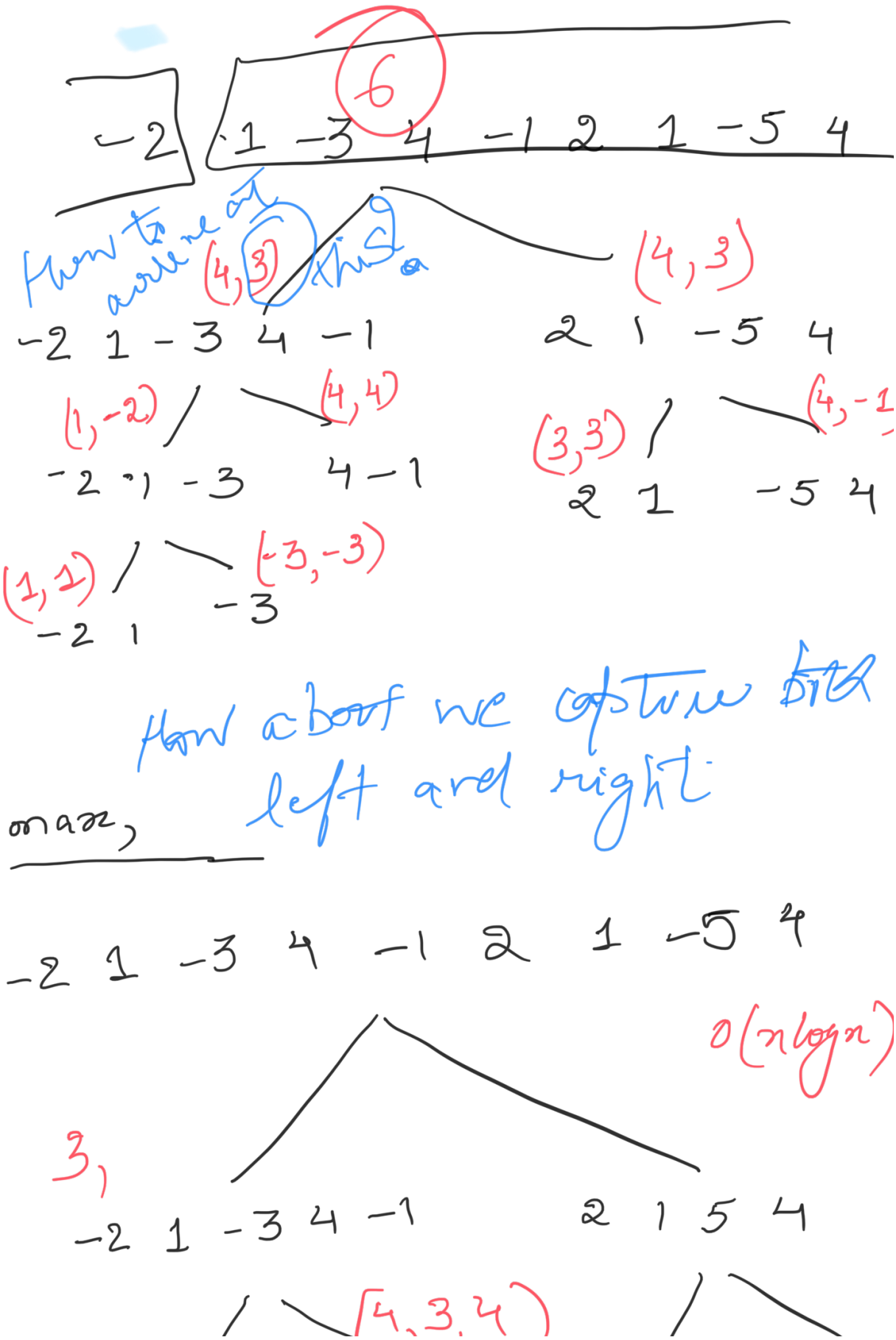
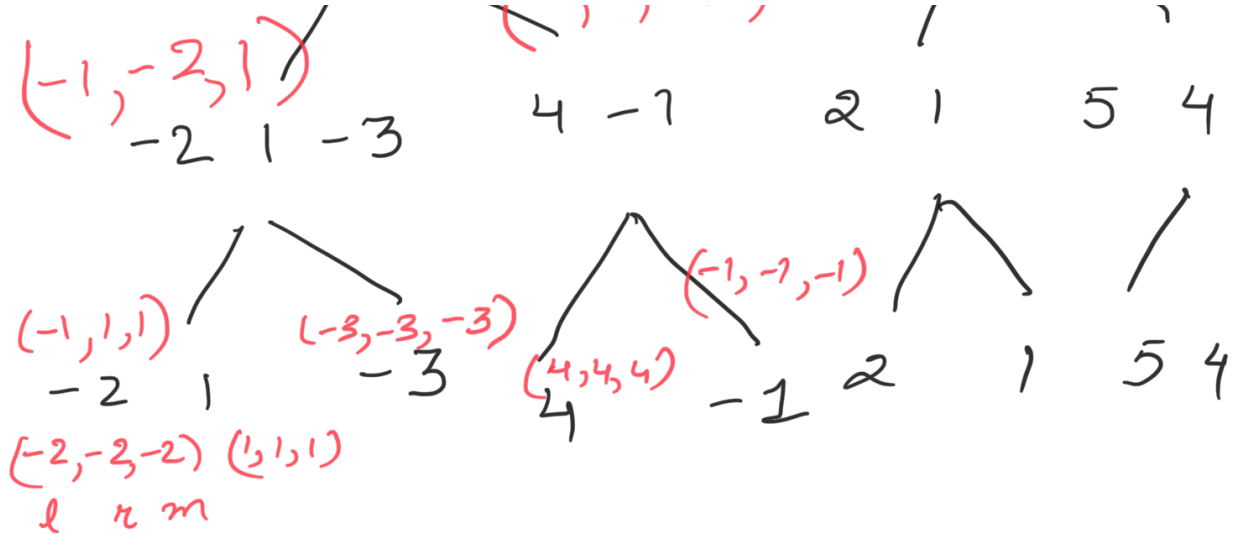
