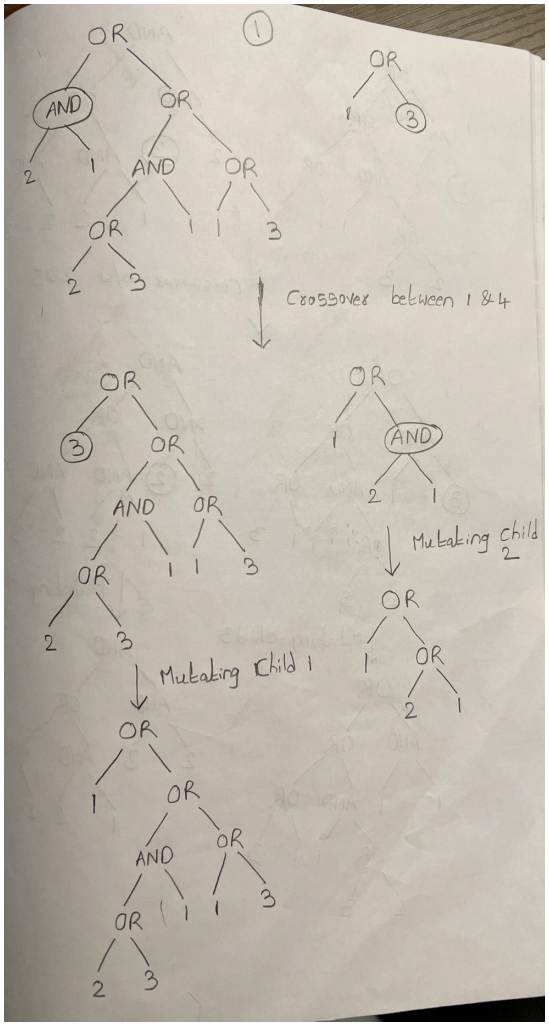
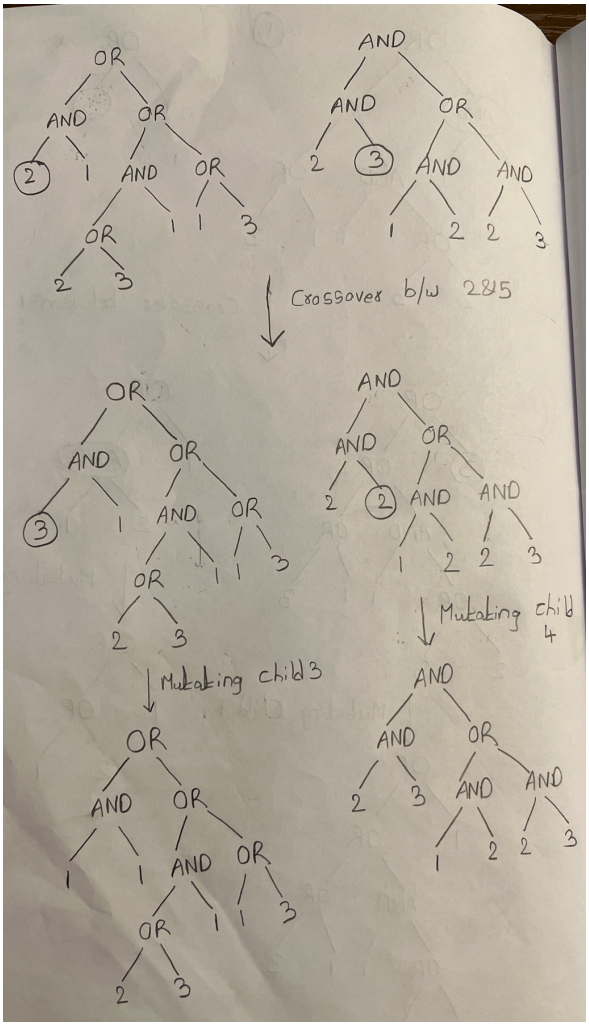
Team Members:

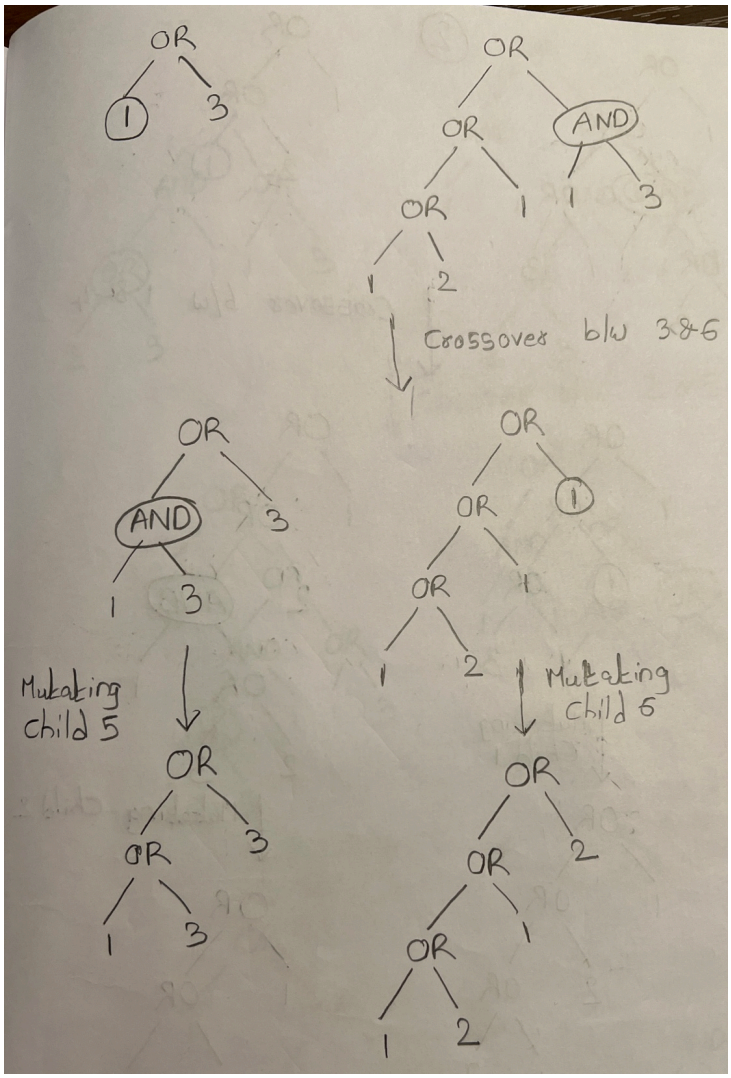
Shanmukha Datta Sai Sasank Tadepalli - NetID: na2500
Bhavana Kamasani - NetID: tl1593
Neeraj Kumar Maraka - NetID: ho4913

1. Show each of the first 10 generations of this genetic programming experiment. It's probably easiest to draw the trees for each chromosome if you're not fluent in Lisp.

Generation 0:
1: (OR (AND 2 1) (OR (AND (OR 2 3) 1) (OR 1 3))) - Fitness = 6
2: (OR 1 3) - Fitness = 6
3: (AND (AND 2 3) (OR (AND 1 2) (AND 2 3))) - Fitness = 6
4: (OR (OR (OR 1 2) 1) (AND 1 3)) - Fitness = 6
5: (AND (AND 1 3) 2) - Fitness = 5
6: (OR (AND 1 1) (OR 3 2)) - Fitness - 5
Selection:
1: (OR (AND 2 1) (OR (AND (OR 2 3) 1) (OR 1 3))) - Fitness = 6
2: (OR (AND 2 1) (OR (AND (OR 2 3) 1) (OR 1 3))) - Fitness = 6
3: (OR 1 3) - Fitness = 6
4: (OR 1 3) - Fitness = 6
5: (AND (AND 2 3) (OR (AND 1 2) (AND 2 3))) - Fitness = 6
6: (OR (OR (OR 1 2) 1) (AND 1 3)) - Fitness = 6
Crossover & Mutation:



