CSE12 - Lecture 23

Wednesday, May 29, 2024 10:00 AM

PA8 - hard deadline - Thursday @ 8am
PA6 Late/Resubmit - hard deadline - Thursday @ 8am
Exam 3 - next Wednesday (June 5th)

- https://ucsd-cse12-sp24.github.io/lectures/exam3.html
  - Trees, BST, Heaps, Iterators, Improving Lists
    - No design patterns

Lecture 23

Create a RandomStream class that generates random numbers in an enhanced for loop:.

Generating a random number:

From java.util.*

class Random
public int nextInt(int bound)

Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.

Random random;
int value = random.nextInt(100); //random number between 0 and 99

Note: always use a field, never create a new one over and over again in a loop.

```
RandomStream s = new RandomStream(10, 100);
for (Integer i : s) {
   System.out.println(i);
 }
```

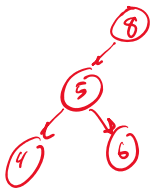
of #s bound

```
class RandomStream {
    int size = 0;
    int bound = 0;
    Random random;
    public RandomStream(int size, int bound) {
        this.size = size;
        this.bound = bound;
        this.random = new Random();
    }
    class RandomIterator implements Iterator<Integer> {
        int count = 0;
        public boolean hasNext() {
            return count < size;
        }
        public Integer next() {
            int value = random.next(bound);
            count++;
            return value;
        }
    }
    public Iterator<Integer> iterator() {
        return new RandomIterator();
    }
}
```

debugging: can use int seed to repeat sequence

Name: _____ PID: _____ Code: 1176

How would we make a BST iterator?



pre
post
in-order traversal

Iterator class (add to BST class)
- save state
  - create an ArrayList
  - fill AL with an in-order traversal
- iterator → next/hasNext → same as ArrayList

Run-Time

Constructor
- $\Theta(N)$ to traverse and copy to AL

next()
- $\Theta(1)$ to get an element from AL using an index

How would we make a Heap iterator?

use like an AList

(1) copy the heap into the iterator
- use poll() of the copy in Next()
- all elements in proper heap order

Run-time

Constructor
- $\Theta(N)$ to copy heap
- next $\Theta(\log N)$

(2) copy the array to the iterator
- sort in heap order
- in next() just the array (like an AList

Constructor
- $\Theta(N \log_2(N))$ → sort

Next
- $\Theta(1)$ → AList get