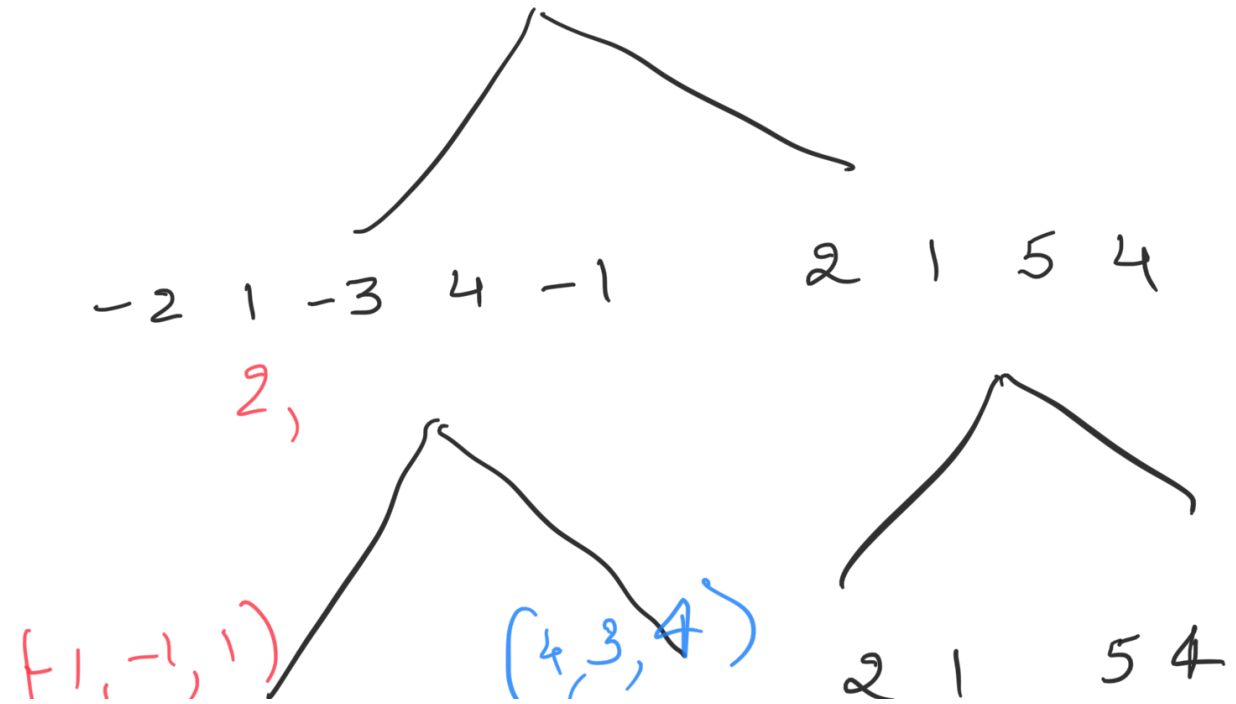

