CS61B Week 6: Bit Business & Algorithmic Analysis

Bit Business

1. Write 47 in binary. 0b101111
2. Write 463 in hexadecimal. 0x1CF

For the next 4 problems assume we are working with byte-length numbers.

3. Write `0b11001001` in decimal. What is `0b11001001 >> 2`? What is it in decimal?
   `0b11001001 = -55`
   `0b11001001 >> 2 = 0b11110010 = -14`
4. Write `0xC9` in decimal. What is `0xC9 >>> 2`? What is it in decimal?
   `0xC9 = -55 = 0b11001001`
   `0xC9 >>> 2 = 0b11001001 >>> 2 = 0b00110010 = 0x32 = 50`
5. What is `23 << 2`? What about `23 << 3`? `23 << 4`?
   `23 = 0b00010111`
   `23 << 2 = 0b00010111 << 2 = 0b01011100 = 92`
   `23 << 3 = 0b00010111 << 3 = 0b10111000 = -72`
   `23 << 4 = 0b00010111 << 4 = 0b01110000 = 112`
6. What is `35 >> 2 << 2`?
   `35 = 0b00100011`
   `35 >> 2 << 2 = 0b00100011 >> 2 << 2 = 0b00001000 << 2 = 0b00100000 = 32`
7. Assume a, b, and c are integers. What is `(((a | b) & a) ^ c) ^ c`?
   `(((a | b) & a) ^ c) ^ c = ((a) ^ c) ^ c = a`

Algorithmic Analysis

1. What is the big-Theta running time of the following code in terms of n?

```
public static void printStuff(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < i; j++) {
            System.out.println(i + j);
        }
    }
}
```

   $\theta(n^2)$

2. Assume sortedList is a sorted list of length n with no duplicates. What is the running time of the function useless? What does it print?

```
static void useless(int[] sortedList) {
    for (int i = 0; i < sortedList.length; i++) {
        System.out.println(foo(sortedList, sortedList[i]));
    }
}
```

```
static int foo(int[] lst, int toFind) {
    return bar(lst, toFind, 0, lst.length);
}

static int bar(int[] lst, int toFind, int lower, int upper) {
    if (lower == upper) {
        return -1;
    }
    int mid = (lower + upper) / 2;
    if (lst[mid] > toFind) {
        return bar(lst, toFind, lower, mid);
    } else if (lst[mid] < toFind) {
        return bar(lst, toFind, mid + 1, upper);
    }
    return mid;
}
```

$\theta(n \log(n))$

The function useless prints the integers 0, 1, 2, ..., n-1 in order on subsequent lines. We are essentially running a binary search for each element of the list, in the list itself, so the function will find each element and print out its index in the list. Since we are running through the list in order, we will simply print out the list indices in order.

3. The following function returns the nth fibonacci number. What is its running time?

```
public static int fib(int n) {
    if (n == 1 || n == 0)
        return 1;
    return fib(n - 1) + fib(n - 2);
}
```

The running time is exponential. Technically it converges to $\theta(\phi^n)$, where ϕ is the golden ratio, but it suffices to put something like $O(2^n)$.

4. Rewrite the fibonacci function to use iteration instead of recursion. What is the new running time?

```
public static int iterativeFib(int n) {
    if (n == 1 || n == 0)
        return 1;
    int last = 1;
    int cur = 1;
    int i = 1;
    while (i < n) {
       int next = last + cur;
       last = cur;
       cur = next;
       i++;
    }
    return cur;
}
```

$\theta(n)$

Sample Midterm Question of the Week:
What does the following code print?

```
public class What {
    public long n;
    public void increment() {
        n++;
    }
    public static void reset(What w) {
        w.increment();
        w = new What();
        w.n = 0;
    }
    public static void main(String[] args) {
        What w = new What();
        w.n = 7;
        reset(w);
        System.out.println("The number is " + w.n);
    }
}
```

The program prints "The number is 8". The reset method, via the increment method, successfully increments the n field in the original What object from 7 to 8. Then the reset method creates a new What object and sets its n field to zero. But main passed the reference w by value. When reset changes its local variable w, the local variable w in main still references the original object, which still holds an 8 in its field n.

Sample Interview Questions of the Week:

1. ```
   int temp = b;
   b = a;
   a = temp;
   ```

   The above code snippet swaps the value of two integers, a and b. Write code to swap them without using a temporary variable.

   ```
   a = a ^ b;
   b = a ^ b;
   a = a ^ b;
   ```

2. You're given an array of integers of length n such that only one of the elements in the array appears once, while every other element appears exactly twice. How can you figure out the value of this one element that appears only once in O(n) time and O(1) space? (O(1) space means you use a constant number of variables no matter how long the given array is)

   XOR everything in the array together. For any integer `a`, we can see that `a ^ a = 0`, so we will be left with `0 ^ x` for our unique element `x`, and this will give us `x`.
   ```