated by,
    - h(k, 1), h(k, 2), h(k, 3), and so on

- For example, consider a hash table of size =10. Using double hashing, insert the keys 72, 27, 36, 24, 63, 81, and 92 into the table. Take h1 = k mod 10 and h2 = k mod 8.

Open Addressing (cont.): Double Hashing

- For example, consider a hash table of size =10. Using double hashing, insert the keys 72, 27, 36, 24, 63, 81, and 92 into the table. Take h1 = k mod 10 and h2 = k mod 8.

0	1	2	3	4	5	6	7	8	9
92	81	72	63	24	–1	36	27	–1	–1

Open Addressing (cont.): Rehashing

- What if the hash table becomes **nearly full**?
  - increase the number of collisions
  - degrading performance of insertion and search operations
- Create a **new hash table** with size double of the original hash table
  - move all the entries
    - compute a new hash value for each entry
  - Insert each entry to the new hash table

Open Addressing (cont.): Rehashing (cont.)

- For example, the original hash table,

0	1	2	3	4
	26	31	43	17

  - new hash table, double the size of the original hash table

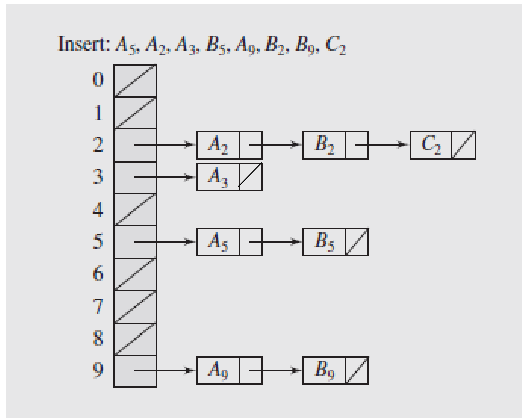
0	1	2	3	4	5	6	7	8	9

  - rehash the key values from the old hash table into the new one
    - h(x) = x % 10

0	1	2	3	4	5	6	7	8	9
	31		43			26	17		

Collisions (cont.): Chaining

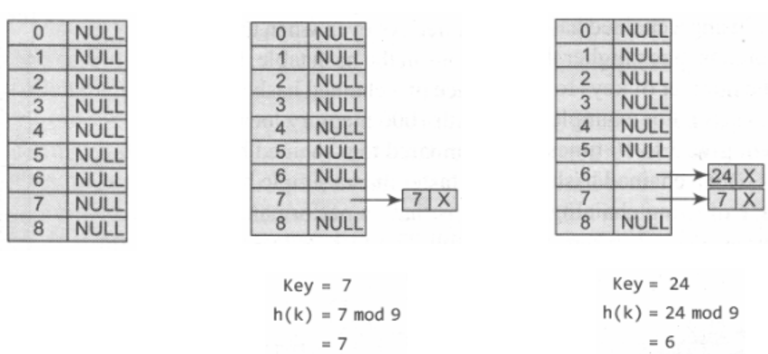
- Store a **pointer** in a hash table
- Searching
  - scanning a linked list for an entry with the given key
- Simplicity
  - inserting, deleting, and searching a key
  - inserting a key, O(1)
  - deleting and searching a value, O(m), where m is he number of elements in the list of that location
  - worst case, O(n), where n is the number of key values stored in the chained hash table



CS

Collisions (cont.): Chaining (cont.)

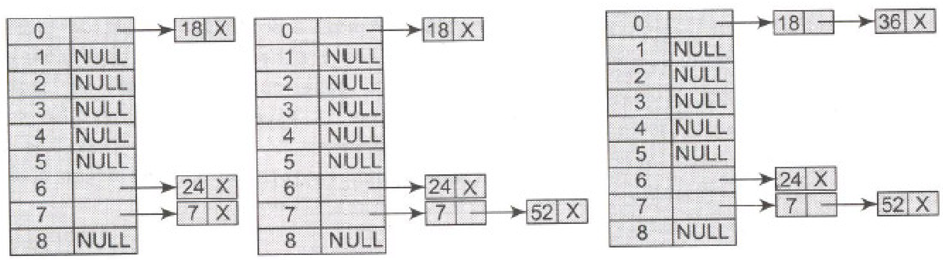
- For example, insert the keys 7, 24, 18, 52, 36, 54, 11, and 23 in a chained hash table of 9 memory locations. Use h(k) = k mod m. In this case, m = 9. Initially, the has table can be given as:



CS2413:

Collisions (cont.): Chaining (cont.)

- For example, insert the keys 7, 24, 18, 52, 36, 54, 11, and 23 in a chained hash table of 9 memory locations. Use h(k) = k mod m. In this case, m = 9. Initially, the has table can be given as: (cont.)



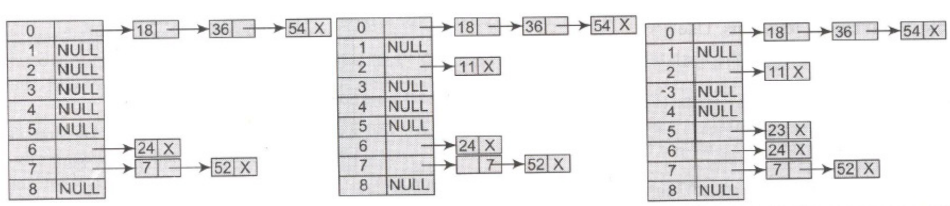
Key = 18
h(k) = 18 mod 9 = 0

Key = 52
h(k) = 52 mod 9 = 7

Key = 36
h(k) = 36 mod 9 = 0

Collisions (cont.): Chaining (cont.)

- For example, insert the keys 7, 24, 18, 52, 36, 54, 11, and 23 in a chained hash table of 9 memory locations. Use h(k) = k mod m. In this case, m = 9. Initially, the has table can be given as: (cont.)



Key = 54
h(k) = 54 mod 9 = 0

Key = 11
h(k) = 11 mod 9 = 2

Key = 23
h(k) = 23 mod 9 = 5