Hamming Distance

Assignment 6
Group 2G

Explanation of Problem:

There is a party (given as Alice) transmitting bits- this is stored as BitStream. Of this P% of them are obtained by the receiving party (Bob) with Q% error. However, there is a time difference involved and hence an offset between the arrays. The receiving party reveals some of the bits' locations and values to the transmitter, who must compare with the original stream to determine the offset.

This is where Hamming Distance comes in- it is the number of correctly matching elements between two arrays. When two arrays perfectly match- Hamming distance is zero. Here we compare the received bits, BitStore-which is C in the problem statement- (which was obtained from an OffsetBitStream, given as B in the problem statement) with the BitStream. When the Hamming distance is minimum, we have reached the correct offset.

Method:

There are two main parts for this code: Creating BitStore and finding the Hamming distances.

To randomly select elements from BitStream, we have used used Fisher-Yates algorithm. Shuffle the array by interchanging the last considered element with a randomly chosen previous element, and then directly choose how many ever elements needed. The problem here is that the locations are lost- therefore we have maintained a record of swaps (SwapMap) to retrieve the locations as well.

Finding Hamming distance is easy, but for larger values of N, efficiency is lost. Therefore, we have used bitwise operators wherever possible instead of multiple ifs.

Outcomes:

Randomness Test

The <dieharder> randomness test is included here for BitStream:

```
#=============================================================================#
#            dieharder version 3.31.1 Copyright 2003 Robert G. Brown          #
#=============================================================================#
   rng_name    |           filename             |rands/second|
        mt19937|             RandomCheck.dat|  1.53e+08  |
#=============================================================================#
        test_name   |ntup| tsamples |psamples|  p-value |Assessment
#=============================================================================#
diehard_birthdays|   0|       100|     100|0.57740265|  PASSED
diehard_operm5|   0|   1000000|     100|0.98961978|  PASSED
diehard_rank_32x32|   0|     40000|     100|0.53538533|  PASSED
```

