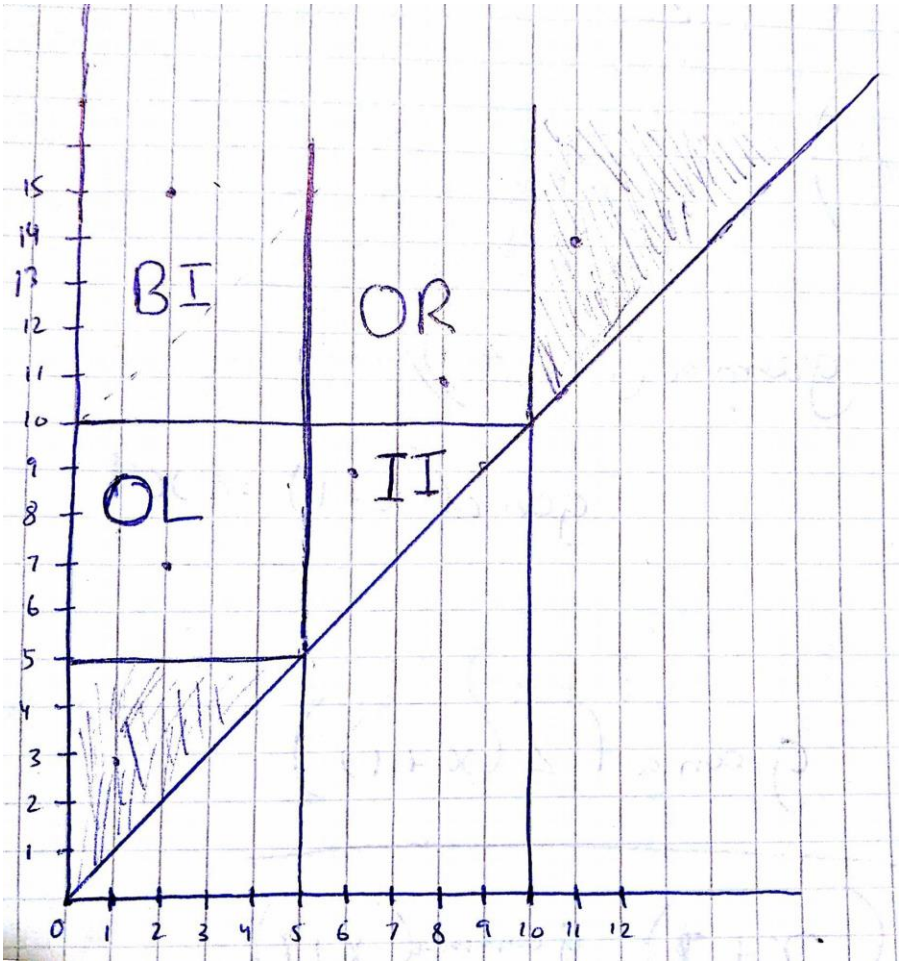
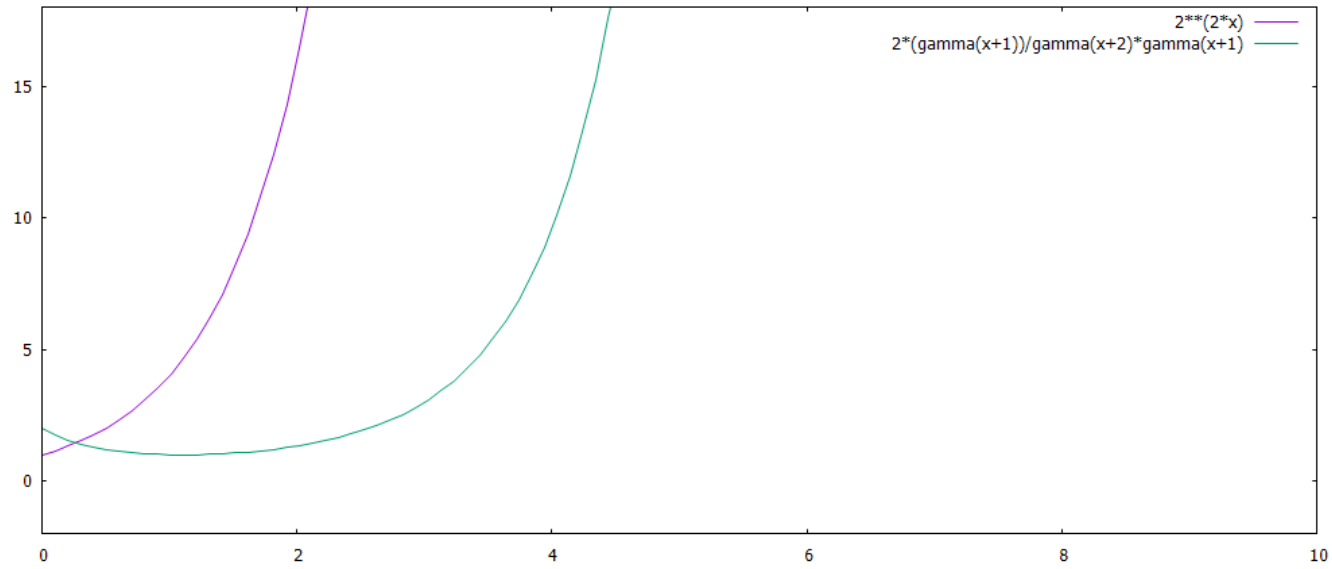
EX-1: Intervals in 2D Co-ordinates:

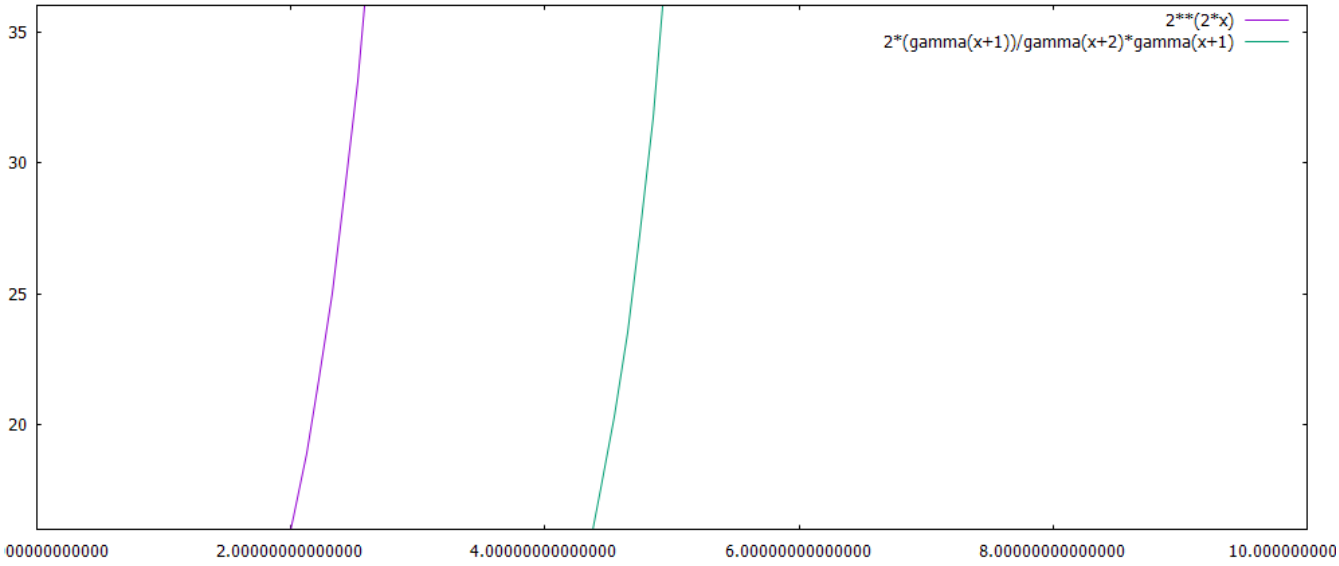


I locked the interval **[5,10]**. Here we have 6 cases. I plotted the same numbers which were in hint and then marked the intervals according to the case in which they lie.In above figure the two shaded areas are non-overlapping intervals. The upper shaded is Non-overlapping from right and the bottom shaded is Non-overlapping from left of the interval.

The **OL** area is overlapping from left,the **OR** is overlapping from right,the **II** is the inside internal and the **BI** is the bounding interval. Kindly note that we are looking at the areas above the diagnonal line.

