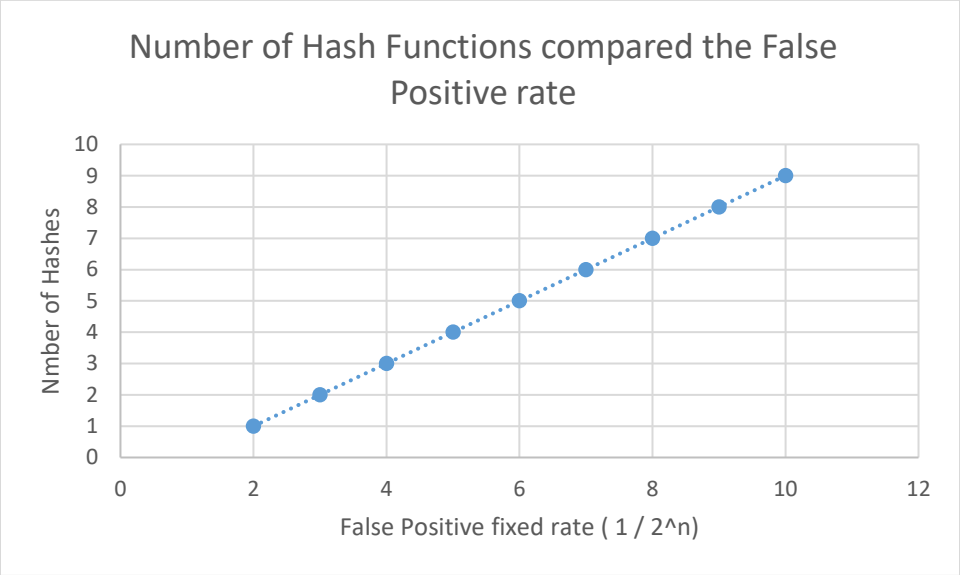
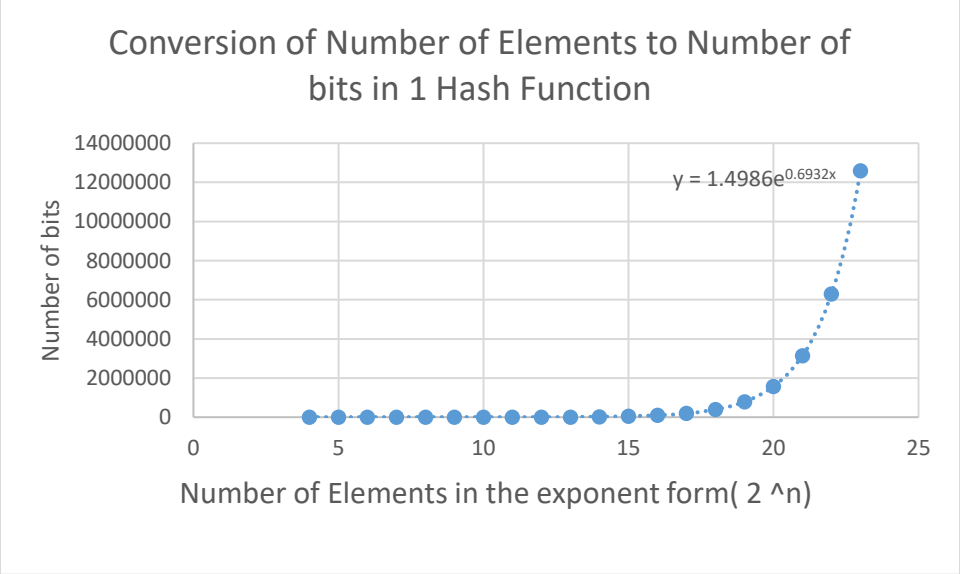
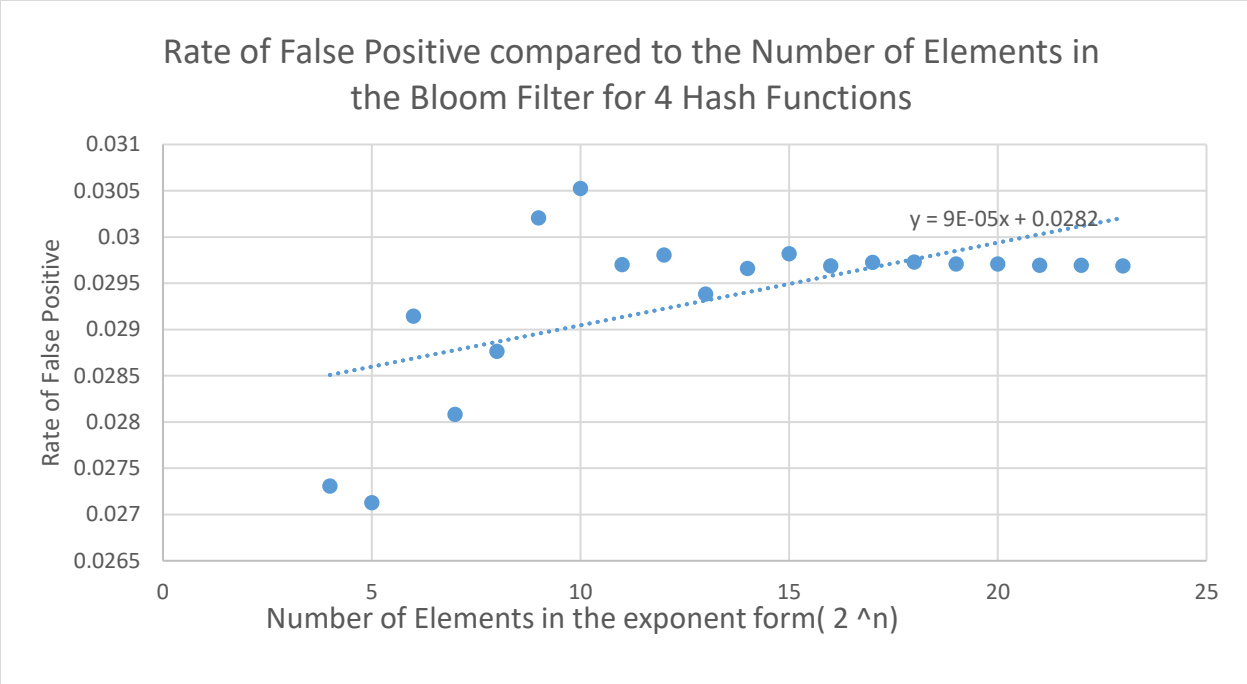