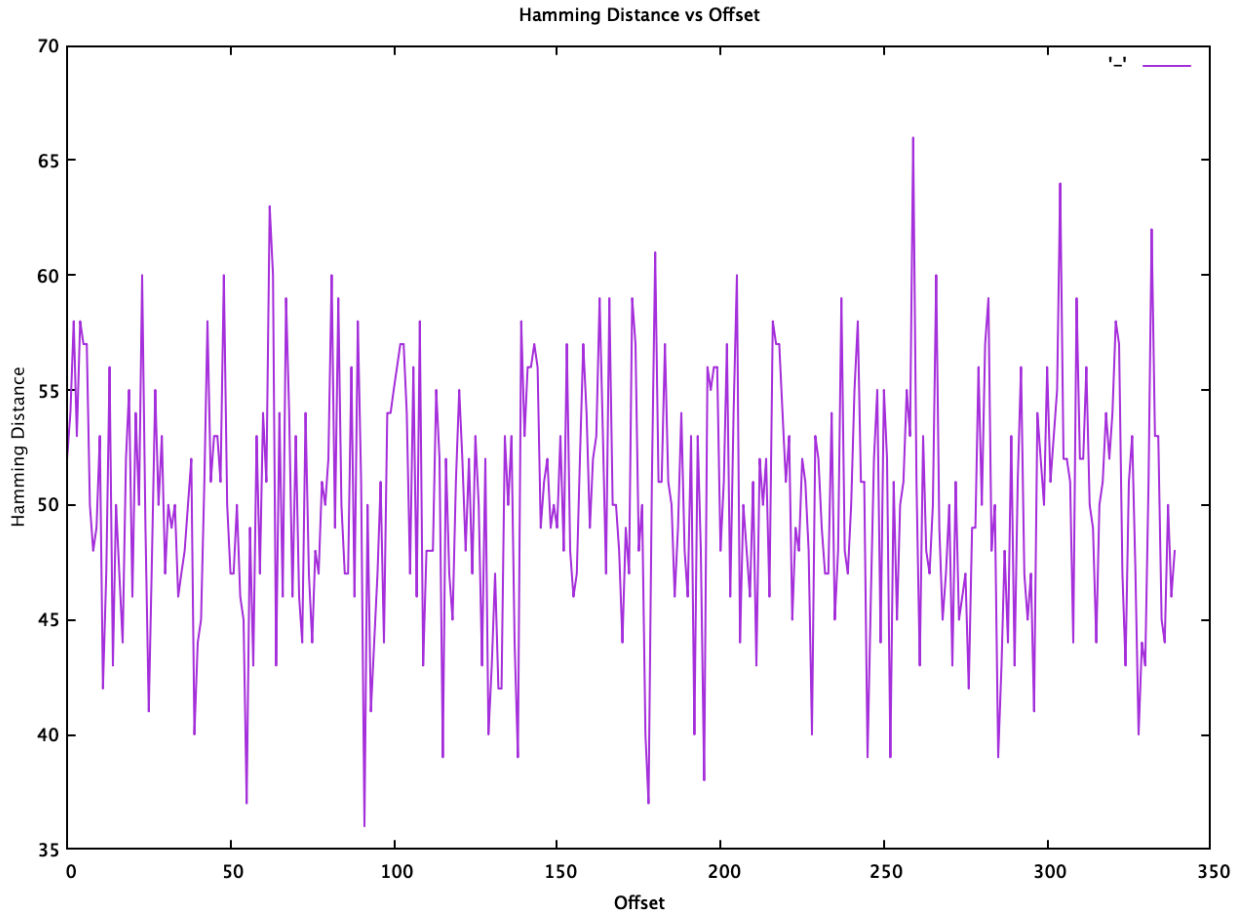
```
diehard_rank_6x8|   0|    100000|     100|0.60418166|  PASSED
diehard_bitstream|   0|   2097152|     100|0.90745670|  PASSED
diehard_opso|   0|   2097152|     100|0.17781820|  PASSED
diehard_oqso|   0|   2097152|     100|0.76910274|  PASSED
diehard_dna|   0|   2097152|     100|0.29672829|  PASSED
diehard_count_1s_str|   0|    256000|     100|0.18584227|  PASSED
diehard_count_1s_byt|   0|    256000|     100|0.74399271|  PASSED
diehard_parking_lot|   0|     12000|     100|0.35161543|  PASSED
diehard_2dsphere|   2|      8000|     100|0.70989128|  PASSED
diehard_3dsphere|   3|      4000|     100|0.27722189|  PASSED
diehard_squeeze|   0|    100000|     100|0.11826415|  PASSED
diehard_sums|   0|       100|     100|0.60215204|  PASSED
diehard_runs|   0|    100000|     100|0.73452719|  PASSED
diehard_runs|   0|    100000|     100|0.45985166|  PASSED
diehard_craps|   0|    200000|     100|0.37676750|  PASSED
diehard_craps|   0|    200000|     100|0.48355435|  PASSED
 marsaglia_tsang_gcd|   0|  10000000|     100|0.89735259|  PASSED
 marsaglia_tsang_gcd|   0|  10000000|     100|0.42334863|  PASSED
        sts_monobit|   1|    100000|     100|0.45334112|  PASSED
          sts_runs|   2|    100000|     100|0.30710368|  PASSED
         sts_serial|   1|    100000|     100|0.91675153|  PASSED
         sts_serial|   2|    100000|     100|0.52552909|  PASSED
         sts_serial|   3|    100000|     100|0.15870568|  PASSED
         sts_serial|   3|    100000|     100|0.48208932|  PASSED
         sts_serial|   4|    100000|     100|0.07077622|  PASSED
         sts_serial|   4|    100000|     100|0.03585343|  PASSED
         sts_serial|   5|    100000|     100|0.95946868|  PASSED
         sts_serial|   5|    100000|     100|0.61143112|  PASSED
         sts_serial|   6|    100000|     100|0.68094404|  PASSED
         sts_serial|   6|    100000|     100|0.79185681|  PASSED
         sts_serial|   7|    100000|     100|0.66504431|  PASSED
         sts_serial|   7|    100000|     100|0.76714822|  PASSED
         sts_serial|   8|    100000|     100|0.57680111|  PASSED
         sts_serial|   8|    100000|     100|0.39063780|  PASSED
         sts_serial|   9|    100000|     100|0.94894091|  PASSED
         sts_serial|   9|    100000|     100|0.76606542|  PASSED
         sts_serial|  10|    100000|     100|0.71933857|  PASSED
         sts_serial|  10|    100000|     100|0.31302418|  PASSED