-2 1 -3 4 -1 2 1 -5 4
6
How to arrive at (4,3) this
(4,3)
-2 1 -3 4 -1
2 1 -5 4
(1,-2)
(4,4)
(3,3)
(4,-1
-2 1 -3 4 -1
2 1 -5 4
(1,1)
(-3,-3)
-2 1
-3
How about we store left and right
max,
-2 1 -3 4 -1 2 1 -5 4
O(nlogn)
3,
-2 1 -3 4 -1
2 1 5 4
(4,3,4)

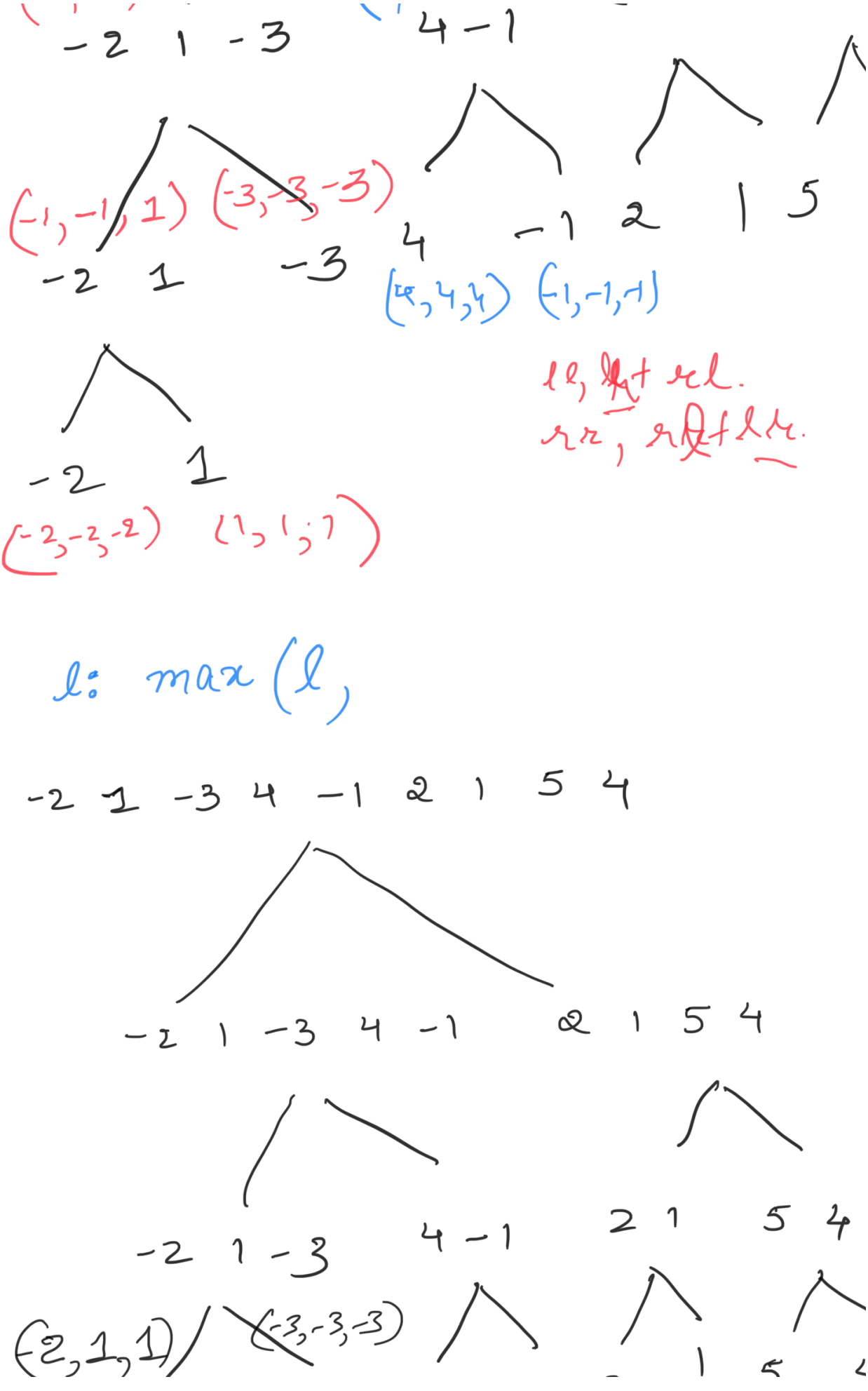


$$\max\left((-2+1),(1),1\right)$$

Left: $\max(ll, ll+rl)$

Right: $\max(rr, lr+rr)$

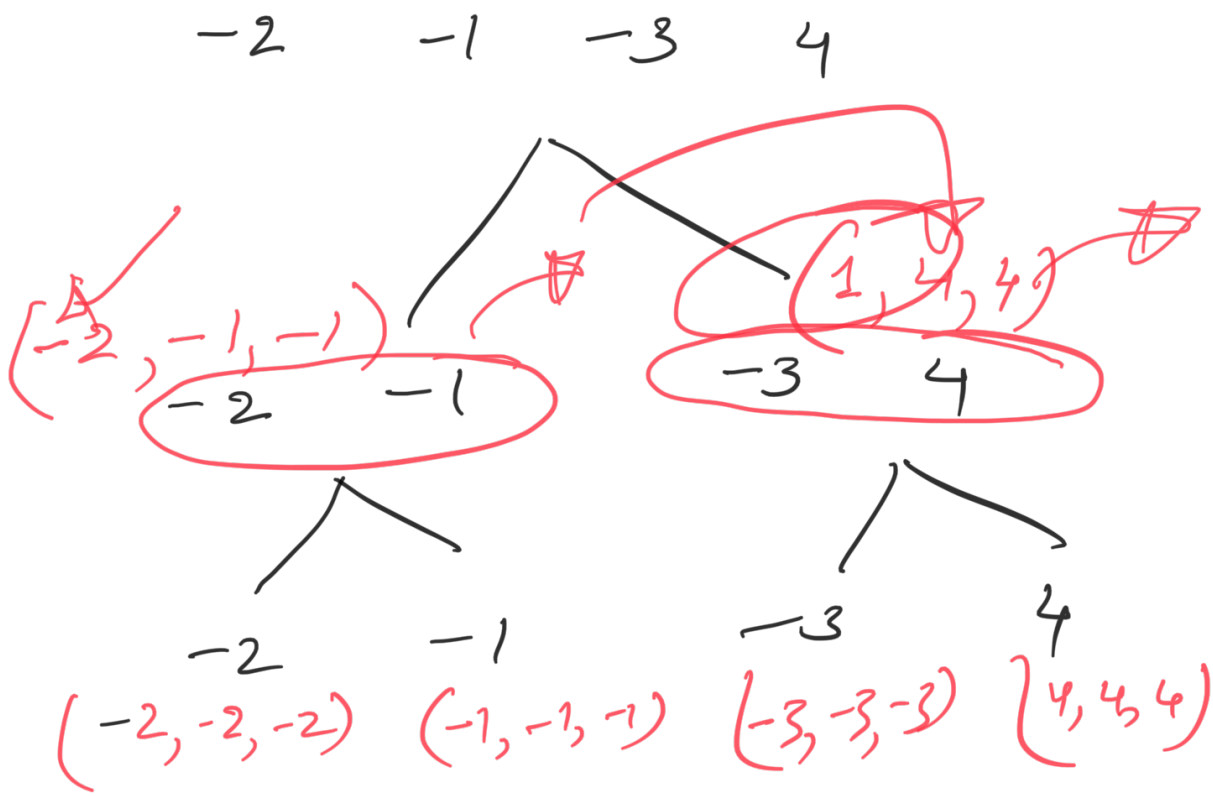


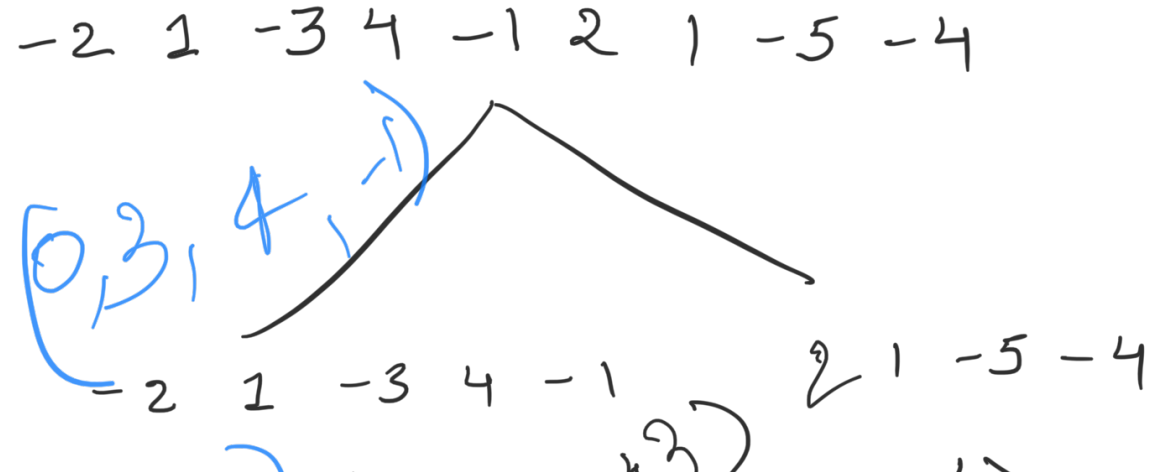

-2 1 -3
4 -1
(-1,-1,1) (-3,-3,-3)
-2 1 -3
4 -1 2 1 5
(4,4,4) (-1,-1,-1)
ll, lr, rl.
rr, rl, lr.
-2 1
(-2,-2,-2) (1,1,1)
l: max(l,
-2 1 -3 4 -1 2 1 5 4
-2 1 -3 4 -1 2 1 5 4
-2 1 -3
4 -1
(2,1,1) (-3,-3,-3)
2 1 5 4
1 5