EX-2: Catalan numbers vs 2^2n

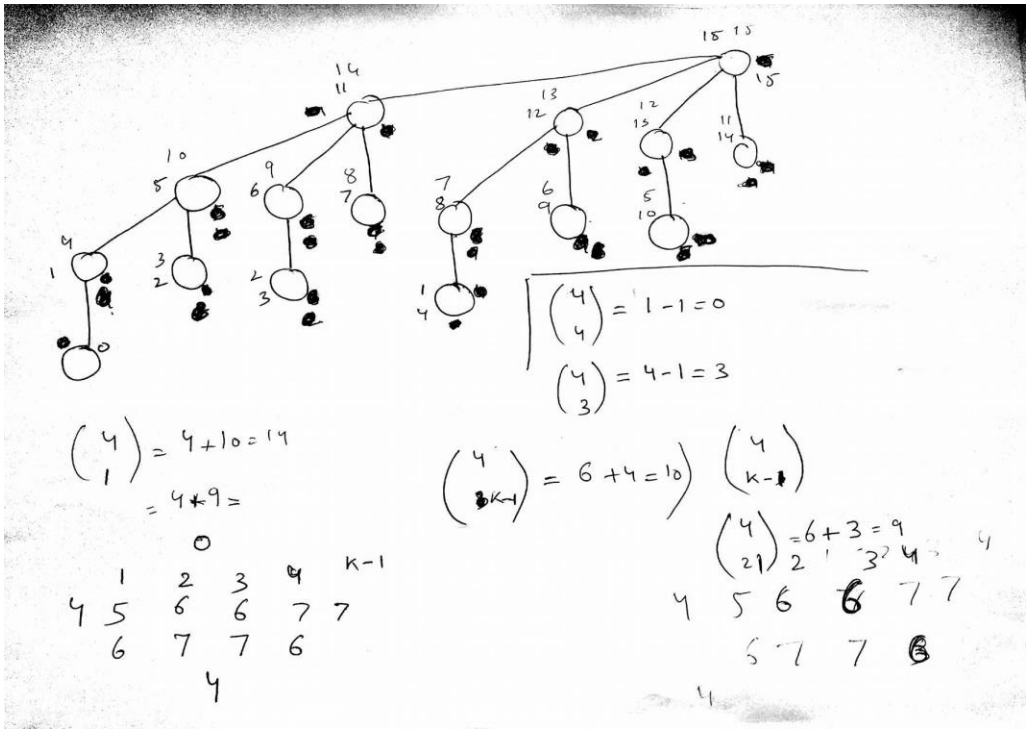


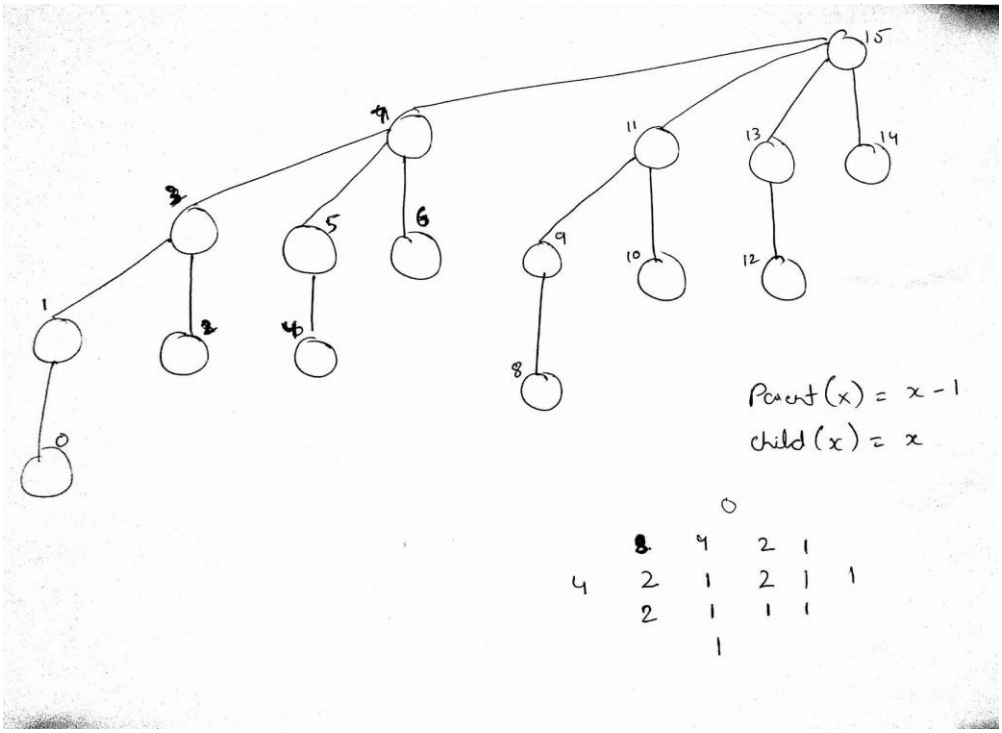


As the graph shows after certain point n_0 the Catalan number grows slower than 2^{2n}. So the Catalan number is upper bounded by **$O(2^{2n})$.**

EX-3: Binomial Heap:

So, I came up with different ideas of traversing a four-degree binomial heap just to store the nodes in an array in such a way that we get a perfect indexed based formula to find parent, child, and first sibling of a node.



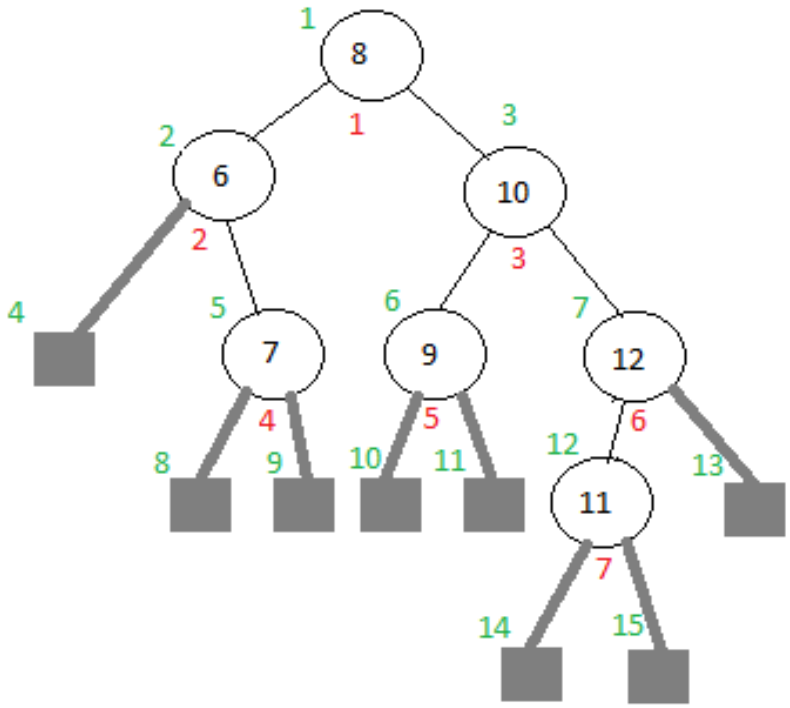


First I used the level order representation mapped them on number of nodes in a particular level combination in order to find a pattern. I did both the left to right and right to left level order but there wasn't any specific pattern or formula that could generalize the parent, child and sibling's representation in array.

Then I did the post-order traversal but the pattern here was also un-clear. I was able to find some repetitions in calculations by doing some addition and subtraction of combinations but still the argument was not strong enough to generalize the parent, child and siblings in this one too.

EX-4: Succinct Representation

Data: 8,10,6,9,7,12,11



1 2 3 4 5 6 7

1 1 1 0 1 1 1 0 0 0 1 0 0 0

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

According to the formulas:

left child(x) = [2x]
right child(x) = [2x+1]
parent(x) = [⌊x/2⌋]

Let's check it:

Left child of value 12: 2(6) =12 which means the left child of 6th node is at 12th node.

