Final Exam

CS 4413 Algorithm Analysis

Q1

$T(n) = 2T(n-1) + 3T(n-2)$, $T(0) = 1$, $T(1) = 2$

characteristic equation using $T(n) = r^n$

$r^n = 2r^{n-1} + 3r^{n-2}$

divide by r^{n-2} and rearrange to get

$r^2 - 2r - 3 = 0$ or $(r-3)(r+1) = 0$

giving us roots of

$r = -1, 3$

giving us a general solution of

$T(n) = A * 3^n + B * (-1)^n$

using the initial conditions we can determine

$A = ¾$ and $B = ¼$

giving us a final solution of

$T(n) = ¾ * 3^n + ¼ * (-1)$

Q2

given binary numbers a and b, represent them as bit arrays, with the least significant bit being in the first index of the array, and the most significant bit being in the last index of the array

construct the carry array c such that element $c_0 = 0$, $c_i = (a_i \wedge b_i) \vee (a_i \wedge c_{i-1}) \vee (b_i \wedge c_{i-1})$

construct the sum array s such that element $s_i = a_i \oplus b_i \oplus c_{i-1}$

note: if the MSB of c is 1, overflow has occurred

this array now represents the binary sum of a and b, with the LSB being in the first index

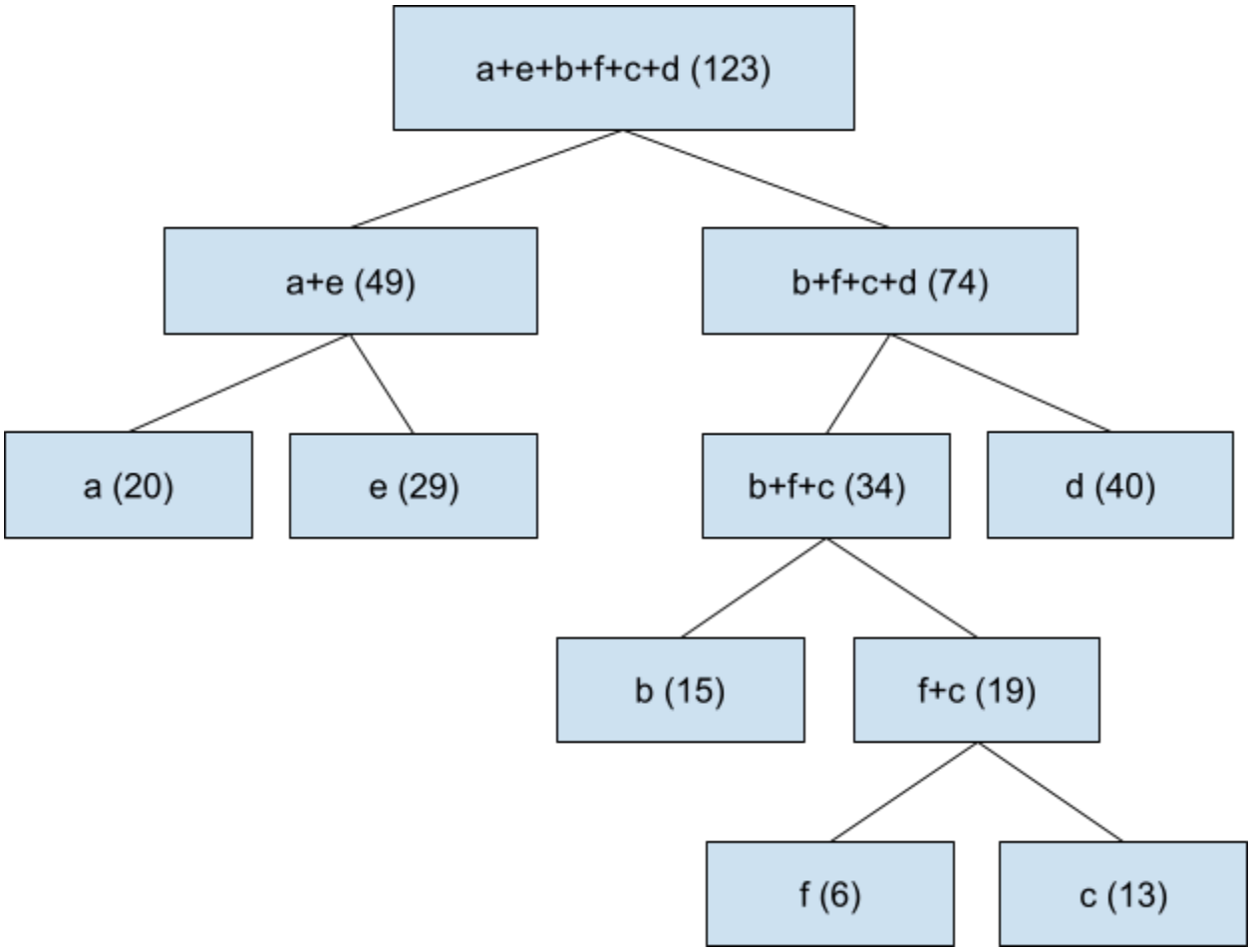
Q3a

Fixed Length: Each symbol in the alphabet is represented by a code of the same set length, regardless of that symbol's frequency.