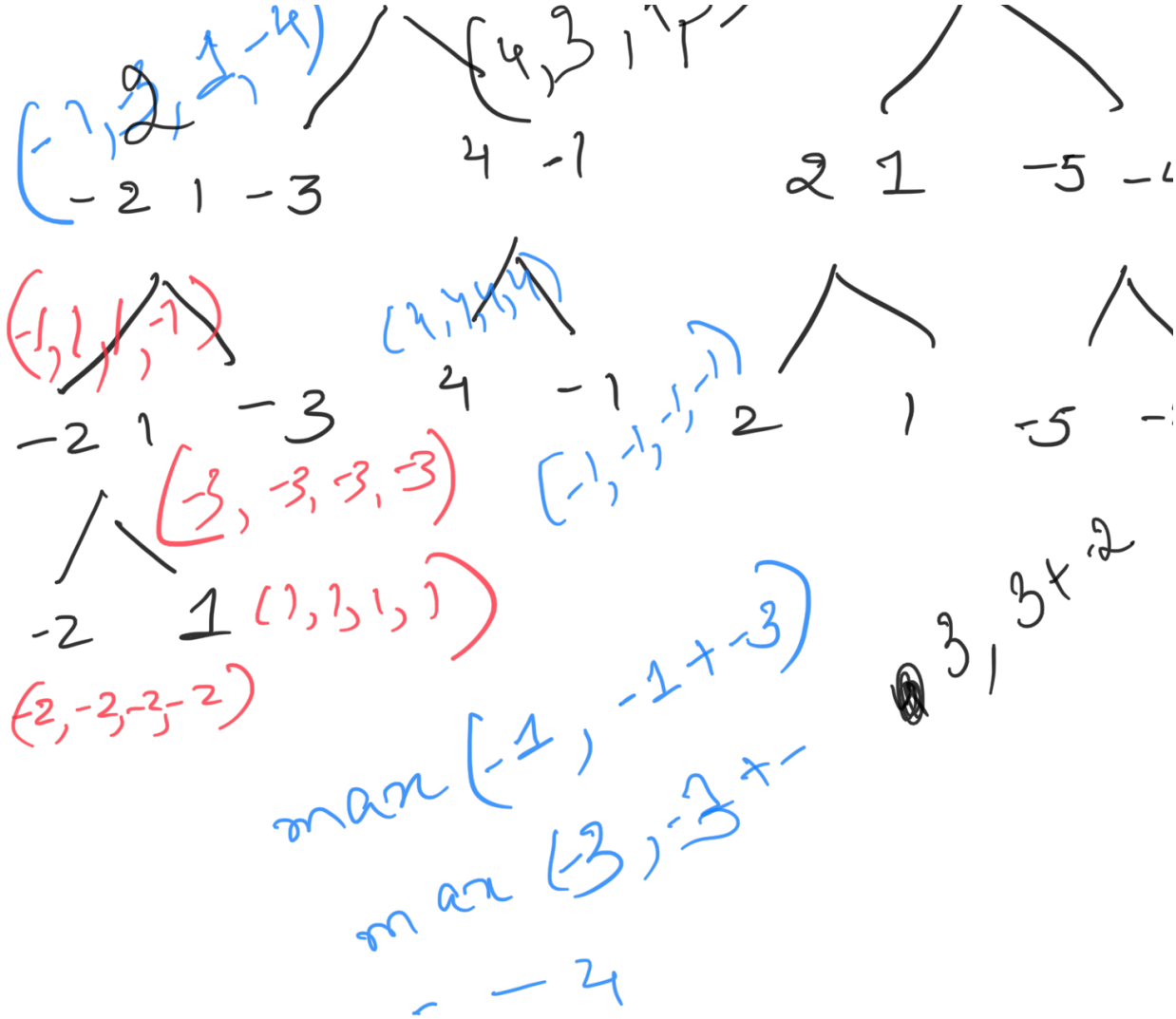
-2 1 -3 4 -1 2

(-2,-2,-2) (1,1,1) (3,-3,-3)

How







Pseudo Code

```
def max_subarray(l, left, right):
    if left == right:
        return (l[left], l[left], l[left], l[left])
    n = len(l)
    mid = (left + right)//2
    lr = max_subarray(l, left, mid)
    rr = max_subarray(l, mid+1, right)
    max_left = max(lr[0], lr[3] + rr[0])
    max_right = max(rr[1], rr[3] + lr[1])
```

```
man_sum = max(max(lr[0], rr[0]),
              lr[2], rr[2], lr[3]+rr[3]
                                  lr[1]+rr[
return (man_left, man_right, man_sum,
        lr[3]+rr[3])
```