The horse numbers H(n) as introduced to me by Bill Gasarch, are simply the number of ways some group of n arbitrarily assigned integers may be ordered, if the orderings possible are less than, greater than, and equal. The name horse numbers comes from the fact that in a horse race result, there may not only be horse x beat horse y, but horses may also tie. Gasrach has provided the following problem: given n horses, how many ways may they be ordered including ties, under the constraint that horse A must always beat horse B, or in the language of integers A < B. Calculating the possible orderings under this constraint has been given the name Bill numbers B(n). Bill has provided a way to calculate these numbers; the form is a recursion on H(n) and B(n).

The following simpler expression of B(n) is given. B(n) = (H(n) - H(n - 1))/2. Two arguments of symmetry are needed to justify this formula. First H(n) - H(n - 1) is the number of ways horses may finish a race if two horses A,B are not allowed to tie ie A ≠ B. Then the division by 2 is due to the symmetry that A < B -> B < A always produces a valid counted horse race configuration for A,B arbitrary. Therefore to count only cases A < B config, we know for each there exists a config where A > B, just divide by 2 to remove the unwanted counting.

This new formulation for B(n) appears to be more quickly computable on the surface, this may just be an illusion of recursing on two named functions rather than one.

I am curious what imposing further restrictions will look like, for any arbitrary set of compatible constrains of form (A < B)... do the symmetries exploited continue to work, allowing the number of configurations to be simply expressed as a linear combination of H(n - k) for k in a finite set independent of n? Put simply, are the modified horse numbers computable with some finite collection of nearby horse numbers for all n?

The following python3 program was written to generate numbers according to each formulation of B(n) for comparison sake:

```
from math import comb
def H(n):
    if n == 0:
        return 1
    if n == 1:
        return 1
    if n == 2:
        return 3
    if n == 3:
        return 13
    sum = 0
    for i in range(1,n + 1):
        sum += H(n - i) * comb(n,i)
    return sum
```

```
def B(n):
    sum = 0
    for i in range(0,n-1):
        sum +=comb(n-2,i) * H(n-i-1)
    for i in range(1,n-1):
        sum+=comb(n-2,i) * B(n - i)
    return sum
print(H(4))

for i in range(4,100):
    bill = (B(i))

    ben = ((H(i) - H(i - 1))//2)

    if (ben != bill):
        print("B(",i,")",bill,ben,"oops.")
    else:
        print("B(",i,")",bill,ben,)


"""
B( 4 ) 31 31
B( 5 ) 233 233
B( 6 ) 2071 2071
B( 7 ) 21305 21305
B( 8 ) 249271 249271
B( 9 ) 3270713 3270713
B( 10 ) 47580151 47580151
B( 11 ) 760192505 760192505
B( 12 ) 13234467511 13234467511
B( 13 ) 249383390393 249383390393
B( 14 ) 5057242311031 5057242311031
B( 15 ) 109820924003705 109820924003705
B( 16 ) 2542685745501751 2542685745501751
B( 17 ) 62527556173577273 62527556173577273
B( 18 ) 1627581948113854711 1627581948113854711
B( 19 ) 44708026328035782905 44708026328035782905
B( 20 ) 1292443104462527895991 1292443104462527895991
B( 21 ) 39223568601129844839353 39223568601129844839353
```

```
B( 22 ) 124685979740234315533119l 124685979740234315533119l
B( 23 ) 41431803458471259455018105 41431803458471259455018105
"""
```