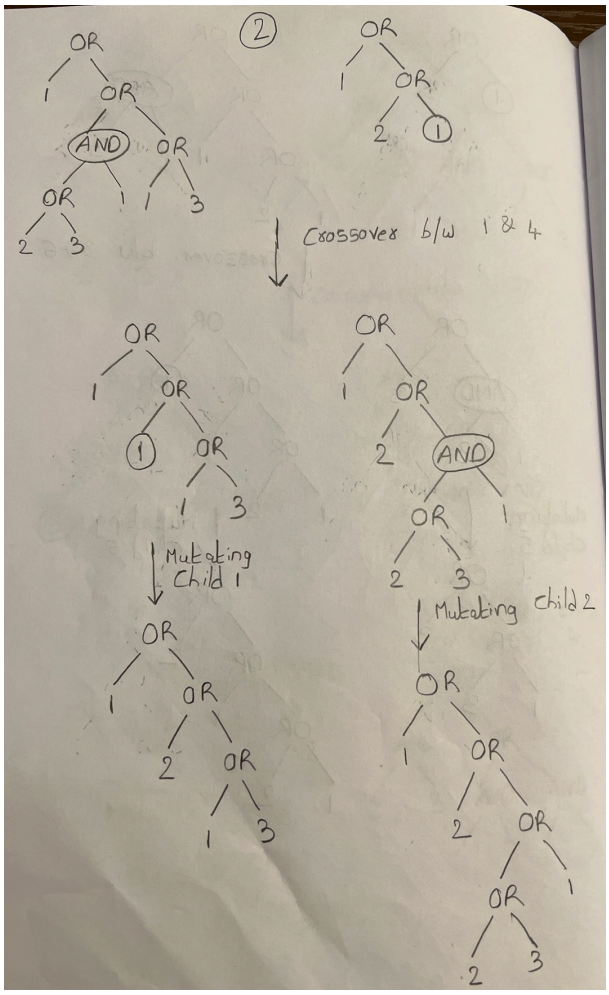


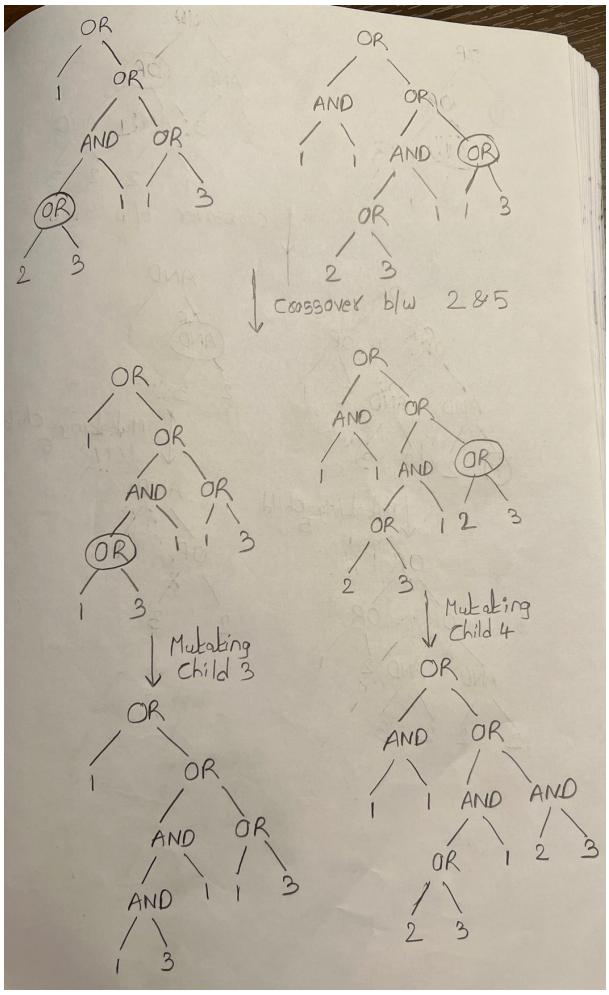


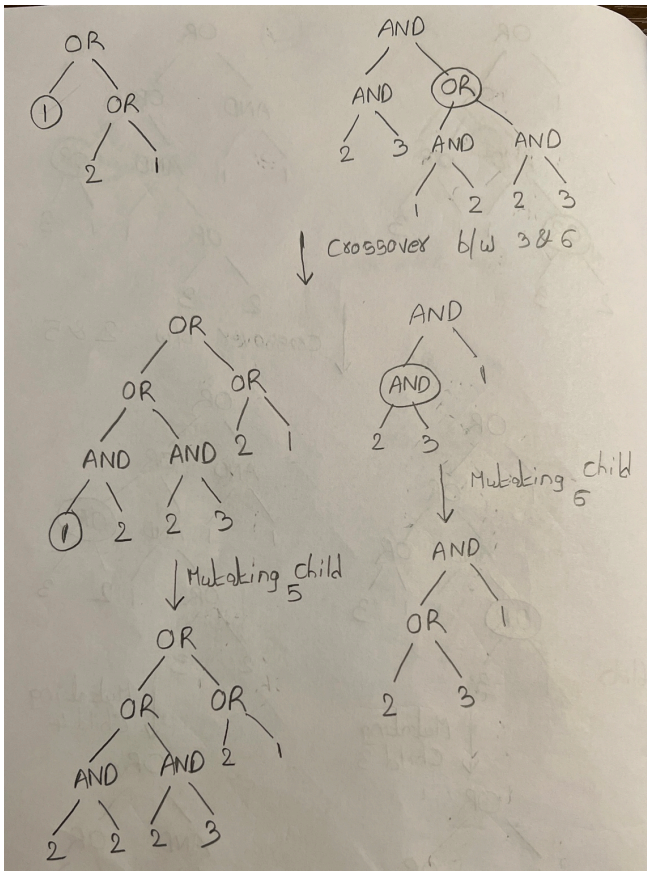
Generation 1:
1: (OR 1 (OR (AND (OR 2 3) 1) (OR 1 3))) - Fitness Score = 6
2: (OR 1 (OR 2 1)) - Fitness Score = 6
3: (OR (AND 1 1) (OR (AND (OR 2 3) 1) (OR 1 3))) - Fitness Score = 6
4: (AND (AND 2 3) (OR (AND 1 2) (AND 2 3))) - Fitness Score = 6
5: (OR (OR 1 3) 3) - Fitness Score = 6
6: (OR (OR (OR 1 2) 1) 2) - Fitness Score = 6

Selection:
1: (OR 1 (OR (AND (OR 2 3) 1) (OR 1 3))) - Fitness Score = 6
2: (OR 1 (OR (AND (OR 2 3) 1) (OR 1 3))) - Fitness Score = 6
3: (OR 1 (OR 2 1)) - Fitness Score = 6
4: (OR 1 (OR 2 1)) - Fitness Score = 6
5: (OR (AND 1 1) (OR (AND (OR 2 3) 1) (OR 1 3))) - Fitness Score = 6
6: (AND (AND 2 3) (OR (AND 1 2) (AND 2 3))) - Fitness Score = 6

Crossover & Mutation:



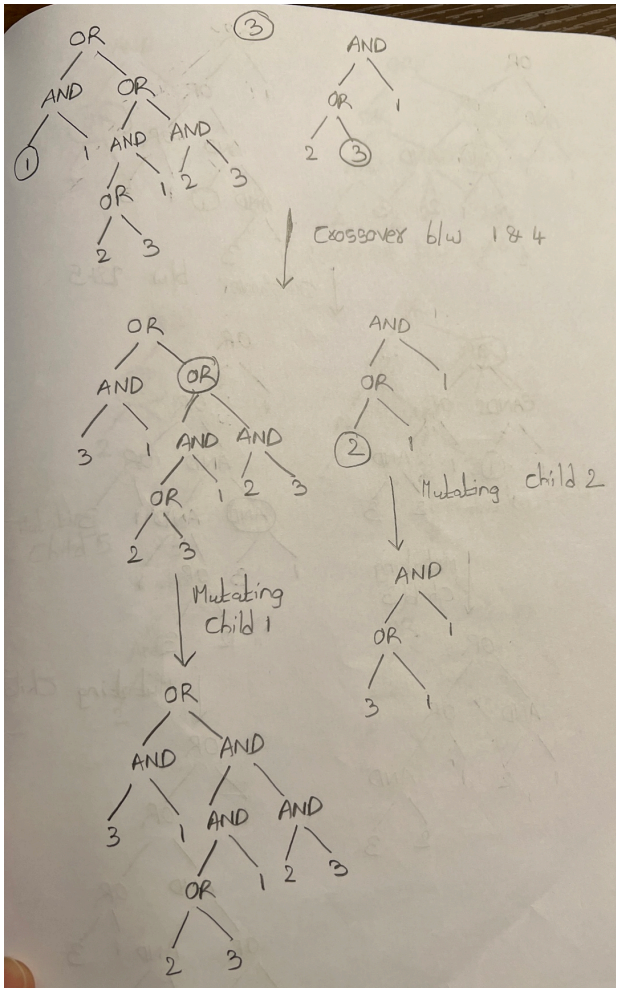


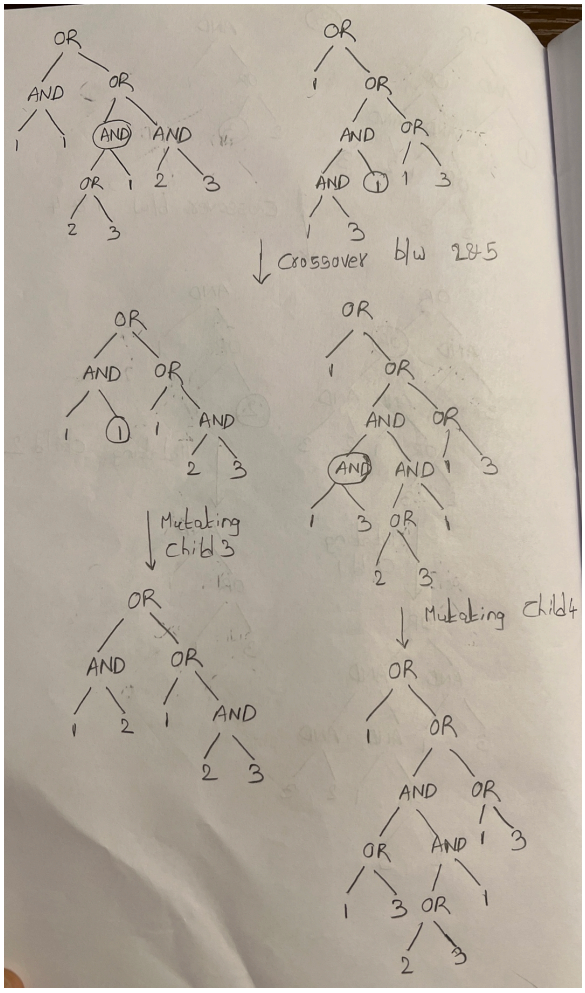


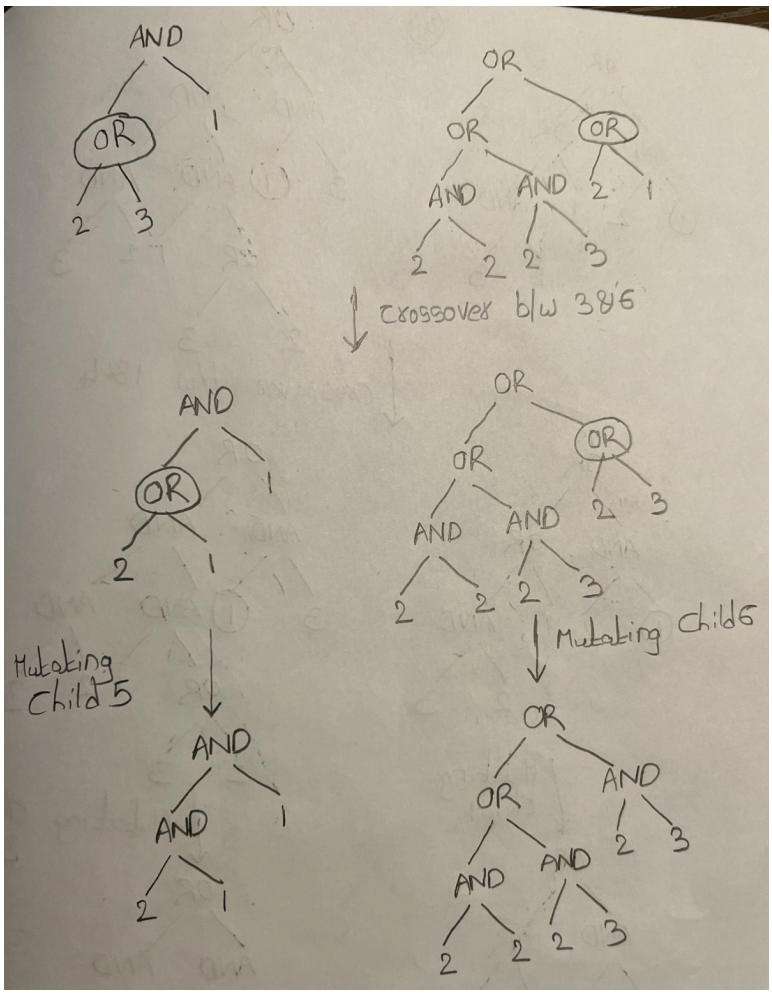
Generation 2:
1: (OR 1 (OR 2 (OR 1 3))) - Fitness Score = 5
2: (OR 1 (OR 2 (OR (OR 2 3) 1))) - Fitness Score = 5
3: (OR 1 (OR (AND (AND 1 3) 1) (OR 1 3))) - Fitness Score = 6
4: (OR (AND 1 1) (OR (AND (OR 2 3) 1) (AND 2 3))) - Fitness Score = 7
5: (OR (OR (AND 2 2) (AND 2 3)) (OR 2 1)) - Fitness Score = 6
6: (AND (OR 2 3) 1) - Fitness Score = 7

Selection:
1: (OR (AND 1 1) (OR (AND (OR 2 3) 1) (AND 2 3))) - Fitness Score = 7
2: (OR (AND 1 1) (OR (AND (OR 2 3) 1) (AND 2 3))) - Fitness Score = 7
3: (AND (OR 2 3) 1) - Fitness Score = 7
4: (AND (OR 2 3) 1) - Fitness Score = 7
5: (OR 1 (OR (AND (AND 1 3) 1) (OR 1 3))) - Fitness Score = 6
6: (OR (OR (AND 2 2) (AND 2 3)) (OR 2 1)) - Fitness Score = 6

Crossover & Mutation:







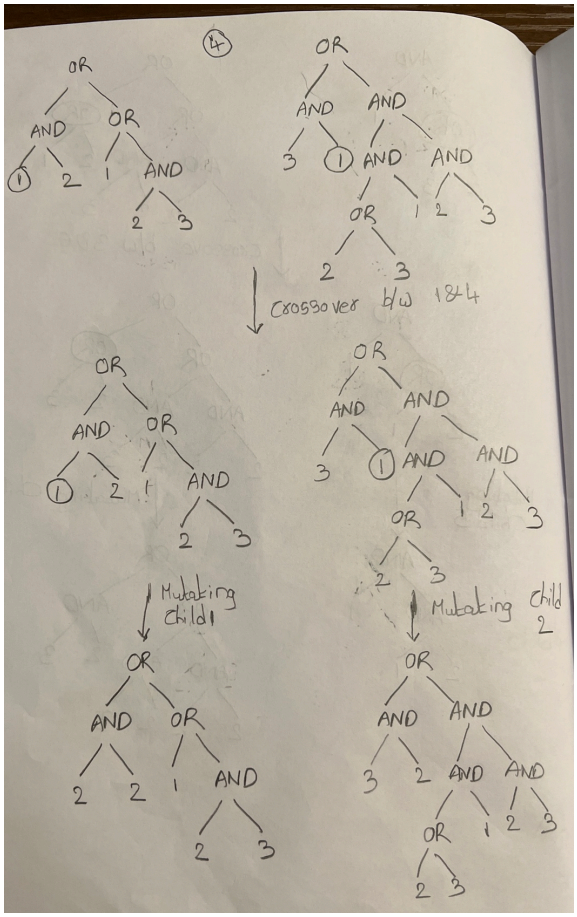
Generation 3:

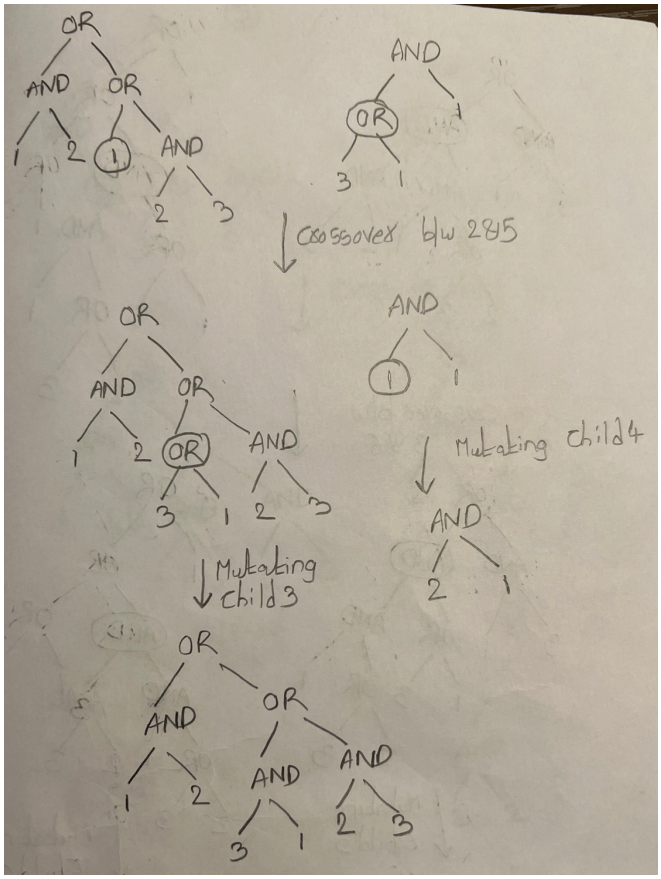
1: (OR (AND 3 1) (AND (AND (OR 2 3) 1) (AND 2 3))) - 6
2: (AND (OR 3 1) 1) - 6
3: (OR (AND 1 2) (OR 1 (AND 2 3))) - 7
4: (OR 1 (OR (AND (OR 1 3) (AND (OR 2 3) 1)) (OR 1 3))) - 6
5: (AND (AND 2 1) 1) - 6
6: (OR (OR (AND 2 2) (AND 2 3)) (AND 2 3)) - 6

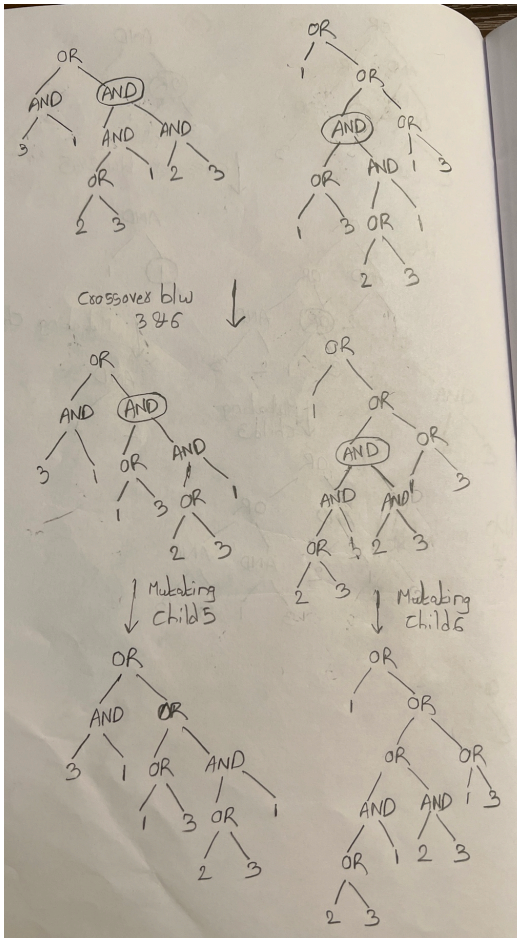
Selection:

1: (OR (AND 1 2) (OR 1 (AND 2 3))) - 7
2: (OR (AND 1 2) (OR 1 (AND 2 3))) - 7
3: (OR (AND 3 1) (AND (AND (OR 2 3) 1) (AND 2 3))) - 6
4: (OR (AND 3 1) (AND (AND (OR 2 3) 1) (AND 2 3))) - 6
5: (AND (OR 3 1) 1) - 6
6: (OR 1 (OR (AND (OR 1 3) (AND (OR 2 3) 1)) (OR 1 3))) - 6

Crossover & Mutation:



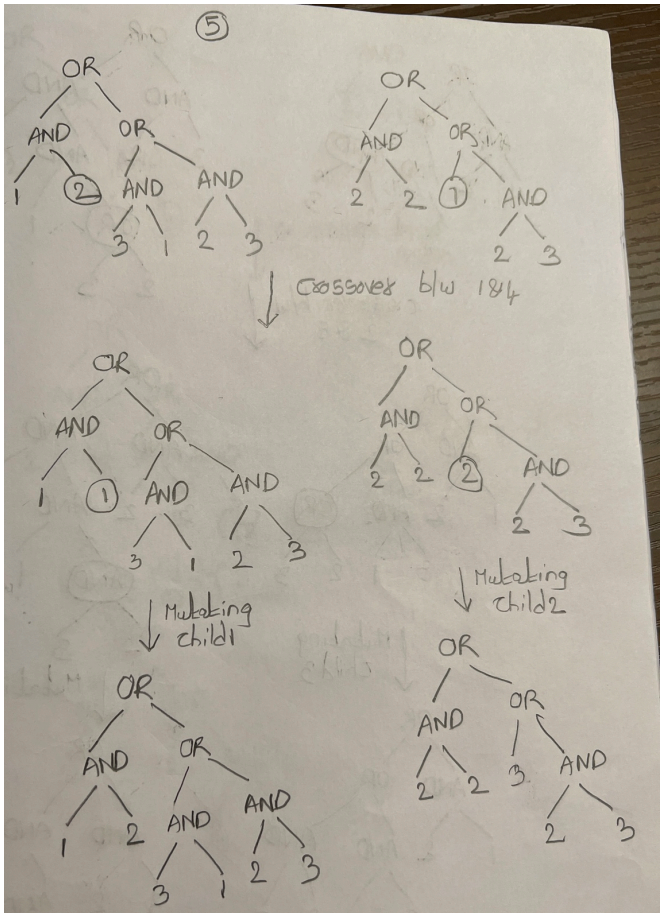


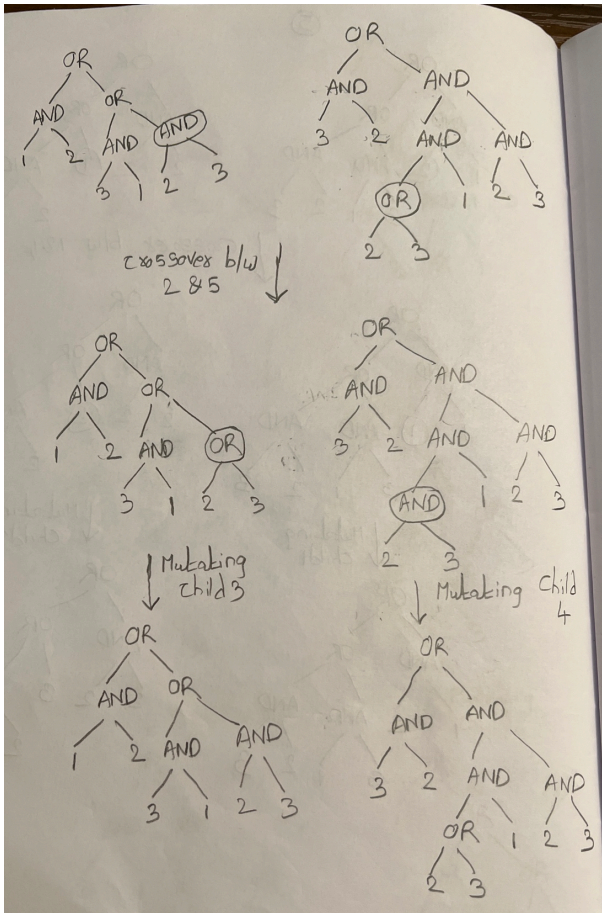


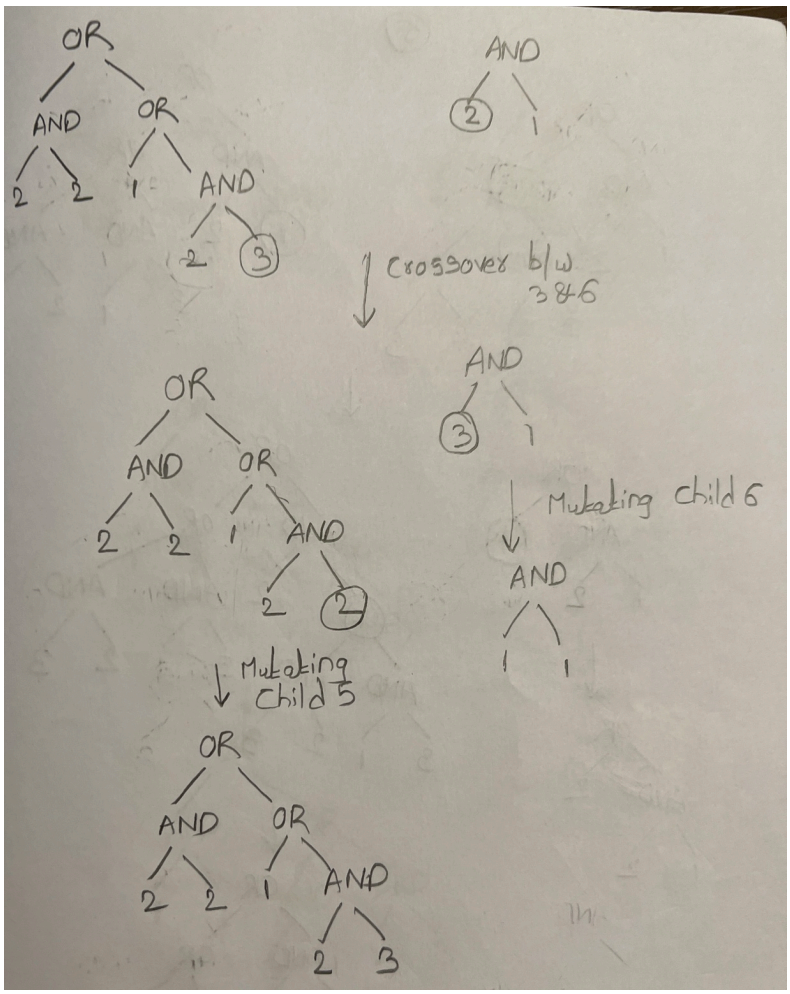
Generation 4:
1: (OR (AND 2 2) (OR 1 (AND 2 3))) - 6
2: (OR (AND 3 2) (AND (AND (OR 2 3) 1) (AND 2 3))) - 6
3: (OR (AND 1 2) (OR (AND 3 1) (AND 2 3))) - 8
4: (AND 2 1) - 6
5: (OR (AND 3 1) (OR (OR 1 3) (AND (OR 2 3) 1))) - 6
6: (OR 1 (OR (OR (AND (OR 2 3) 1) (AND 2 3)) (OR 1 3))) - 6

Selection:
1: (OR (AND 1 2) (OR (AND 3 1) (AND 2 3))) - 8
2: (OR (AND 1 2) (OR (AND 3 1) (AND 2 3))) - 8
3: (OR (AND 2 2) (OR 1 (AND 2 3))) - 6
4: (OR (AND 2 2) (OR 1 (AND 2 3))) - 6
5: (OR (AND 3 2) (AND (AND (OR 2 3) 1) (AND 2 3))) - 6
6: (AND 2 1) - 6

Crossover & Mutation:







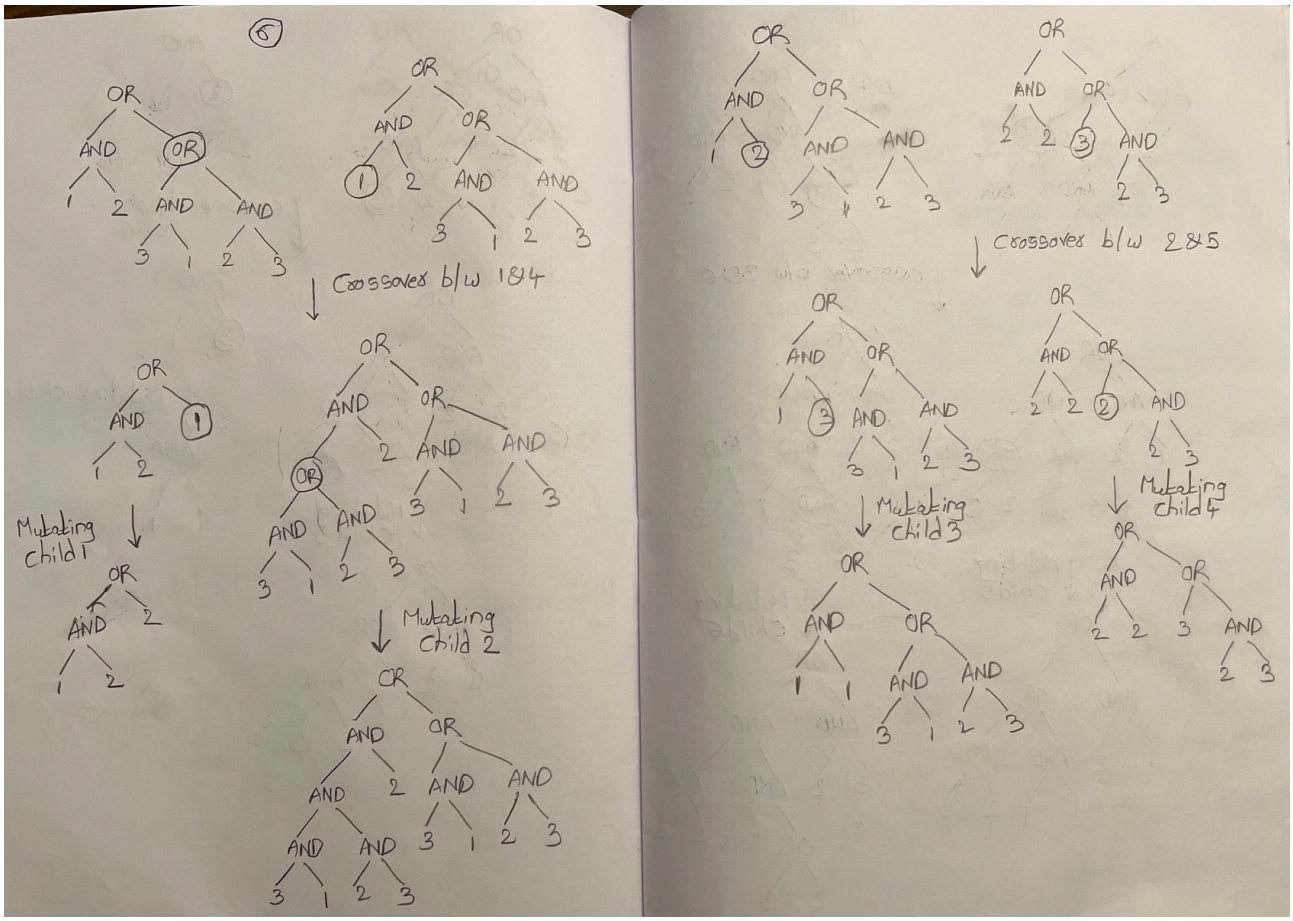
Generation 5:

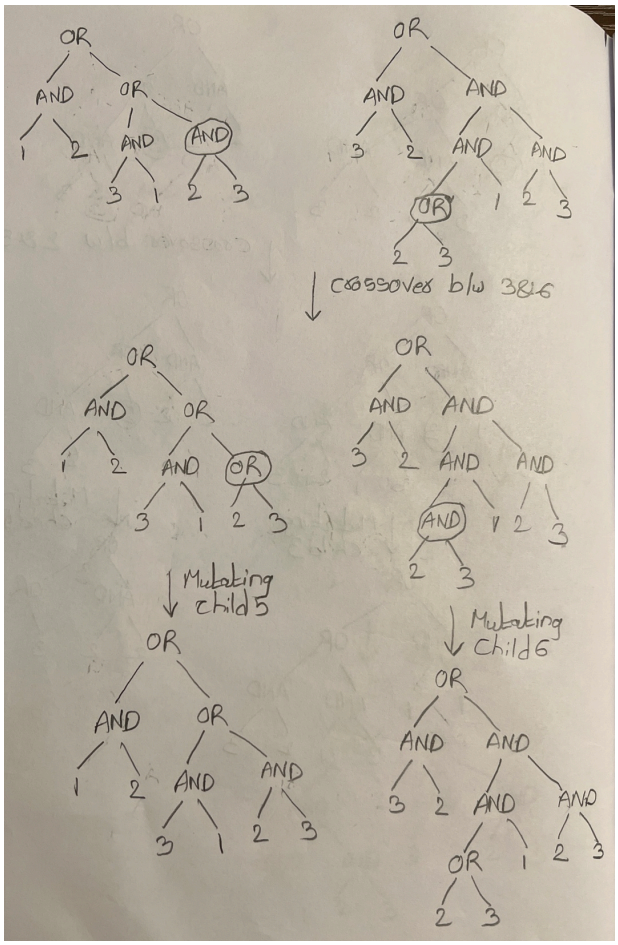
1: (OR (AND 1 2) (OR (AND 3 1) (AND 2 3))) - 8
2: (OR (AND 2 2) (OR 3 (AND 2 3))) - 6
3: (OR (AND 1 2) (OR (AND 3 1) (AND 2 3))) - 8
4: (OR (AND 3 2) (AND (AND (OR 2 3) 1) (AND 2 3))) - 6
5: (OR (AND 2 2) (OR 1 (AND 2 3))) - 6
6: (AND 1 1) - 6

Selection:

1: (OR (AND 1 2) (OR (AND 3 1) (AND 2 3))) - 8
2: (OR (AND 1 2) (OR (AND 3 1) (AND 2 3))) - 8
3: (OR (AND 1 2) (OR (AND 3 1) (AND 2 3))) - 8
4: (OR (AND 1 2) (OR (AND 3 1) (AND 2 3))) - 8
5: (OR (AND 2 2) (OR 3 (AND 2 3))) - 6
6: (OR (AND 3 2) (AND (AND (OR 2 3) 1) (AND 2 3))) - 6

Crossover & Mutation:

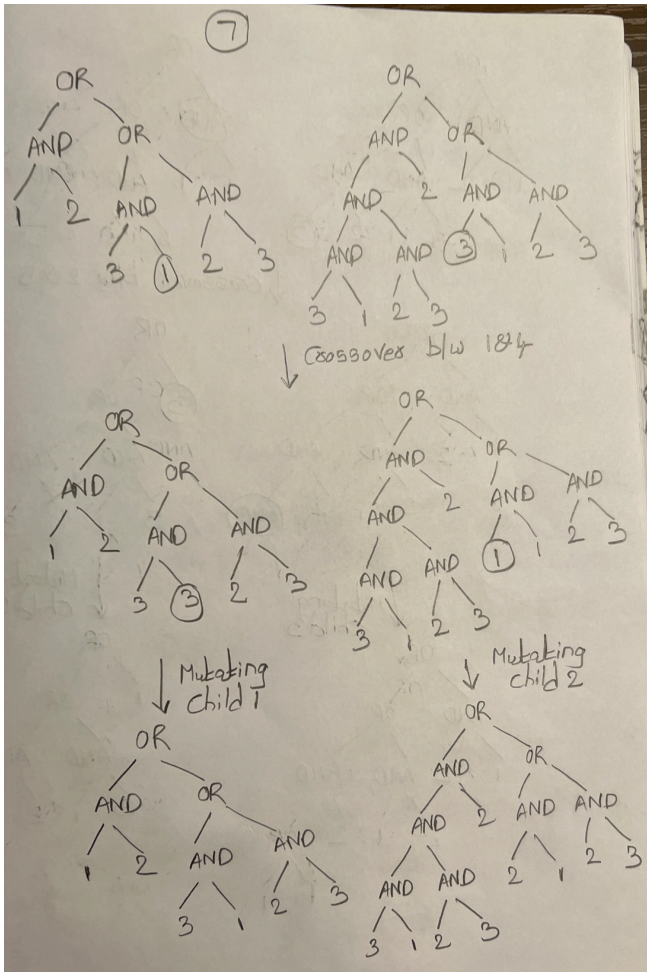


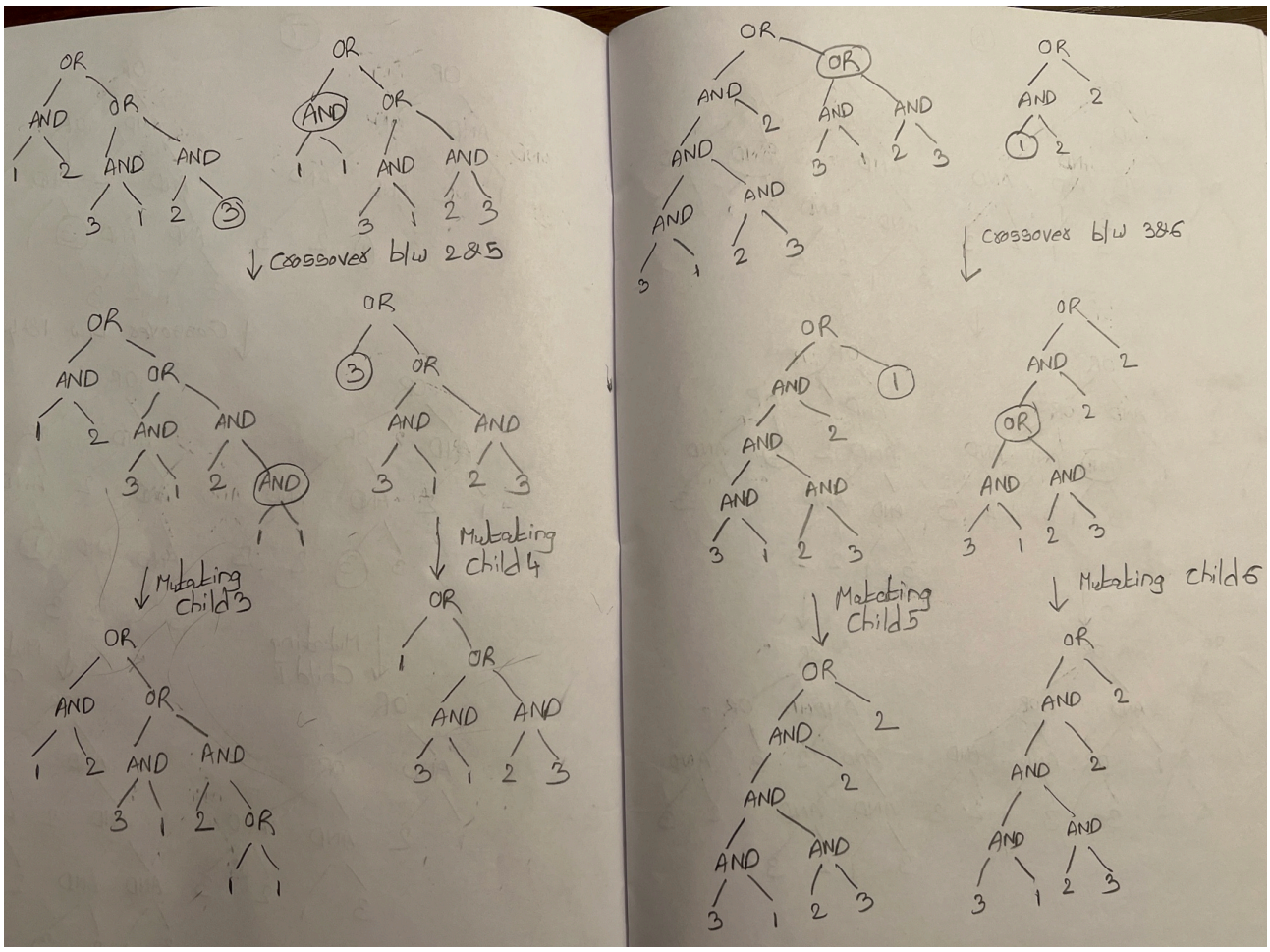


Generation 6:
1: (OR (AND 1 2) 2) - 6
2: (OR (AND (AND (AND 3 1) (AND 2 3)) 2) (OR (AND 3 1) (AND 2 3))) - 7
3: (OR (AND 1 1) (OR (AND 3 1) (AND 2 3))) - 7
4: (OR (AND 2 2) (OR 3 (AND 2 3))) - 6
5: (OR (AND 1 2) (OR (AND 3 1) (AND 2 3))) - 8
6: (OR (AND 3 2) (AND (AND (OR 2 3) 1) (AND 2 3))) - 6

Selection:
1: (OR (AND 1 2) (OR (AND 3 1) (AND 2 3))) - 8
2: (OR (AND 1 2) (OR (AND 3 1) (AND 2 3))) - 8
3: (OR (AND (AND (AND 3 1) (AND 2 3)) 2) (OR (AND 3 1) (AND 2 3))) - 7
4: (OR (AND (AND (AND 3 1) (AND 2 3)) 2) (OR (AND 3 1) (AND 2 3))) - 7
5: (OR (AND 1 1) (OR (AND 3 1) (AND 2 3))) - 7
6: (OR (AND 1 2) 2) - 6

Crossover & Mutation:

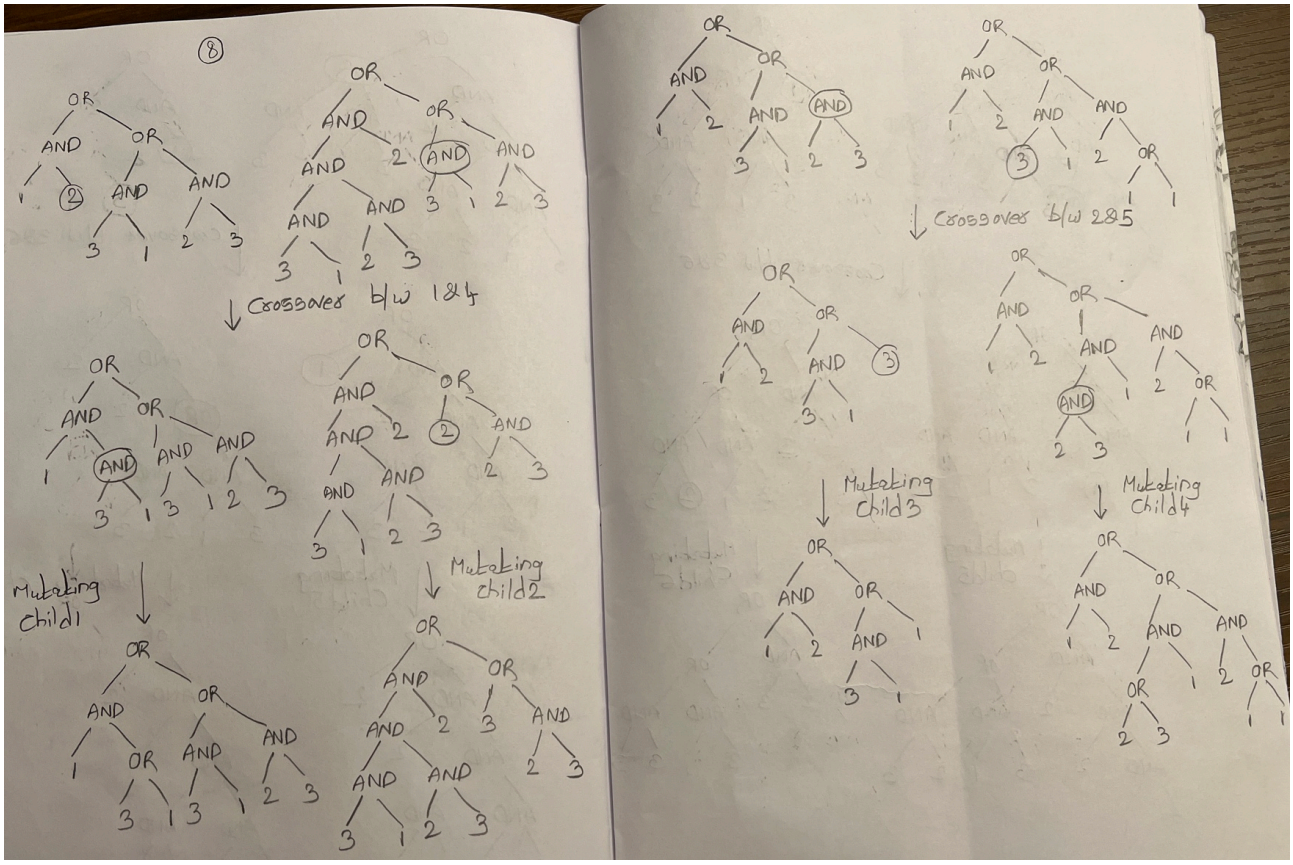


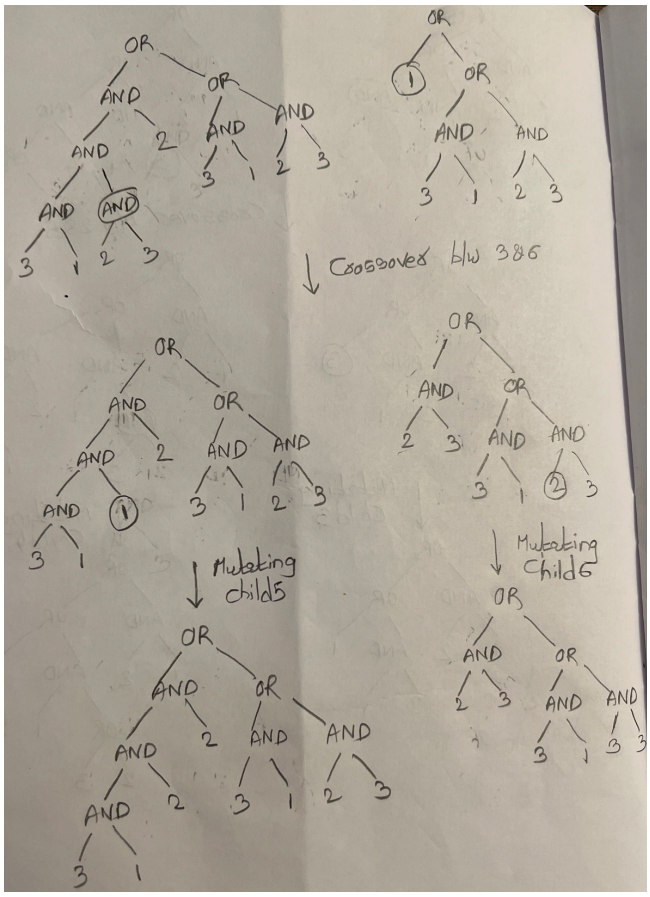


Generation 7:
1: (OR (AND 1 2) (OR (AND 3 1) (AND 2 3))) - 8
2: (OR (AND (AND (AND 3 1) (AND 2 3)) 2) (OR (AND 3 1) (AND 2 3))) - 7
3: (OR (AND 1 2) (OR (AND 3 1) (AND 2 (OR 1 1)))) - 7
4: (OR 1 (OR (AND 3 1) (AND 2 3))) - 7
5: (OR (AND (AND (AND 3 1) (AND 2 3)) 2) 2) - 6
6: (OR (AND (AND (AND 3 1) (AND 2 3)) 2) 2) - 6

