Output Format

```
3 - 4bit sboxes pbox size 12

SBOX : [ 8, 15, 2, 10, 12, 11, 6, 13, 14, 5, 4, 0, 1, 3, 9, 7 ]
PBOX : [ 9, 8, 2, 6, 1, 7, 11, 3, 10, 4, 0, 5 ]


0000|0000|1101 <-- input bits to first round
---- ---- ---- <-- sbox
0000|0000|0010 <-- output bits
               <-- permutation box
1000|0000|0000 <-- input to next round
---- ---- ----
1000|0000|0000

0000|0000|0100
---- ---- ----
0000|0000|0001

0000|0100|0000
---- ---- ----
0000|0001|0000

0001|0000|0000
---- ---- ----
0001|0000|0000

0000|0010|0000
---- ---- ----
0000|1001|0000

0101|0000|0000
---- ---- ----
1111|0000|0000

0010|0010|1100
total bias: 3/512 #note that only absolute value of bias is calculated
return value = ([13, 2048, 4, 64, 256, 32, 1280, 556], [2, 2048, 1, 16, 256, 144, 3840])
where
[13, 2048, 4, 64, 256, 32, 1280, 556] are all the input masks
[2, 2048, 1, 16, 256, 144, 3840] are all the output masks
```

Claim 2.1

The claim is definitely false, the greedy staratergy being, having the optimal linear relationship for n round SPN, we can take the maximum combinations for n+1th round from the n-round relationship. It is highly possible (and is easily observed if we increase the number of rounds) that forcing the given combination for n round, we might end up (owing to the permutation box) with input bit combinations which lead to really bad bias no matter what outputs we take and the optimal path could be some sub-optimal path for n-1th round.

This can be also seen in the example above the biases for input_mask,output_mask = (13,2) is 4/16, but the max bias observed is 6/16 (so the optimal path might not take the given output of maximal bias as it may lead to lower overall bias)

Algorithm

The algorithm involves a couple of things
z3 SMT solver is used involving theory of bitvectors and a MAX-SAT solver - `bias` is calculated as original absolute bias times `2**sbox_size` - `sboxf` uninterprated function is created on `BitVector(n) BitVector(n)` where `n` is sbox size in bits to represent the constraints for sbox bias for a corresponding input/output mask - `permutation` function creates constraints between the pbox relationship of input of round n+1 to output of round n - `inps` and `oups` are 2d arrays (num sboxes in a round times number of rounds) of bitvectors representing the input and output masks to consider in each round - `objectives` is the objective we want the SMT solver to optimize i.e. the final bias of the network - Additional constraints are added namely - If input masks of a corresponding sbox is zero, output mask should be zero - If input masks are non-zero, output mask should be non-zero - If bias corresponding to given input/output is zero, ignore that input output - Atleast one input mask to first layer should be non-zero - `non_zero` is the array of sbox indices which we need the final round output masks to end up to (`non_zero = []` means we can take any optimal path we like) (used in question3 to reduce the number of key bits to bruteforce)

The theoretical time complexity involved is guessing `2**pbox` sized input and output masks in each round which result in n-round linear approximation for the spn network. We figure out all input output masks for the first round, rest all input rounds are permutation of output mask of last round. So we search over 1 input mask round, and n output mask rounds. So the complexity should be $O((2^p)^{n+1})$ where p is the size of pbox and n is the number of rounds.

But since a SMT solver is used, predicting the actual complexity is not feasible since it depends on the constraints employed which themselves depend on the sbox and pbox hugely (observed times are quite faster and robust than a mere BFS).