         sts_serial|  11|    100000|     100|0.86108881|  PASSED
         sts_serial|  11|    100000|     100|0.67562150|  PASSED
         sts_serial|  12|    100000|     100|0.67205669|  PASSED
         sts_serial|  12|    100000|     100|0.96897855|  PASSED
         sts_serial|  13|    100000|     100|0.92146528|  PASSED
         sts_serial|  13|    100000|     100|0.99854626|   WEAK
         sts_serial|  14|    100000|     100|0.95625462|  PASSED
         sts_serial|  14|    100000|     100|0.48534070|  PASSED
         sts_serial|  15|    100000|     100|0.91100642|  PASSED
         sts_serial|  15|    100000|     100|0.78826856|  PASSED
         sts_serial|  16|    100000|     100|0.14981135|  PASSED
         sts_serial|  16|    100000|     100|0.42832633|  PASSED
        rgb_bitdist|   1|    100000|     100|0.94783579|  PASSED
        rgb_bitdist|   2|    100000|     100|0.94332197|  PASSED
        rgb_bitdist|   3|    100000|     100|0.67478905|  PASSED
        rgb_bitdist|   4|    100000|     100|0.70999932|  PASSED
        rgb_bitdist|   5|    100000|     100|0.77047751|  PASSED
        rgb_bitdist|   6|    100000|     100|0.46081104|  PASSED
        rgb_bitdist|   7|    100000|     100|0.98544189|  PASSED
        rgb_bitdist|   8|    100000|     100|0.20526603|  PASSED
        rgb_bitdist|   9|    100000|     100|0.20432461|  PASSED
        rgb_bitdist|  10|    100000|     100|0.93590252|  PASSED
        rgb_bitdist|  11|    100000|     100|0.78331992|  PASSED
        rgb_bitdist|  12|    100000|     100|0.34089202|  PASSED
rgb_minimum_distance|   2|     10000|    1000|0.58505604|  PASSED
rgb_minimum_distance|   3|     10000|    1000|0.87065412|  PASSED
```

```
rgb_minimum_distance|   4|     10000|    1000|0.29126590|  PASSED
rgb_minimum_distance|   5|     10000|    1000|0.45513700|  PASSED
   rgb_permutations|   2|    100000|     100|0.71102830|  PASSED
   rgb_permutations|   3|    100000|     100|0.94263073|  PASSED
   rgb_permutations|   4|    100000|     100|0.92006333|  PASSED
   rgb_permutations|   5|    100000|     100|0.43943158|  PASSED
     rgb_lagged_sum|   0|   1000000|     100|0.76292664|  PASSED
     rgb_lagged_sum|   1|   1000000|     100|0.55217161|  PASSED
     rgb_lagged_sum|   2|   1000000|     100|0.54887568|  PASSED
     rgb_lagged_sum|   3|   1000000|     100|0.94495244|  PASSED
     rgb_lagged_sum|   4|   1000000|     100|0.04883133|  PASSED