Selection:
1: (OR (AND 1 2) (OR (AND 3 1) (AND 2 3))) - 8
2: (OR (AND 1 2) (OR (AND 3 1) (AND 2 3))) - 8
3: (OR (AND (AND (AND 3 1) (AND 2 3)) 2) (OR (AND 3 1) (AND 2 3))) - 7
4: (OR (AND (AND (AND 3 1) (AND 2 3)) 2) (OR (AND 3 1) (AND 2 3))) - 7
5: (OR (AND 1 2) (OR (AND 3 1) (AND 2 (OR 1 1)))) - 7
6: (OR 1 (OR (AND 3 1) (AND 2 3))) - 7

Crossover & Mutation:

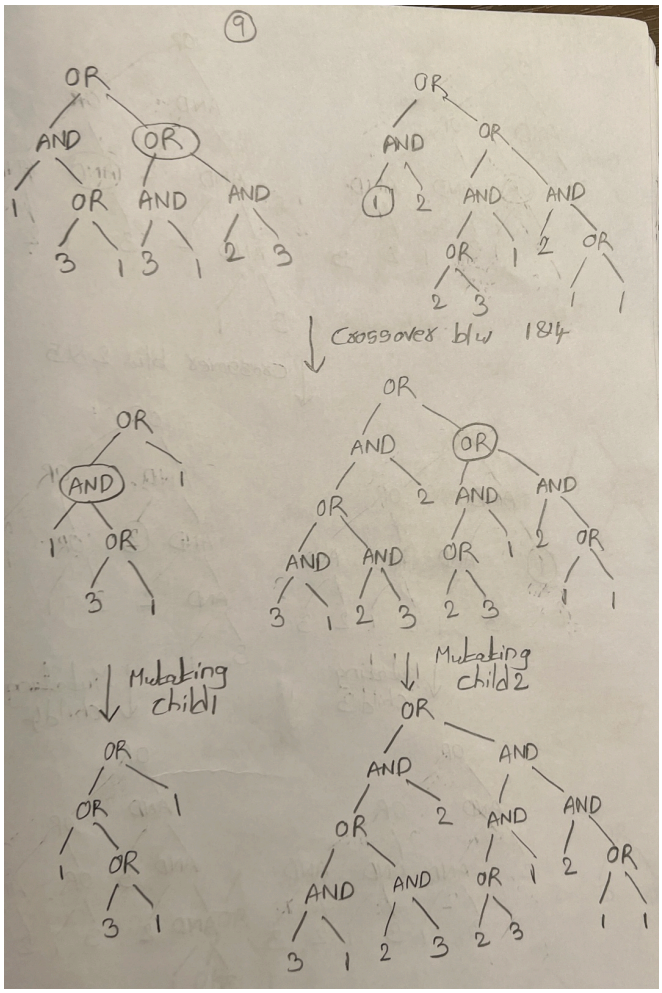


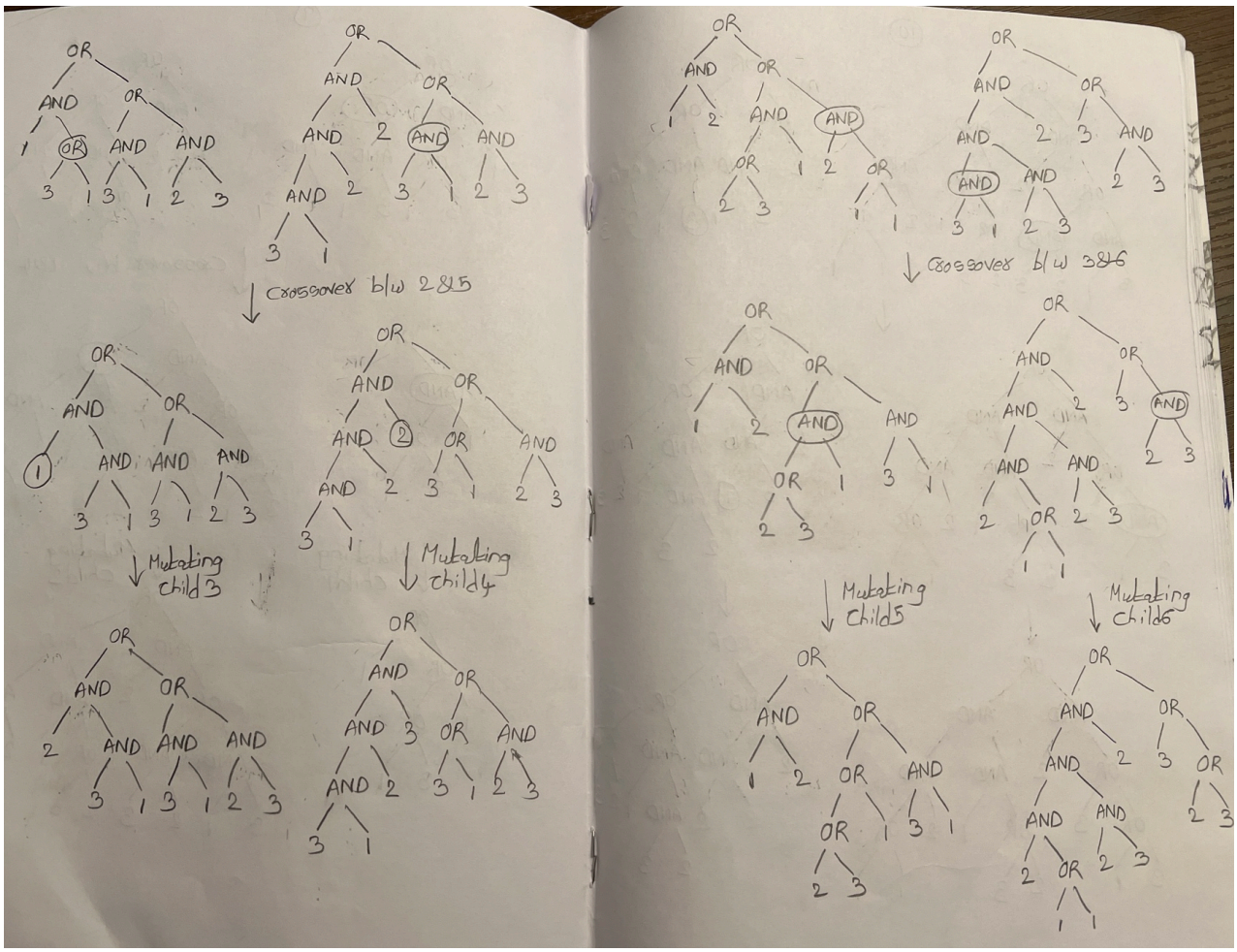


Generation 8:
1: (OR (AND 1 (OR 3 1)) (OR (AND 3 1) (AND 2 3))) - 7
2: (OR (AND (AND (AND 3 1) (AND 2 3)) 2) (OR 3 (AND 2 3))) - 6
3: (OR (AND 1 2) (OR (AND 3 1) 1)) - 6
4: (OR (AND 1 2) (OR (AND (OR 2 3) 1) (AND 2 (OR 1 1)))) - 7
5: (OR (AND (AND (AND 3 1) 2) 2) (OR (AND 3 1) (AND 2 3))) - 7
6: (OR (AND 2 3) (OR (AND 3 1) (AND 3 3))) - 6

Selection:
1: (OR (AND 1 (OR 3 1)) (OR (AND 3 1) (AND 2 3))) - 7
2: (OR (AND 1 (OR 3 1)) (OR (AND 3 1) (AND 2 3))) - 7
3: (OR (AND 1 2) (OR (AND (OR 2 3) 1) (AND 2 (OR 1 1)))) - 7
4: (OR (AND 1 2) (OR (AND (OR 2 3) 1) (AND 2 (OR 1 1)))) - 7
5: (OR (AND (AND (AND 3 1) 2) 2) (OR (AND 3 1) (AND 2 3))) - 7
6: (OR (AND (AND (AND 3 1) (AND 2 3)) 2) (OR 3 (AND 2 3))) - 6

