CS 61B Spring 2022

More Algorithmic Analysis

Discussion 8: March 7, 2022

1 Best and Worst Case with Iteration

For the following functions, provide asymptotic bounds for the best case and worst case runtimes in $\Theta(\cdot)$ notation.

(a) Give the best and worst runtimes in terms of N.

```
1
2  public static void removeIndex(int[] arr, int i) {
3      // Assume i > 0
4      int N = arr.length;
5      for (int j = i; j < N; j += 1) {
6          arr[j - 1] = arr[j];
7      }
8  }
```

(b) Give the best and worst case runtimes in terms of M and N. Assume that `slam()` $\in \Theta(1)$ and returns a boolean.

```
1  public void comeon(int M, int N) {
2      int j = 0;
3      for (int i = 0; i < N; i += 1) {
4          for (; j < M; j += 1) {
5              if (slam(i, j))
6                  break;
7          }
8      }
9
10     for (int k = 0; k < 1000 * N; k += 1) {
11         System.out.println("space jam");
12     }
13 }
```

2 Best and Worst Case with Recursion

For the following recursive functions, provide asymptotic bounds for the best case and worst case runtimes in $\Theta(\cdot)$ notation.

(a) Give the runtime in terms of N.

```
public void andslam(int N) {
    if (N > 0) {
        for (int i = 0; i < N; i += 1) {
            for (int j = 1; j < 1024; j *= 2) {
                System.out.println(i + j);
            }
        }
        andslam(N / 2);
    }
}
```

(b) Give the runtime for `andwelcome(arr, 0, N)` in terms of N, where N is the length of the input array `arr`. `Math.random()` returns a double with a value from the range [0,1).

```
public static void andwelcome(int[] arr, int low, int high) {
    System.out.print("[ ");
    for (int i = low; i < high; i += 1) {
         System.out.print("loyal  ");
    }
    System.out.println("]");
    if (high - low > 1) {
        double coin = Math.random();
        if (coin > 0.5) {
            andwelcome(arr, low, low + (high - low) / 2);
        } else {
            andwelcome(arr, low, low + (high - low) / 2);
            andwelcome(arr, low + (high - low) / 2, high);
        }
    }
}
```

(c) Give the runtime in terms of N.

```
public int tothe(int N) {
    if (N <= 1) {
        return N;
    }
    return tothe(N - 1) + tothe(N - 1) + tothe(N - 1);
}
```

(d) Give the runtime of `recurse` in terms of N.

```
public static int recurse(int N) {
    return helper(N, N / 2);
}
private static int helper(int N, int M) {
    if (N <= 1) {
        return N;
    }
    for (int i = 1; i < M; i *= 2) {
        System.out.println(i);
    }
    return helper(N - 1, M) + helper(N - 1, M);
}
```

(e) *Extra* Give the best case and worst case runtimes for `find` in terms of N, where N is the length of the input array `arr`.

```
1  public static boolean find(int tgt, int[] arr) {
2      int N = arr.length;
3      return find(tgt, arr, 0, N);
4  }
5  private static boolean find(int tgt, int[] arr, int lo, int hi) {
6      if (lo == hi || lo + 1 == hi) {
7          return arr[lo] == tgt;
8      }
9      int mid = (lo + hi) / 2;
10     for (int i = 0; i < mid; i += 1) {
11         System.out.println(arr[i]);
12     }
13     return arr[mid] == tgt || find(tgt, arr, lo, mid)
14                            || find(tgt, arr, mid, hi);
15 }
```