     rgb_lagged_sum|   5|   1000000|     100|0.62984368|  PASSED
     rgb_lagged_sum|   6|   1000000|     100|0.86537248|  PASSED
     rgb_lagged_sum|   7|   1000000|     100|0.35020959|  PASSED
     rgb_lagged_sum|   8|   1000000|     100|0.27551529|  PASSED
     rgb_lagged_sum|   9|   1000000|     100|0.69205041|  PASSED
     rgb_lagged_sum|  10|   1000000|     100|0.99978040|   WEAK
     rgb_lagged_sum|  11|   1000000|     100|0.33087514|  PASSED
     rgb_lagged_sum|  12|   1000000|     100|0.01994448|  PASSED
     rgb_lagged_sum|  13|   1000000|     100|0.15930777|  PASSED
     rgb_lagged_sum|  14|   1000000|     100|0.80015736|  PASSED
     rgb_lagged_sum|  15|   1000000|     100|0.92900946|  PASSED
     rgb_lagged_sum|  16|   1000000|     100|0.82809487|  PASSED
     rgb_lagged_sum|  17|   1000000|     100|0.38865841|  PASSED
     rgb_lagged_sum|  18|   1000000|     100|0.91421611|  PASSED
     rgb_lagged_sum|  19|   1000000|     100|0.96098326|  PASSED
     rgb_lagged_sum|  20|   1000000|     100|0.54824982|  PASSED
     rgb_lagged_sum|  21|   1000000|     100|0.05348352|  PASSED
     rgb_lagged_sum|  22|   1000000|     100|0.78106985|  PASSED
     rgb_lagged_sum|  23|   1000000|     100|0.04427423|  PASSED
     rgb_lagged_sum|  24|   1000000|     100|0.83883864|  PASSED
     rgb_lagged_sum|  25|   1000000|     100|0.88614916|  PASSED
     rgb_lagged_sum|  26|   1000000|     100|0.17381860|  PASSED
     rgb_lagged_sum|  27|   1000000|     100|0.84421700|  PASSED
     rgb_lagged_sum|  28|   1000000|     100|0.15896963|  PASSED
     rgb_lagged_sum|  29|   1000000|     100|0.48686919|  PASSED
     rgb_lagged_sum|  30|   1000000|     100|0.10292736|  PASSED
     rgb_lagged_sum|  31|   1000000|     100|0.56990044|  PASSED
     rgb_lagged_sum|  32|   1000000|     100|0.85486606|  PASSED
    rgb_kstest_test|   0|     10000|    1000|0.41739099|  PASSED
    dab_bytedistrib|   0|  51200000|       1|0.98736407|  PASSED
           dab_dct| 256|     50000|       1|0.77589160|  PASSED
Preparing to run test 207.  ntuple = 0
       dab_filltree|  32|  15000000|       1|0.20850121|  PASSED
       dab_filltree|  32|  15000000|       1|0.20229280|  PASSED
Preparing to run test 208.  ntuple = 0
      dab_filltree2|   0|   5000000|       1|0.23146833|  PASSED
      dab_filltree2|   1|   5000000|       1|0.93880177|  PASSED
Preparing to run test 209.  ntuple = 0
       dab_monobit2|  12|  65000000|       1|0.92363509|  PASSED
```

