CSCI 241

Scott Wehrwein

Hash Tables:
Motivation, Direct-Address tables

Goals

Understand the motivation for hash tables, and the workings of its simplistic cousin, the Direct Address Table.

Reminder: The **Set** ADT

A **Set** maintains a collection of **unique** things.

Java has this ADT built in as an interface:

```
java.util.Set<T>
```

Some methods from `java.util.Set`:

- `boolean add(T ob)`
- `boolean contains(T ob)`
- `boolean remove(T ob)`

Hash Tables: Motivation

Consider implementations of the Set ADT:

	`add`	`contains`	`remove`
Unsorted Array or Linked List	O(n)	O(n)	O(n)
Sorted Linked List	O(n)	O(n)	O(n)
Sorted Array	O(n)	O(log n)	O(n)
AVL Tree	O(log n)	O(log n)	O(log n)
Magical Array	**O(1)***	**O(1)***	**O(1)***

How would you implement a Set that can only contain the digits 0..10?

Remember Radix Sort?

[07, 19, 61, 11, 14, 54, 01, 08]

0	1	2	3	4	5	6	7	8	9

Bukkits on 1's place

insert(4)

boolean[] A:

0	F
1	F
2	F
3	F
4	F
5	F
6	F
7	F
8	F
9	F

insert(4)

boolean[] A:

0	F
1	F
2	F
3	F
4	**T**
5	F
6	F
7	F
8	F
9	F

insert(4)

insert(7)

boolean[] A:

0	F
1	F
2	F
3	F
4	**T**
5	F
6	F
7	F
8	F
9	F

insert(4)

insert(7)

boolean[] A:

0	F
1	F
2	F
3	F
4	**T**
5	F
6	F
7	**T**
8	F
9	F

insert(4)

insert(7)

insert(4)

boolean[] A:

0	F
1	F
2	F
3	F
4	**T**
5	F
6	F
7	**T**
8	F
9	F

Direct Address Table

Stores a Set of a fixed domain of values.

```
public class DigitSet {
  boolean[] A[10];

  /** pre: 0 <= i < 10 */
  void insert(int i) {

  }
  /** pre: 0 <= i < 10 */
  void contains(int i) {

  }

  void remove(int i) {

  }
```

0	F
1	F
2	F
3	F
4	**T**
5	F
6	F
7	**T**
8	F
9	F

Direct Address Table

Stores a Set of a fixed domain of values.

```
public class DigitSet {
  boolean[] A[10];

  /** pre: 0 <= i < 10 */
  void insert(int i) {
    A[i] = true;
  }
  /** pre: 0 <= i < 10 */
  void contains(int i) {

  }

  void remove(int i) {

  }
```

0	F
1	F
2	F
3	F
4	**T**
5	F
6	F
7	**T**
8	F
9	F

Direct Address Table

Stores a Set of a fixed domain of values.

```
public class DigitSet {
  boolean[] A[10];

  /** pre: 0 <= i < 10 */
  void insert(int i) {
    A[i] = true;
  }
  /** pre: 0 <= i < 10 */
  void contains(int i) {
    return A[i];
  }

  void remove(int i) {


  }
```

0	F
1	F
2	F
3	F
4	**T**
5	F
6	F
7	**T**
8	F
9	F

Direct Address Table

Stores a Set of a fixed domain of values.

```
public class DigitSet {
  boolean[] A[10];

  /** pre: 0 <= i < 10 */
  void insert(int i) {
    A[i] = true;
  }
  /** pre: 0 <= i < 10 */
  void contains(int i) {
    return A[i];
  }

  void remove(int i) {
    A[i] = false;
  }
```

0	F
1	F
2	F
3	F
4	**T**
5	F
6	F
7	**T**
8	F
9	F