CS 61B
Spring 2022

Test 2 Review
Discussion 11: April 4, 2022

1 Coins, Trees, and Dogs?

Tod is obsessed with dog-associated cryptocurrencies. After purchasing an unhealthy volume of volatile assets, Tod decides to use the binary tree below to organize his collection.

```
                Corgi Inu
                /       \
         Dogecoin        Dogelon Mars
         /       \              \
DogeToken    Kishu Inu Coin    Shiba Inu
                         \
                         Yuki Inu Token
```

(a) Tod wishes to organize his collection alphabetically. Write out the DFS pre-order, DFS in-order, DFS post-order, and BFS (Level Order) traversals of the following binary tree. Which traversal gives the collection in sorted in alphabetical order?

(b) Tod mistakenly believes that an inorder traversal will yield the collection in alphabetical order. What data structure might Tod have been thinking of? To fix this, draw a tree such that when traversed inorder, it will yield the coins in sorted order.

(c) *Extra:* Provide an example of a tree where the DFS pre-order, DFS in-order, and BFS traversals are the same, and where the DFS post-order traversal is the opposite order of the previous three traversals.

2 Mechanical Heap Practice

Consider the following min-heap:

```
    2
   / \
  5   12
 / \
6   7
```

(a) Draw the heap after inserting the following numbers (in the given order and in succession): 4, 13, 3

(b) Now, returning back to the initially given min-heap, draw the heap after removing the minimum element twice. Assume that when bubbling down, the parent will bubble down towards the minimum of the two children if both children have lower values.

(c) *Extra:* What is the runtime of finding a specific element within the heap, assuming we have access to the underlying data structure (e.g. if the heap is represented as an array, we can scan the array)?

3 Asymptotics Review

Give the tightest bounds (either Ω/O or Θ) for the following functions.

(a) Note that `nextInt(int bound)` returns a random integer between `0` (inclusive) and `bound` (exclusive) and takes constant time.

```java
void f(int N) {
    Random rand = new Random();
    for (int i = 1; i < N; i += rand.nextInt(i) + 1) {
        for (int j = 0; j < i; j++) {
                System.out.println(i + j);
        }
    }
}
```

(b)

```java
void g(int N) {
    if (N < 10000) {
        return;
    }
    for (int i = 0; i < N; i++) {
        i++;
    }
    g(N / 2);
    g(N / 2);
}
```

4 A Bit of Practice

(a) Fill in the missing lines for `set`, a method that takes in an integer `n` and sets the `k`-th bit to to value of `y`, which is either 0 or 1. *Hint:* `0 | 0 = 0` *and* `0 | 1 = 1`.

```
int set(int n, int k, int y) {

    ______________________________________________
}
```

(b) Fill in the method `flipEveryOther` that takes in an integer `n` and flips every other bit, starting by flipping the least significant (rightmost) bit.

```
int flipEveryOther(int n) {
    int m = ______________________________________________

    ______________________________________________
}
```

5 Hash Codes and Runtime

Suppose we’re given the following `Student` class definition.

```
class Student {
    public final static String isStudent = "yes";

    public String name;
    public String major;
    public String school;
    public int year;
    public int id;

    public Student(String name, String major, String school, int year, int id) {
        ...
    }

    @Override
    public int hashCode() {
        ________________________________________
    }

    @Override
    public boolean equals(Object o) {
        if (o == null || this.getClass() != o.getClass()) {
            return false;
        }
        Student other = (Student) o;
        return school.equals(other.school) && id == other.id;
    }
}
```

(a) Assume that `major`, `school`, and `id` are never modified after initializing a `Student`, but `year` can be modified. For each of the following hash codes, answer whether they are ”valid” or ”invalid” and justify why.

1. `return isStudent.hashCode();`

2. `return id`

3. `return year + id;`

4. `return school.hashCode() + id;`

5. `return 17 * school.hashCode() + 5 * major.hashCode() + id;`

(b) 1. What is the best case and worst case runtime of the following function, in terms of N and K? Suppose N is equal to `roster.size()` and the `name`, `major`, and `school` fields for all `Students` are $\Theta(K)$ length. Assume that computing hash codes take constant time.

```java
Set<Student> removeDuplicates(ArrayList<Student> roster) {
    Set<Student> noDuplicates = new HashSet<>();
    for (Student s : roster) {
        if (noDuplicates.contains(s)) {
            System.out.println("Duplicate found: " + s.name);
        }
        noDuplicates.add(s);
    }
    return noDuplicates;
}
```

Best Case: $\Theta(\qquad\qquad)$ Worst Case: $\Theta(\qquad\qquad)$

2. Which of the valid hash codes from above would be most likely to cause these runtimes?