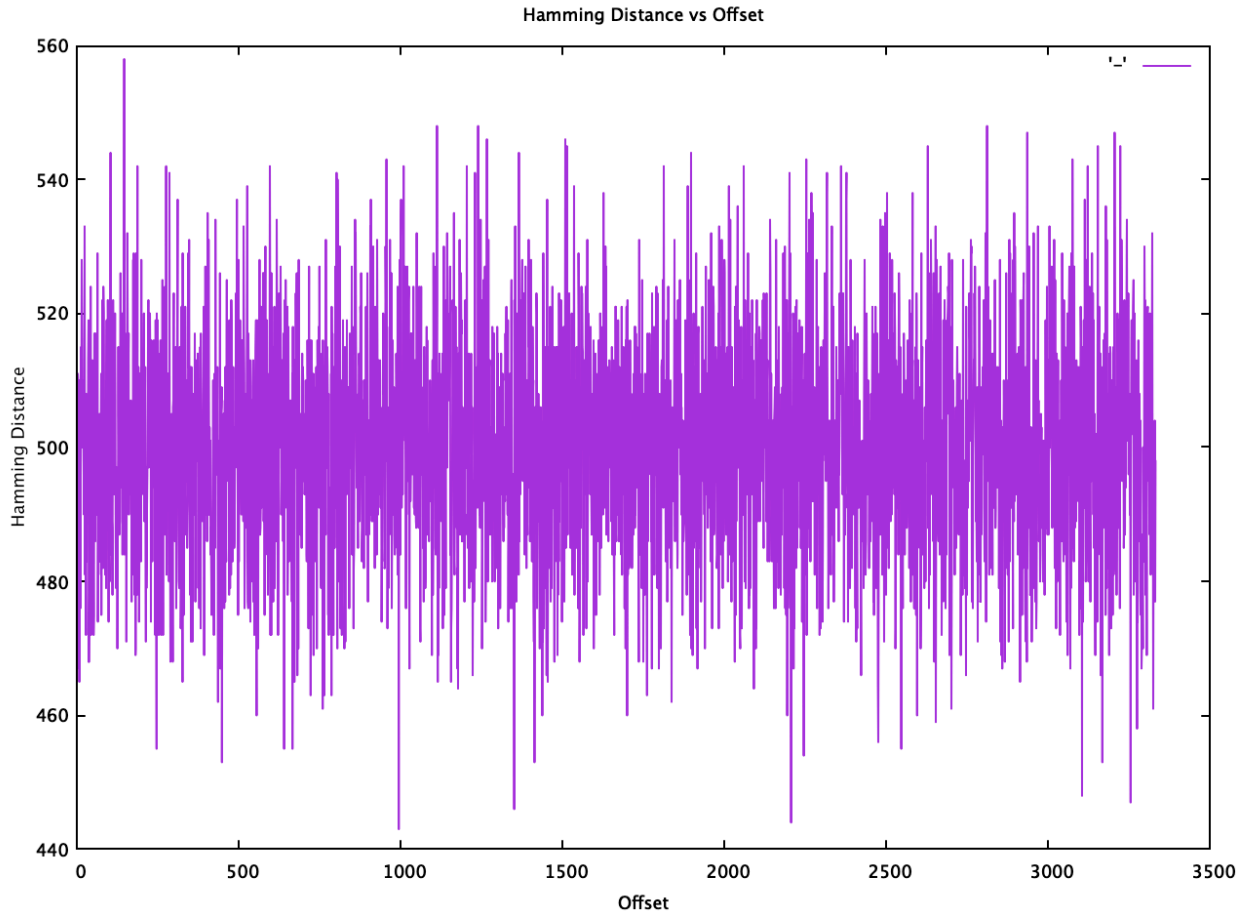
Graphs for different N

We note the problem statement doesn't explicitly require you to generate offset between N/10 and N, so we have generated it between 0 and N/100 to make a nicer graph that isn't overcrowded in the middle.

