Binary search and Map ADT

2/19/16
Announcements

• Monday: Read Chapter 21 (Hashing)

• HW due Wednesday night

• Quiz today: Design problems
New and improved Set ADT

• Represents unordered set of values
  {“hello”, “there”, “=”), “CS 142”}
• **No duplicate elements**

• Supports:
  – boolean add(value)  //returns whether it was new
  – boolean contains(value)
  – Iterator iterator()

Maybe also: size, remove, clear
Array-based implementation of Set

- Store the values in arbitrary order in an array, keeping the size in another variable (Basically an ArrayList)
Array-based implementation of Set

• Store the values in arbitrary order in an array, keeping the size in another variable
  (Basically an ArrayList)

Fill in the following: With this implementation, add runs in ____ while contains runs in ____

A. O(1), O(1)  D. O(n), O(n)
B. O(1), O(n)  E. None of the above
C. O(n), O(1)
Array-based implementation of Set

- Store the values in arbitrary order in an array, keeping the size in another variable
  (Basically an ArrayList)

Fill in the following: With this implementation, add runs in ____ while contains runs in ____

A. O(1), O(1)  
B. O(1), O(n)  
C. O(n), O(1)  
D. O(n), O(n)  
E. None of the above
Implementing contains

boolean contains(int key) {
    //returns whether value is in the set
    for(int i=0; i<size; i++)
        if(vals[i] == key)
            return true;
    return false;
}
Implementing contains

boolean contains(E key) {
    //returns whether value is in the set
    for(int i=0; i<size; i++)
        if(vals[i].equals(key))
            return true;
    return false;
}

Aside: Implementing `.equals`

```java
public class Pair { //represents a pair of ints
    private int x; private int y;

    ...

    public boolean equals(Object o) {
        if(o instanceof Pair) {
            Pair p = (Pair) o;
            return (x == p.x) && (y == p.y);
        }
        return false;
    }
}
```
What if the list is kept sorted?

How can you search faster?
Binary search

```java
int rank(E key) { //returns where key goes in keys
    int lo = 0; //lowest index that could have key
    int hi = size-1; //highest index that could have key
    while(lo <= hi) {
        int mid = lo + (hi - lo) / 2;
        int cmp = key.compareTo(keys[mid]);
        if(cmp < 0) hi = mid - 1;
        else if(cmp > 0) lo = mid + 1;
        else return mid;
    }
    return lo;
}
```
int rank(E key) { //returns where key goes in keys
    int lo = 0; //lowest index that could have key
    int hi = size-1; //highest index that could have key
    while(lo <= hi) {
        int mid = lo + (hi – lo) / 2;
        int cmp = key.compareTo(keys[mid]);
        if(cmp < 0) hi = mid – 1;
        else if(cmp > 0) lo = mid + 1;
        else return mid;
    }
    return lo;
}
Binary search

int rank(E key) {  //returns where key goes in keys
    int lo = 0;  //lowest index that could have key
    int hi = size-1;  //highest index that could have key
    while(lo <= hi) {
        int mid = lo + (hi - lo) / 2;
        int cmp = key.compareTo(keys[mid]);
        if(cmp < 0)   hi = mid - 1;
        else if(cmp > 0) lo = mid + 1;
        else return mid;
    }
    return lo;
}
How long does it take to add a new key into the sorted array?

A. O(1)
B. O(1) amortized
C. O(log n)
D. O(n)
E. None of the above
How long does it take to add a new key into the sorted array?

A. $O(1)$
B. $O(1)$ amortized
C. $O(\log n)$
D. $O(n)$
E. None of the above
Fill in the following: With a sorted linked list implementation of a set, add runs in ____ while contains runs in ____

A. O(1), O(1)
B. O(1), O(log n)
C. O(log n), O(1)
D. O(n), O(log n)
E. None of the above
Fill in the following: With a sorted linked list implementation of a set, add runs in _____ while contains runs in _____

A. O(1), O(1)
B. O(1), O(log n)
C. O(log n), O(1)
D. O(n), O(log n)
E. None of the above: O(n), O(n)
Dictionary/Map ADT

- Map\(<K,V>\)  //key type K, value type V
  - V put(K,V)
  - V remove(K)
  - V get(K)
  - boolean contains(K)
  - Set\(<K>\) keySet()
  - Collection\(<V>\) values()
  - boolean isEmpty()
  - int size()
  - void clear()