Answers to Cracking the Coding Interview in LaTeXby Ross Spencer

1 Big O

Big O Additional Problems:

1.1 O(b)

1.2 O(b)

1.3 O(1)

1.4 O($\frac{a}{b}$)

1.5 O($\log_2(n)$)

1.6 O($\sqrt[2]{n}$)

1.7 O(n) in the case that each node has 1 child in the same direction (degenerate tree).

1.8 O(n), as you have no heuristics on where the node is located

1.9 O(n^2) as each copy is $1 + 2 + 3 + ... + n - 1 <= n(n) \in$ O(n^2)

1.10 O($\log_1 0(n)$), which is equalivent to O($\log_2(n)$) (up to a constant factor for change of base)

1.11 Checking if is in order takes O(s) in size of string s, otherwise makes successive calls to every possible string with c^s possibilities, so O($s * c^s$)

1.12 Total is O($b$log$b$) for mergesort + $a$log$b$ for binary searching b for each int in a. So, O(($a + b$)log$b$).

2 Arrays & Strings

Chapter 1: Arrays & Strings Interview Questions

1.1

Algorithm 1: IsUnique

```
arr = zeros(26);
for char c in string do
    if arr[int(c)] == 0 then
        arr[int(c)] += 1;
    else
        Return False
    Return True
```

1.2

Algorithm 2: IsPermutation

```
if len(string1) != len(string2) then
    Return False;
arr = zeros(26);
for char c in string1 do
    arr[int(c)] += 1;
for char c in string2 do
    arr[int(c)] -= 1;
for int i = 0; i < 26; ++i do
    if arr[i] != 0 then
        Return False;
Return True;
```

1.3

Algorithm 3: URLify

```
arr = ‘’
for char c in string1 do
    if c == ’ ’ then
        arr += ’%20’
    else
        arr += c
Return arr;
```

1.4 Thought process: a palindrome has a multiple of 2 of all but at most one character (e.g., ...abcdcba..., so d could appear an odd number of times as long as the rest appear an even number of times). Iterate through string and count all chars, then set a boolean flag variable that results in False when multiple characters appear an odd number of times.

Algorithm 4: Palindrome Permutation

```
dict count =  for char c in string1 do
    count(c) += 1
boolean oddAlreadyPresent = False for char c in count do
    if count(c) mod 2 == 1 then
        if oddAlreadyPresent == False then
            oddAlreadyPresent = True
        else
            Return False
Return True
```

1.5

Algorithm 5: One Away

```
if string1 == string2 then
    Return True
if abs(len(string1)-len(string2))>1 then
    Return False
diffsFromMissing = 0
dict count =  for c in string1 do
    count[c] += 1
for c in string2 do
    //Catch error if not in count by adding 1 to diffsFromMissing
     count[c] -= 1
if diffsFromMissing > 1 then
    Return False
if sum(counts) > 1 then
    Return False
Return True
```