Data Structure

Lab Session #11: Searching 2

U Kang
Seoul National University

Goals

- Implement **both Open and Closed Hash Table**
  - Fill in the following methods in HashTable.java.
    - public void hashInsert(Key k, E r);
    - public E hashSearch(Key k);
    - public E hashRemove(Key k);
    - public void changeToClosed();
    - public void hashPrint();
  - Use the implemented interface and classes.
    - Dictionary, HashDictionary, and KVpair.
- Print the sample output corresponding to the sample input.

Notice

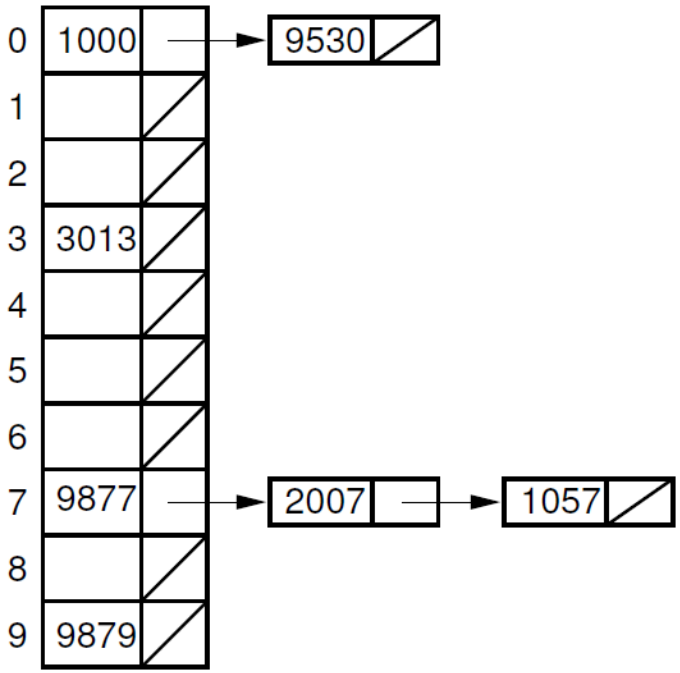
- After implementing “HashTable”, check if your program works well.
  - Check sample input and output files from the ‘testcase’ folder.
  - Test your program by using it.
- When you finish implementing the program, you can leave.
  - But, you need to stay for at least an hour.
- Check your attendance.

Build a project

- Download the project for this lab from eTL.
- Extract the project, and open it in IntelliJ.
  - See the slide of 1st lab session to check how to open the project in IntelliJ.

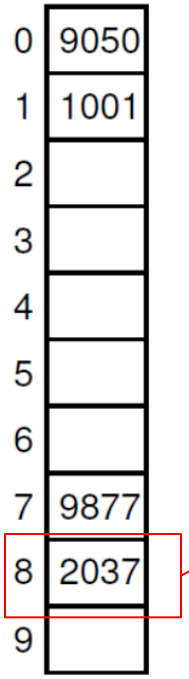
Open Hashing

- Review Lectures 21 & 22
- Open hashing (also called ‘separate chaining’)
  - Collisions are stored outside the table
  - Limitation: some slots in the table may not be used



Closed Hashing

- Review Lecture 21 & 22
- Closed hashing with linear probing
  - Collisions are stored inside the table
  - Linear probing: insert a colliding value into the next available slot in the table



Insertion order:
1001, 9050, 9877, 2037

"Linearly probed"

I/O Specification (1)

- public void hashInsert(Key k, E r);

Input form	Output form
`insert (key) (value)`	-

Description
• **Inserts a record of ((key), (value)).** • **(key) is an integer and (value) is a string not containing spaces.** • **Assumes there is no insertion of records with duplicate keys.**

Sample Input	Sample Output
`insert 1 apple1`	-

I/O Specification (2)

- public E hashSearch(Key k);

Input form	Output form
`find (key)`	`FIND: ((key), (value))`

Description
• **Finds a record with (key).** • **Does not change the table.** • **Assumes there is no query for a non-existing key.**

Sample Input	Sample Output
`find 1`	`FIND: (1, apple1)`

I/O Specification (3)

- public E hashRemove(Key k);

Input form	Output form
`remove (key)`	`REMOVE: ((key), (value))`

Description
• **Removes a record with (key).** • **Assumes there is no query for a non-existing key.**

Sample Input	Sample Output
`remove 1`	`REMOVED: (1, apple1)`

I/O Specification (3)

- public void changeToClosed();

Input form	Output form
change	-

Description
• Changes open hashing formatted table to closed hashing table. • * Check sample output (page 11) to see how it should be changed. • Hint: use hashRemove()

Sample Input	Sample Output
change	-

I/O Specification (3)

- public void print();

Input form	Output form
`print`	`PRINT HASH: ((key), (value))…`

Description
• **Prints elements in hash table with keys and values in order** • **It should be able to print both open and closed hashing table**

Sample Input	Sample Output

`print`	`PRINT HASH:`
	`(1, apple1) (111, apple3)`
	`(2, banana1) (22, banana2)`
	
	`PRINT HASH:`
	`(1, apple1)`
	`(2, banana1)`
	`(111, apple3)`
	`(22, banana2)`

Sample Input & Sample Output

<Sample Input>

- insert 1 apple1
- insert 11 apple2
- insert 111 apple3
- insert 2 banana1
- insert 22 banana2
- print
- find 11
- find 2
- remove 11
- remove 2
- find 111
- find 22
- print
- change
- print
- quit

<Sample Output>

- PRINT HASH:
  (1, apple1), (11, apple2) (111, apple3)
  (2, banana1) (22, banana2)
- FIND: (11, apple2)
- FIND: (2, banana1)
- REMOVE: (11, apple2)
- REMOVE: (2, banana1)
- FIND: (111, apple3)
- FIND: (22, banana2)
- PRINT HASH:
  (1, apple1) (111, apple3)
  (2, banana1) (22, banana2)
- PRINT HASH:
  (1, apple1)
  (2, banana1)
  (111, apple3)
  (22, banana2)

Questions?