Crossover & Mutation:

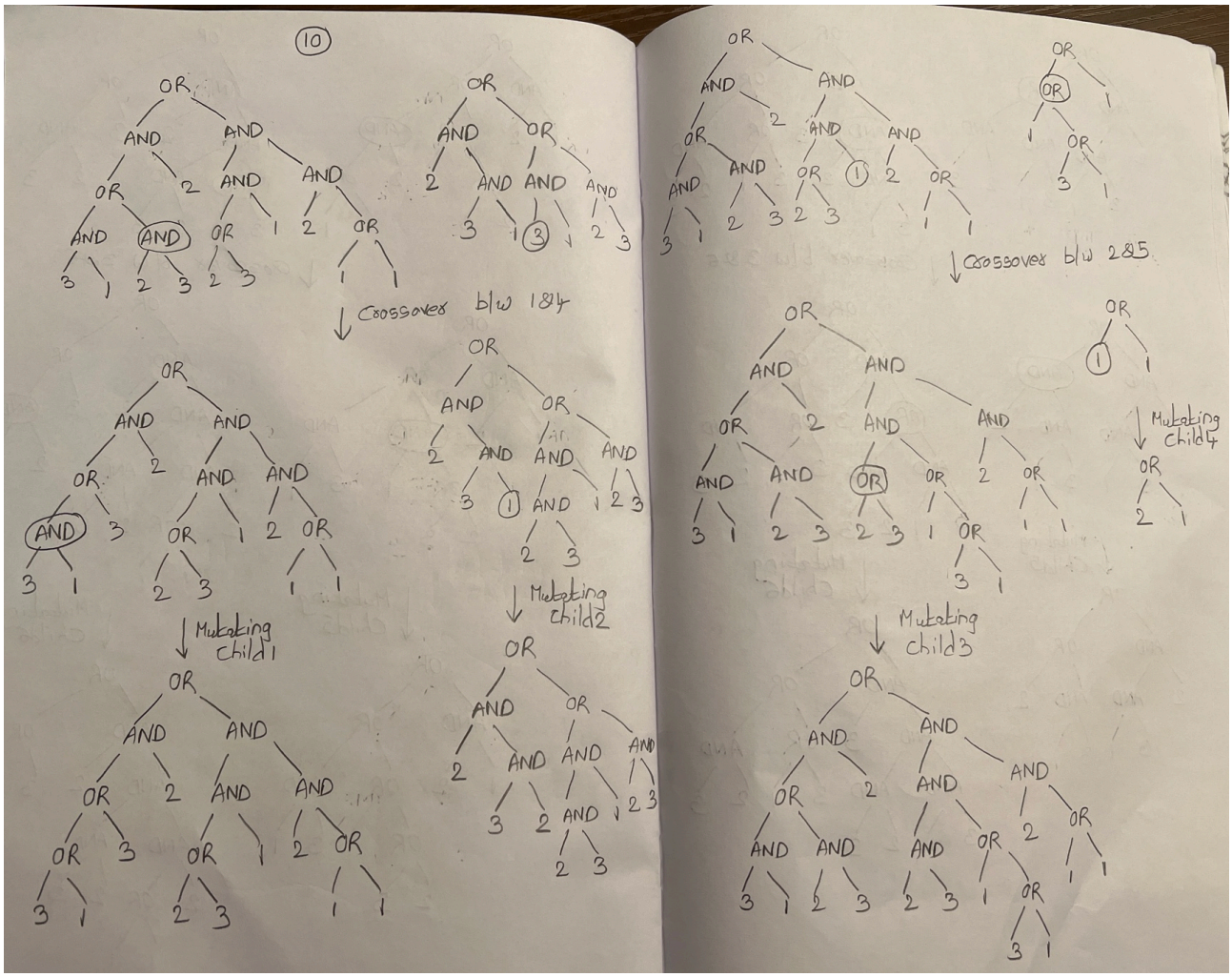


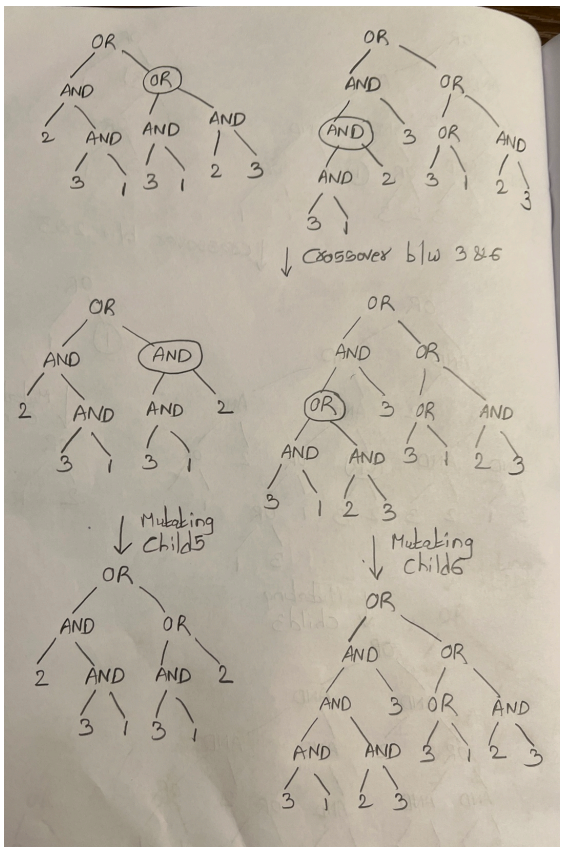


Generation 9:
1: (OR (OR 1 (OR 3 1)) 1) - 6
2: (OR (AND (OR (AND 3 1) (AND 2 3)) 2) (AND (AND (OR 2 3) 1) (AND 2 (OR 1 1)))) - 7
3: (OR (AND 2 (AND 3 1)) (OR (AND 3 1) (AND 2 3))) - 7
4: (OR (AND (AND (AND 3 1) 2) 3) (OR (OR 3 1) (AND 2 3))) - 6
5: (OR (AND 1 2) (OR (OR (OR 2 3) 1) (AND 3 1))) - 5
6: (OR (AND (AND (AND 2 (OR 1 1)) (AND 2 3)) 2) (OR 3 (OR 2 3))) - 6

Selection:
1: (OR (AND (OR (AND 3 1) (AND 2 3)) 2) (AND (AND (OR 2 3) 1) (AND 2 (OR 1 1)))) - 7
2: (OR (AND (OR (AND 3 1) (AND 2 3)) 2) (AND (AND (OR 2 3) 1) (AND 2 (OR 1 1)))) - 7
3: (OR (AND 2 (AND 3 1)) (OR (AND 3 1) (AND 2 3))) - 7
4: (OR (AND 2 (AND 3 1)) (OR (AND 3 1) (AND 2 3))) - 7
5: (OR (OR 1 (OR 3 1)) 1) - 6
6: (OR (AND (AND (AND 3 1) 2) 3) (OR (OR 3 1) (AND 2 3))) - 6

Crossover & Mutation:





Generation 10:
1: (OR (AND (OR (OR 3 1) 3) 2) (AND (AND (OR 2 3) 1) (AND 2 (OR 1 1))))
2: (OR (AND 2 (AND 3 2)) (OR (AND (AND 2 3) 1) (AND 2 3)))
3: (OR (AND (OR (AND 3 1) (AND 2 3)) 2) (AND (AND (AND 2 3) (OR 1 (OR 3 1))) (AND 2 (OR 1 1))))
4: (OR 1 1)
5: (OR (AND 2 (AND 3 1)) (OR (AND 3 1) 2))
6: (OR (AND (AND (AND 3 1) (AND 2 3)) 3) (OR (OR 3 1) (AND 2 3)))

2. What issues did you encounter with this exercise? You may need to consult the genetic programming literature. I am expecting an answer worthy of a graduate student, and not just a sentence or two.

The main issue that we encounter while solving a problem using Genetic Programming is Code Bloat. Code Bloat means in every generation the fattest trees survive and go to next generation. This results in increasing depth of the trees as we progress through the generations.

Another issue that we encounter is we are never changing the root node of the tree. So, we will never get alternative solutions where the root node is different. In this way we are losing search in some part of the search space.

3. What could be done to improve the performance of this simulation? I am expecting an answer worthy of a graduate student, not just a sentence or two or something copied from the web. Think about your answer to 2., and what you could do to address the issues you encountered.

One thing we can do to solve root never changing issue is we can start running count from the root node so that when running count is 1 for crossover or mutation we will change the value of the root node thereby increasing our search space to find more optimal solution.

To solve Code Bloat issue we can do either or both of the following possible solutions:

a) Taking a maximum depth allowed and discard children that gets depth greater than the maximum depth that we set. This will eliminate code bloat but we might lose some search space since we can't guarantee that we will find the optimal solution within that maximum depth.
b) Adding a penalty to children whose depth exceeds a certain limit. This is better when compared to above approach because we are not losing search space in this method. We just add penalty to trees. We need to figure out that penalty value and that penalty should be proportional to the depth of the tree, not a constant value for all depths.