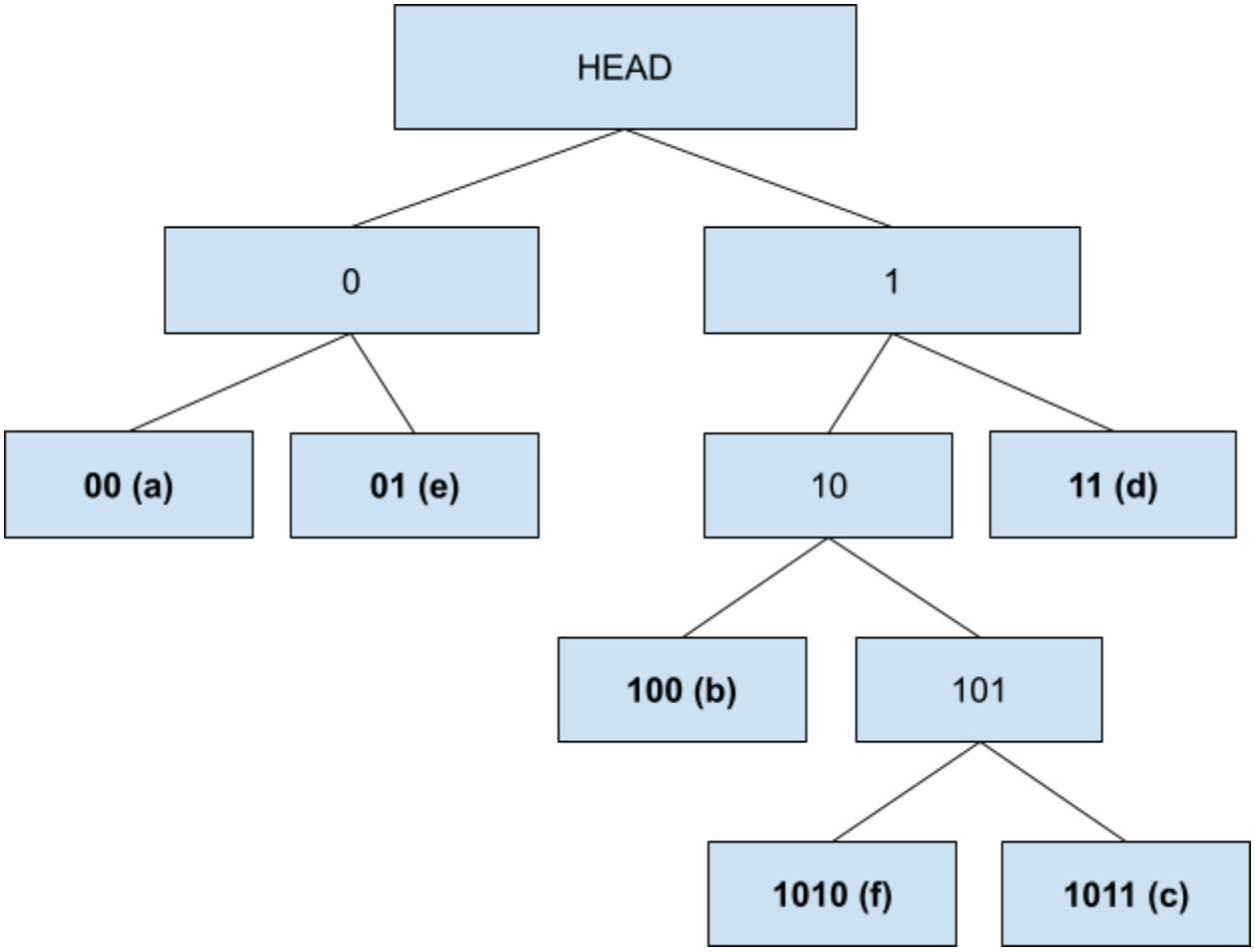
Variable Length: Symbols are represented by a code that is proportional to the frequency of that symbol appearing. Symbols that appear more frequently have shorter codes, while symbols that appear less frequently have longer codes.

Q3b

Construct a tree that combines symbols with the lowest frequencies til they are all combined:



We can then assign a 0 to the left node, and a 1 to the right node of the head, and continue to do this recursively until we have reached the bottom of the tree. This will now give us the Huffman Codes for each symbol.



Doing this gives us the following codes:

a	00
b	100
c	1011
d	11
e	01
f	1010

Q3c

Fixed Length: 3 bits ($\lceil \log_2(6) \rceil$)
Average Variable Length: (20(2) + 15(3) + 13(4) + 40(2) + 29(2) + 6(4)) / 123 = 2.43

Compression Ratio: ⌈ $\log_2(6)$ ⌉ / 2.43 = **1.23**

Q3d

the word “faced” becomes **10100010110111**

using the tree to decode this, you would split it as: 1010 | 00 | 1011 | 01 | 11

which produces the word “**faced**”

Q4a

```python
import math

def logStar(n):
    return 0 if n <= 1 else 1 + logStar(math.log(n, 2))

print(logStar(10**12))
```

log * (10^{12}) = 5

for n = 2^k and f (n) = n / 2, **f * (n) = k**

Q4b

n = 2^(2^k), T(n) = T(√n) + 2, T(2) = 1

n	k	T(n)
2	0	1
4	1	3
16	2	5
256	3	7

T(n) = 2 * log (log (n)) + 1, O(n) = log (log (n))

Q5

We can solve the TSP using a Minimum Spanning Tree using a preorder traversal

1. Compute the distance between every pair of cities
2. Construct a MST (using Prim's Algorithm)
3. Perform a recursive preorder traversal of the MST

The MST Matrix:

0	5	7.07106781	4.12310563	2.23606798	5.09901951
5	0	0	0	0	0
7.07106781	0	0	0	0	0
4.12310563	0	0	0	0	0
2.23606798	0	0	0	0	0
5.09901951	0	0	0	0	0

Approximate Tour Cost: **22.6205**