A5

Jianan Luo
20523403

1. a) Preprocess modify:
add 2 field for each node in BST.
The first one represent # of node under its left sub-tree. // name: ln
The second one represent # of node under its right sub-tree. // name: rn

Lets say the query is [a, b], and the BST t.

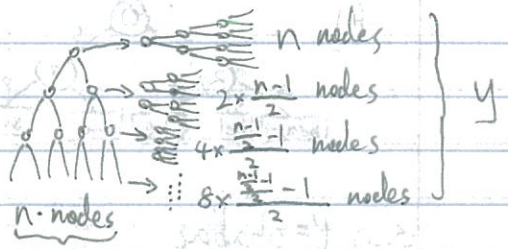
Range Counting:
First of all, we need to find the first node in the query range. Just normal BST-Search which takes log(n) time. If can't find, then return 0. // no number within the range.
After we find the first node with range, we return rangeCountLeft + 1 + rangeCountRight on that node.
RangeCountLeft take a node and check if the node within [a, b]. If yes, return 1 + its rn + RangeCountLeft on its left node. It will takes runtime log(n). Just trace from root to leaf like a normal BST-Search.
(If no, return RangeCountLeft on its right subtree)
RangeCountRight takes a node and check if the node within [a, b]. If yes, return 1 + its ln + RangeCountRight on it right node. If no, return RangeCountRight on its left sub-tree. It will takes runtime log(n). Just trace from root to leaf like a normal BST-Search.
The Base case for RangeCountLeft and RangeCountRight is when the node they take is a leaf. In the case, return 1 if in range [a, b], 0 otherwise.
So basically, this algorithm's runtime can be write as $O(\log n) + O(\log n) + O(\log n) \in O(\log n)$, according to the specification and runtime describe above.
Therefore the runtime for the algorithm is $O(\log n)$.

b) Preprocess modify:
add 2 field for each node in x-BST and y-BST.

Range Counting:
All other thing remands the same. Only change for each node we found on the range [a, b], we return what we returned in part a) + RangeCounting apply to the y-BST relates to the current node. The part after + represent total # of node with the range [c, d].
(query change to [c, d])
Therefore according the runtime from part a) which is $O(\log n)$, the new algorithm's runtime will be $O(\log n) \cdot O(\log n)$ since the new implement is apply to each single node in part a).
So the new runtime will be $O(\log n) \cdot O(\log n)$ which can be write as $O((\log n)^2)$.
Therefore the runtime is $O((\log n)^2)$.

Lets say the query for x is [a, b]
y is [c, d]

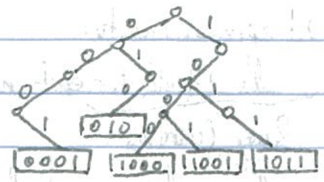
c) The two-dimensional range tree is like:



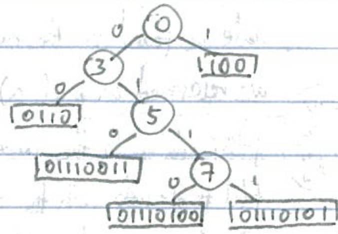
$$\text{Total node} = n + \underbrace{\left(n + 2 \times \frac{n-1}{2} + 4 \times \frac{\frac{n-1}{2}-1}{2} + 8 \times \frac{\frac{\frac{n-1}{2}-1}{2}-1}{2} + \cdots\right)}_{\lfloor \log n \rfloor \text{ terms}}$$

$$= n + (n + (n-1) + (n-2-1) + (n-4-2-1) + \cdots)$$
$$= n + (n\lfloor \log n \rfloor + (1 + 3 + 7 + 15 + \cdots))$$
$$= n + n\lfloor \log n \rfloor - ((2^1-1) + (2^2-1) + (2^3-1) + (2^4-1) + \cdots)$$
$$= n + n\lfloor \log n \rfloor + \lfloor \log n \rfloor - (2^1 + 2^2 + 2^3 + \cdots + 2^{\lfloor \log n \rfloor})$$
$$= n + (n+1)\lfloor \log n \rfloor - 2(2^0 + 2^1 + 2^2 + \cdots + 2^{\lfloor \log n \rfloor - 1})$$
$$= n + (n+1)\lfloor \log n \rfloor - 2(2^{\lfloor \log n \rfloor} - 2)$$
$$= n + (n+1)\lfloor \log n \rfloor - 2^{\lfloor \log n \rfloor + 1} + 4$$

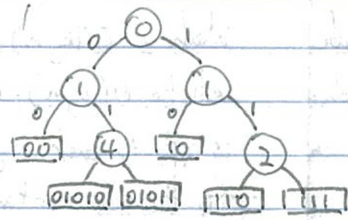
2.a)



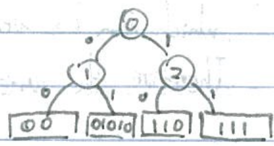
b)



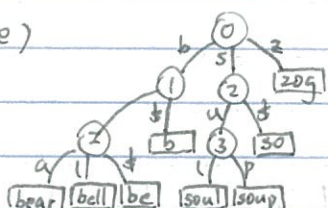
c)



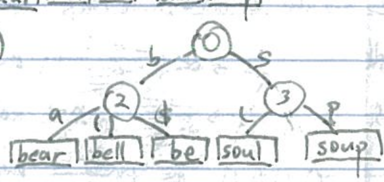
d)



e)



f)



3.a) P= ababac

j	P[1...j]	P	F[j]
0	—	ababac	0
1	b	ababac	0
2	ba	ababac	1
3	bab	ababac	2
4	baba	ababac	3
5	babac	ababac	0

F = 001230

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) All we need to do to find the first occurrence of P in T is to check from i=0 to length of PϕT−1, we find the first F[i]=m, where m is the length of P. And (i−2m) is the first index of the first occurrence of P in T. For example, P=abc, ϕ=x, T=aaabccc, ⇒ PϕT=abcxaaabccc. m=3, (length of abc)
F of PϕT is 00001112300 (0 1 2 3 4 5 6 7 8 9 10), when i=8, (index 8), F[8]=3 (3 is m, length of P). So (i−2m) is (8−2×3)=2. So the first index of the first occurrence of P in T is 2. Since T=aaabccc (0 1 2 3 4 5 6). So Proved ▨. Note: if we can't find F[i]=m, then, there is no occurrence of P in T.

4. a) P= ratatat, Σ={a, r, t}

c	a	r	t
L(c)	5	0	6

b) P= ratatat

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: abracadabra

0	1	2	3	4	5	6	7	8	9	10
a	b	r	a	c	a	d	a	b	r	a