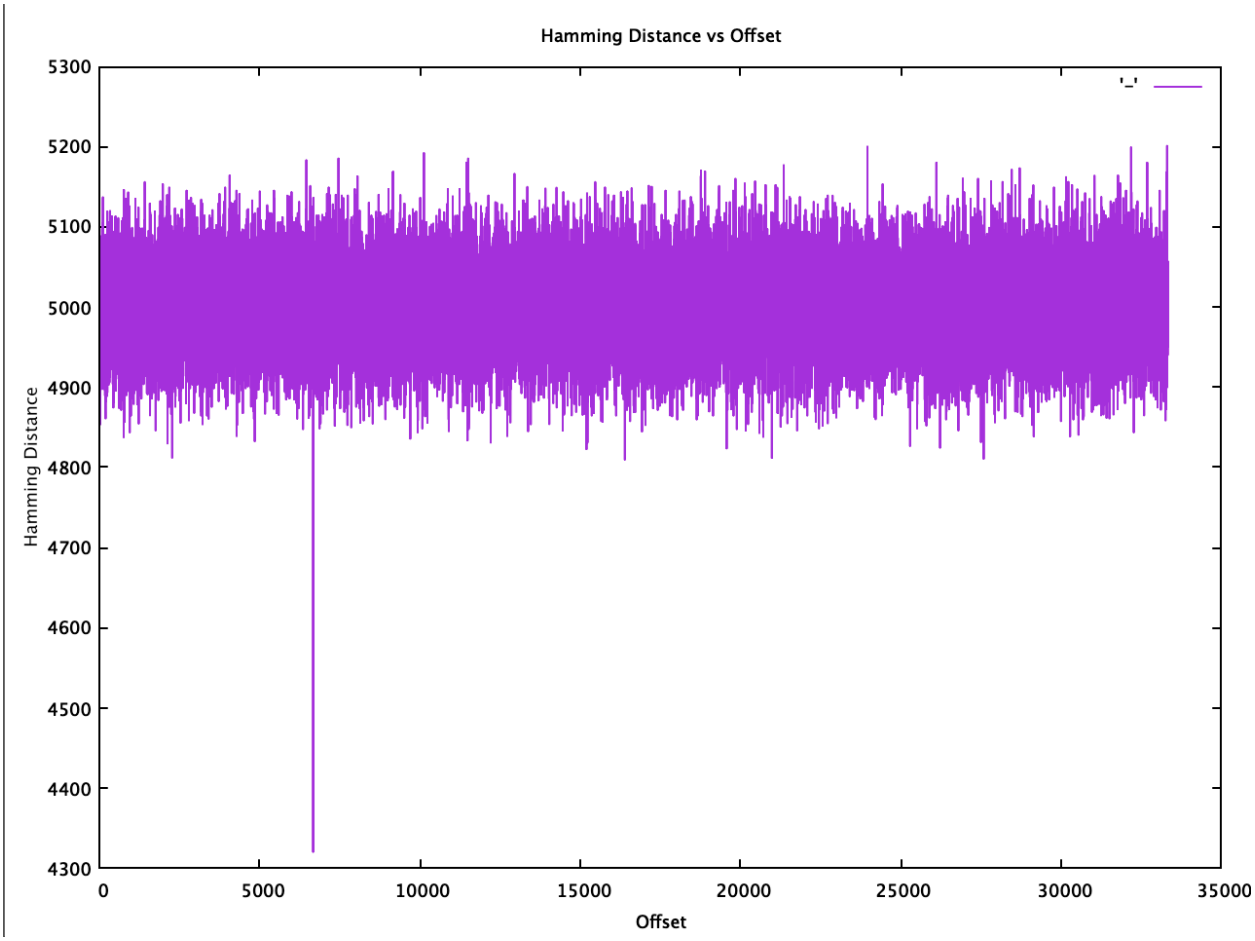
1. For N=1000, P=0.1, Q=0.43
   Bit Error Rate=46.000000
   MaxLocation=659
   MaxOffset=340
   Generated Offset=66
   Calculated Offset=91



2. For N=10000, P=0.1, Q=0.43
   Bit Error Rate=43.000000
   MaxLocation=659
   MaxOffset=340
   Generated Offset=66
   Calculated Offset=91



3. For N=100000, P=0.1, Q=0.43
   Bit Error Rate=43.210000
   MaxLocation=66661
   MaxOffset=33338
   Generated Offset=6666
   Calculated Offset=6666



We see that the larger the value of N, the greater the our ability to calculate the correct offset.

Thanks!