Common Data Structures

9/13/2017

Lists and Sets and Maps, Oh My!

- A Tour of java.util
  - Lists: ArrayList, LinkedList, Stacks, Queues
  - Sets: HashSet, TreeSet
  - Maps: HashMap, TreeMap
- Solve it together
  - Un-bear-able Zoo
  - Adding Words
- Google Competition Review
- Your Turn
  - Synchronizing Lists

Using structures with custom types.

For objects to be sorted properly, you must implement either the comparable interface or provide a comparator for that class when sorting.

For objects to work in hashing based maps and sets properly, you must override the equals and hashCode methods in Object.

Lists (and List - Like Things)

Lists are good at storing **ordered** information.

Arrays are nice and efficient for many tasks, but they can be a pain to work with in many situations.

Common operations:

add, remove, indexOf, get, size

Can be sorted using Collections.sort()

Implementing the Comparable Interface

```
class Point implements Comparable<Point> {
    int x;
    int y;

    public int compareTo(Point other) {
        if (this.x != other.x)
            return this.x - other.x;
        return this.y - other.y;
    }
}
```

Building a Comparator

```
class PointComparator implements Comparator<Point> {
    public int compare(Point p1, Point p2) {
        if (p1.y != p2.y)
            return p1.y - p2.y
        return p1.x - p2.x
    }
}
// Sort by x’s, then y’s
Collections.sort(pointList) // will sort by x’s, then y’s.
// Sort by y’s then x’s
Collections.sort(pointList, new PointComparator())
```

ArrayList

add

remove

indexOf

get

size

ArrayList

add	O(1)
remove	O(N)
indexOf	O(N)
get	O(1)
size	O(1)

LinkedList

add

remove

indexOf

get

size

LinkedList

add	O(1)
remove	O(N)
indexOf	O(N)
get	O(N)
size	O(1)

Stack

peek

pop

push

empty

Stack

peek - O(1)

pop - O(1)

push - O(1)

empty - O(1)

Queue

add

peek

poll

Queue

add - O(1)

peek - O(1)

poll - O(1)

Which Should I Use - Stack or Queue?

I am the chef at a restaurant and want to know what dish to cook next.

I want to interpret postfix notation.

1 2 + 5 * 4 3 - * Means (1 + 2) * 5 * (4 - 3) in postfix notation.

I am binging Wikipedia and want to keep track of which pages to visit.

Sets

Order may not matter.

Whether an element is in the set is what matters.

Duplicates are ignored.

Common operations:

- add, remove, contains, size

Creating a hashable object

```
class Point {
    public boolean equals(Object other) {
        if ( !(other instanceof Point) )
            return false;
        Point p = (Point) other;
        return p.x == this.x && p.y == this.y;
    }

    public int hashCode() {
        return p.x * 101 + p.y;
    }
}
```

HashSet

add - O(1)

remove - O(1)

contains - O(1)

size - O(1)

TreeSet

Works on sortable elements. Iterating over a TreeSet will iterate over the values of the set in sorted order.

add - O(log(N))

remove - O(log(N))

contains - O(log(N))

size - O(1)

Maps

Maps are similar to sets, but instead of just checking whether each key exists in the map, we associate it with a particular value.

Common operations:

put, remove, get, containsKey, size

HashMap

put

get

containsKey

remove

size

HashMap

put - O(1)

get - O(1)

containsKey - O(1)

remove - O(1)

size - O(1)

HashMap

put

get

containsKey

remove

size

TreeMap

Works on sortable elements, like a TreeSet Iterating over a TreeMap will iterate over the values of the set in sorted order.

put - O(log(N))

get - O(log(N))

containsKey - O(log(N))

remove - O(log(N))

size - O(1)

Solve It Together - Un-Bear-Able Zoo

open.kattis.com/problems/zoo

What basic type of data structure should we use?

List, Set, or Map?

Are there any specific implementations that can make our life easier?

What type(s) of elements will this structure use?

How will we end up using this data structure?

Solve It Together - Un-Bear-Able Zoo

What basic type of data structure should we use? - **Map**

Are there any specific implementations that can make our life easier? - **TreeMap**

What type(s) of elements will this structure use? - **String**, **Integer**

How will we end up using this data structure? -

Iterate over all animals. Map each animal type to its count.

Iterate over all keys (yay TreeMap) and print the key and its value.

Solve It Together - Adding Words

open.kattis.com/problems/addingwords

What basic type of data structure should we use?

List, Set, or Map?

Are there any specific implementations that can make our life easier?

What type(s) of elements will this structure use?

How will we end up using this data structure?

Solve It Together - Adding Words

What basic type of data structure should we use? - **Map**

Are there any specific implementations that can make our life easier? - **HashMap**

What type(s) of elements will this structure use? - **String**, **Integer** and **Integer, String**

How will we end up using this data structure? -

Iterate over all expressions. If it is a def, then put the corresponding value in both maps.

When evaluating, use the forward map to look up variables, and the backwards map to see if the result is a variable.

Google Competition Review

- False Equivalence
- Pangram
- Solve Maze
- Spiraling Matrix
- IsFib
- Two DImensional City
- Bank Error
- Train Tables
- Adding Arrays

Your Turn

Synchronizing Lists

https://open.kattis.com/problems/synchronizinglists