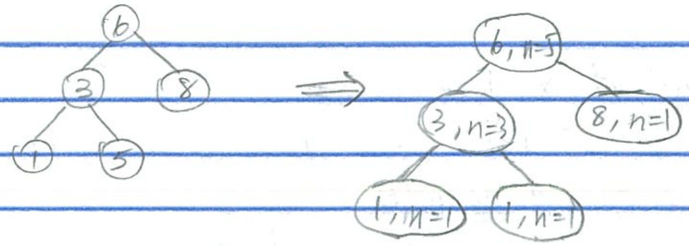
CS 240 Assignment 5

1a). We introduce a counter n on each node which stores the number of nodes beneath the current node (as the root of a subtree). For example,



Now, we use binary search on the root node to find the boundary paths. For each node on the path, if the node is in range, 1 is added to the total of the range counting query. For each node inside the boundary, we just need to look at the n value on the root of the subtree rather than doing binary search one more time, which takes O(1). Then, we add the number of nodes inside the boundary paths to the counter of the query.
The modification takes O(1) to increment the count for each boundary node and then for its top internal node that run the height of the tree of each path. Thus,

$$2\sum_{i=1}^{\lg n} O(1) = 2(O(\log n)) \in O(\log n).$$

∴ The modified range counting query can be performed in $O(\log n)$.

b). We use 2D range troe T sorted by x-coord, and T has Tassoc(v) sorted by y-cood for every point v. Same process as a), store the number of nodes beneath v as another field of v. Then, we do binary search on x-coord to find the two boundary paths. Then, we check for every boundary node visited to see whether it satisfies the y-coord range, count also increased by 1.
For inside nodes, find the top node of their subtree. Perform a search as a) on associated tree. We will find out how many nodes on the inside nodes have both x & y-coord in range, which takes O(log n).
Thus:
Runtime: $T(n) = \sum_{i=1}^{\lg n} O(\log n) \in O((\log n)^2)$

c. # of nodes = $\underbrace{x \text{ nodes}}_{n}$ + y nodes

What about y ?

Let's consider height of x-tree be 2

When $n = 0$, there's 2^0 node, each node's associate tree has $2^3 - 1 = 7$ y nodes

when $n = 1$, there's $2^1 = 2$ nodes, and each node's associate tree has $2^2 - 1 = 3$ y nodes

when $n = 2$, ... $2^2 = 4$ nodes, . . .

. . . $2^1 - 1 = 1$ nodes

∴ # of y nodes $= 2^0 \times 7 + 2^1 \times 3 + 2^2 \times 1$

∴ The general formula is :

$$\sum_{i=1}^{h} (2^i \cdot (2^{h+1-i} - 1)$$

$$= \sum_{i=0}^{h} 2^{h+1} - 2^i$$

$$= 2^{h+1} \cdot (h+1) - \sum_{i=0}^{h} 2^i$$

$$= 2^{h+1} \cdot h + 1 .$$

∵ $h = \log_2(n+1) - 1$

∴ Y nodes $= 2^{\log_2(n+1)+1-1} (\log_2(n+1) - 1) + 1 .$

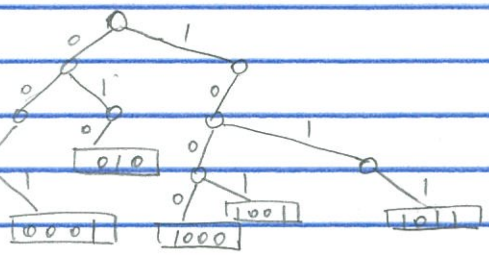
$$= (n+1)\log_2(n+1) - n - 1 + 1$$

$$= n \log_2(n+1) + \log_2(n+1) - n .$$

∴ Total # of nodes $= n + n\log_2(n+1) + \log_2(n+1) - n$

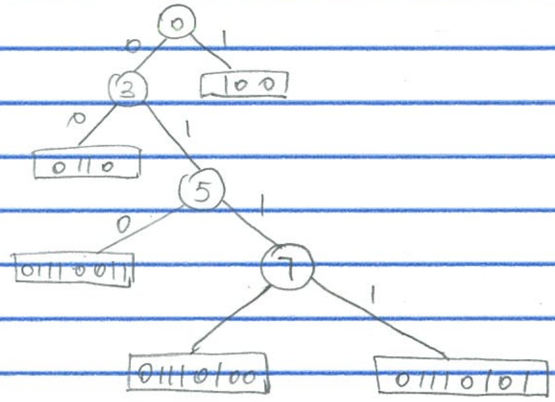
$$= n \log_2(n+1) + \log_2(n+1)$$

2a).



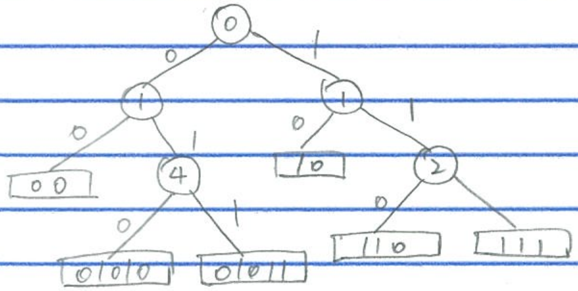
0
1
0
1
0
0
1
0
0
1
1
0
1
1
0001
010
1001
1000
1011

b).