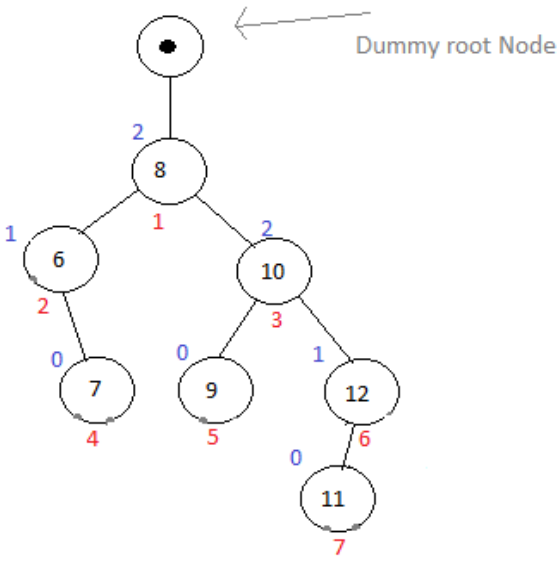
Right child of value 12: 2(6) + 1 =13 which means the right child of 6th node is at 13th node.

Parent of value 9: 6/2=3 which means the parent this node is at 3rd node.

EX-5: Succinct Part-2

Data: 8,10,6,9,7,12,11

1. **Level wise:**



1 2 3 4 5 6 7
1 0 1 1 0 1 0 1 1 0 0 0 1 0 0

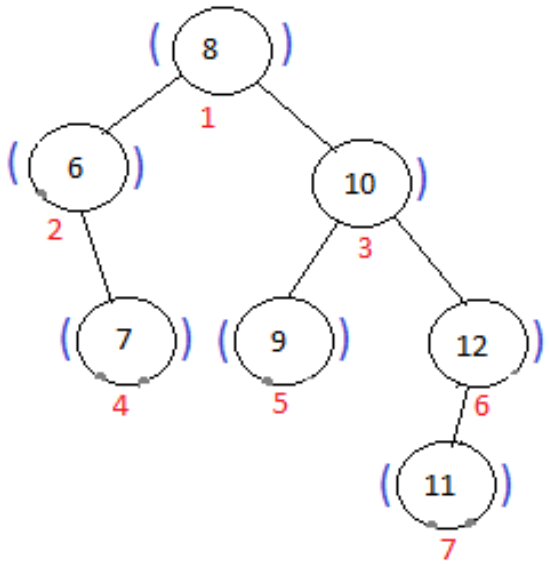
According to the formulas:

Let's check it:

child of value 12: after 6 zeros so child is at 7

Parent of value 9: 3 zeros so parent is at 3.

2. **Parenthesis Representation:**



((( )) (( ) (( ))))

1 2 4 4 2 3 5 5 6 7 7 6 3 1

According to the formulas:

parent – enclosing parenthesis

first child – next parenthesis (if 'open')

Let's check it:

child of value 12: first starting bracket is at 7 so child is at 7

Parent of value 9: enclosing bracket is 3 so parent is at 3.

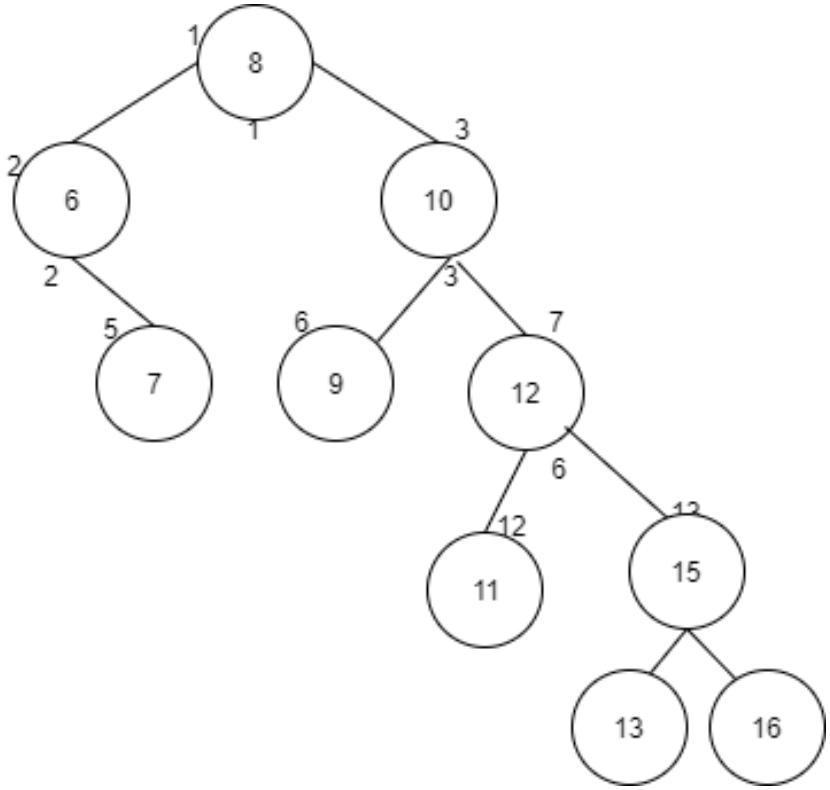
My Thoughts:

In my point of view all of them are very interesting representations but generally the parenthesis one feels natural because we are more familiar with these brackets in solving equations etc. The level wise requires an extra dummy node it doesn't feel that much natural.

EX-6: Subtree and Lowest Common Ancestor

As suggested in question I have added two extra nodes because the tree was too small.

a)-



Subtree size(v)=(findclose(S,v)-v+1)/2

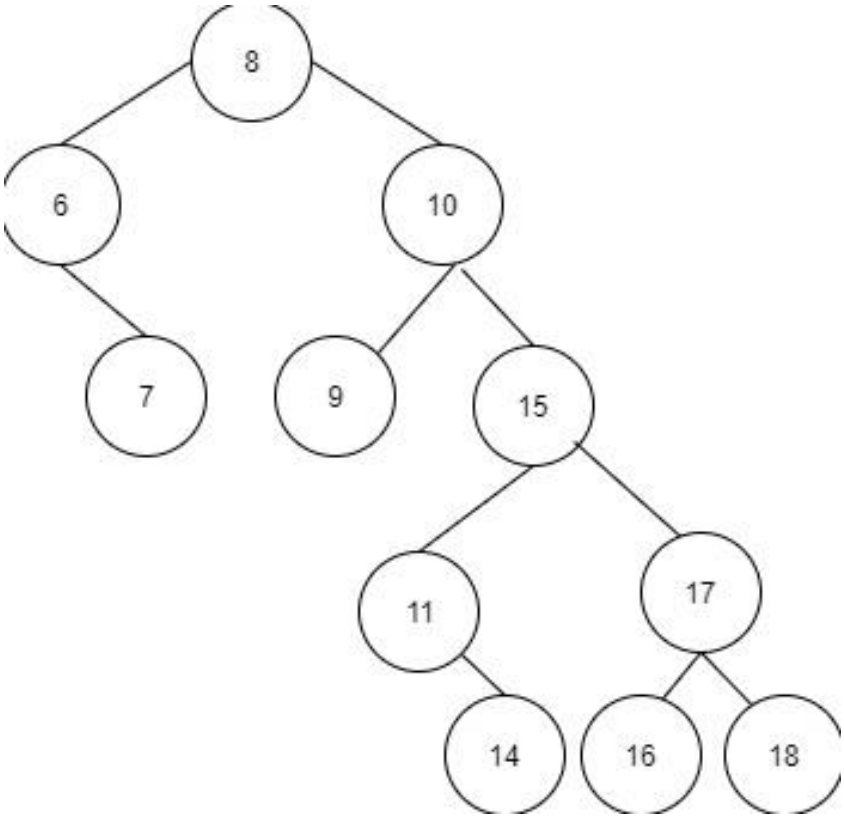
Or

The subtree is calculated by finding the bounding parenthesis, adjusting it by v (the offset of the opening parenthesis) and then dividing by 2 since 2n parentheses represent n nodes.

The parenthesis representation of above modified tree has a subtree in it which is highlighted below.

((( )) (( ) (( ) (( ) ( )))))

b)- I tweaked some of the values in this tree at the bottom to implement the lowest common ancestor function and basically I found the formula its really less understandable from my perspective.



I tried to understand it from this link:

www-erato.ist.hokudai.ac.jp/alsip2011/ALSIP2011_sadakane.pdf

u = **lca(v,w):** common ancestor of **v** and **w** which is furthest from root

u = **parent (RMQ E (v,w)+1)**

- E is the excess array, which represents node depths

m = RMQ E (v,w): the index of a minimum value in **E[v..w] (RMQ = Range Minimum Query)**

If I take **w**=11 and **v**=17

finding out the Lowest Common Ancestor by applying the formula we'll get = **10** as the common ancestor of 11 and 17 which is true actually.