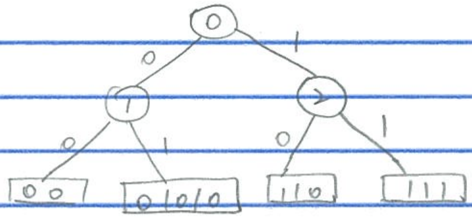
0
0
1
3
100
0
1
0110
5
0
1
0111 0011
7
1
0111 0100
0111 0101

c).



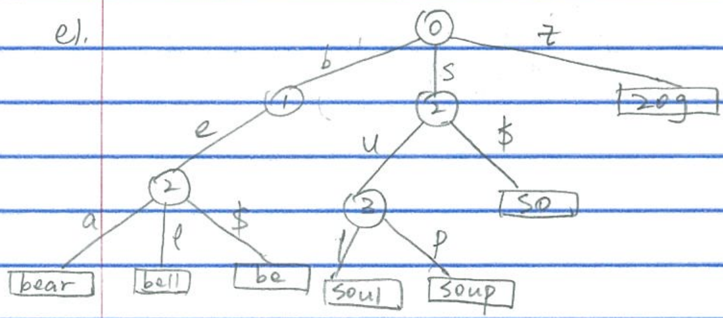
0
0
1
1
1
0
1
0
1
00
4
10
2
0
1
0
01010
01011
110
111

d).



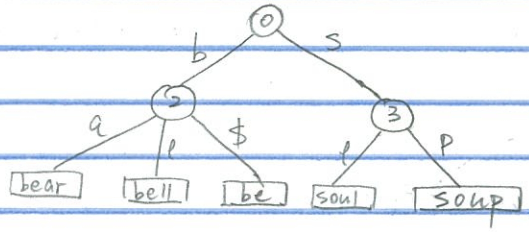
0
0
1
1
2
0
1
0
1
00
01010
110
111

e).



0
b
s
z
1
2
zog
e
u
$
2
so
a
l
$
3
l
p
bear
bell
be
soul
soup

f).



3a).

j	P[0...j]	P[1...j]	F[j]
0	a	–	0
1	ab	b	0
2	aba	ba	1
3	abab	bab	2
4	ababa	baba	3
5	ababac	babac	0

b).

a	b	c	a	a	b	a	a	b	a	b	a	b	a	c	a	b	c	a	a
a	b	a																	
		a																	
			a	b															
				a	b	a	b												
						(a)	b												
							a	b	a	b	a	c							
									(a)	(b)	(a)	b	a	c					

c). Going through values in F[j], where $0 \le j \le len(P \Phi T)$, and check whether $F[j] \geq m$. If this kind of j exists, p is a substring of T. Otherwise, p is not a substring T. Because P does not contain Φ, and $(P + \Phi)$ has no suffix=P. Therefore, the suffix =p can only be found in T.
Thus, if the prefix =p and suffix =p, P occurs at T.

4a).

c	a	r	t
L(c)	5	0	4

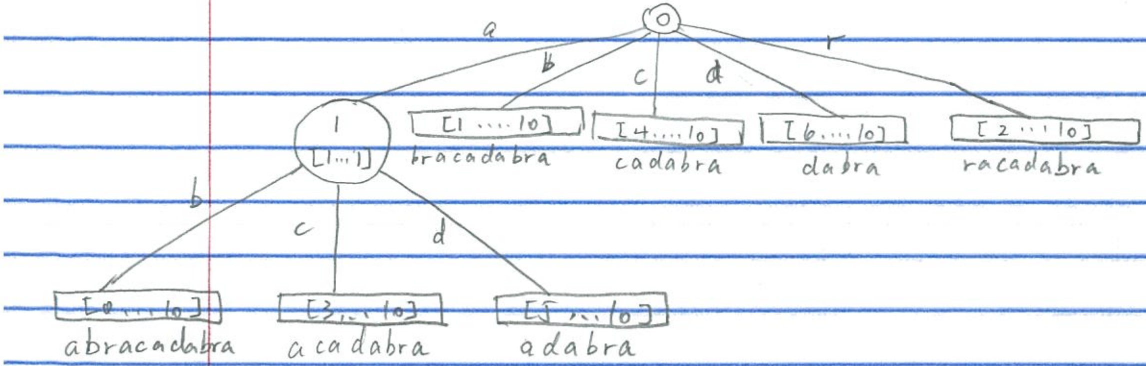
b).

```
i      0   1   2   3   4   5   6
P[i]   r   a   t   a   t   a   t
S[i]  -7  -6   0  -4   0  -2   5
           r   a   t   a   t   a
               r   a   t   a   t
                   r   a   t   a
                       r   a   t
                           r   a
                               r
                               ␣
                           ␣   ␣
                       ␣   ␣   ␣
                   ␣   ␣   ␣   ␣
```

∴

i	0	1	2	3	4	5	6
P[i]	r	a	t	a	t	a	t
S[i]	-7	-6	0	-4	0	-2	5

5. $T = \overset{0}{a}\overset{1}{b}\overset{2}{r}\overset{3}{a}\overset{4}{c}\overset{5}{a}\overset{6}{d}\overset{7}{a}\overset{8}{b}\overset{9}{r}\overset{10}{a}$


0
a
b
c
d
r
1
[1...1]
[1 ....10]
bracadabra
[4.....10]
cadabra
[6.....10]
dabra
[2 ...10]
racadabra
b
c
d
[0....10]
abracadabra
[3,...10]
acadabra
[5 ...10]
adabra