This graph shows the number of Hash Functions compared to the number of elements and is similar to all the fixed rate expressions.



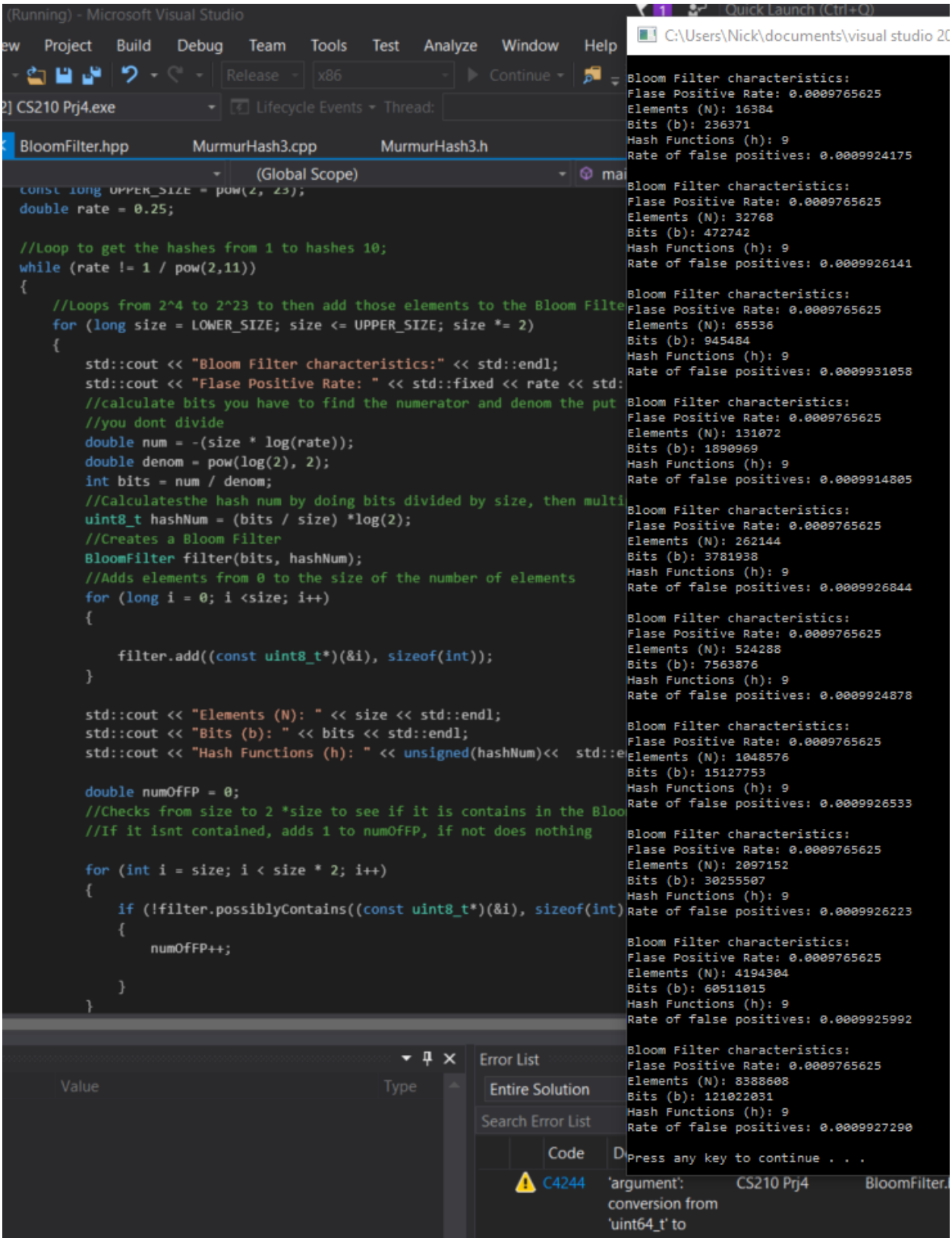
The conversion of number of elements to the number of bits. This is will be similar to the all the other number of Hash Functions. As you can see from the graph, the more number of elements that are in the Bloom Filter, you will need to acquire more space to hold all of these values



Comparing Rate of False Positive to the number of elements in the Bloom Filter. All the other number of Hash Functions will be similar to this scatter plot. As you can see from the graph, with the lower number of elements in the Bloom Filter, there isn't a consist rate of false positives. But as the number of elements increases, the Rate of false positives evens out and becomes more accurate.

Overall Analysis

My analysis of this data structure – Bloom Filter– seems that it is a somewhat nice data structure to use when you need access to data quickly, but not too specific of detail. Pretty much saying that it checks to say, "Hey I looked at something like that before and I believe it is here". Even though this could be nice for some information of data. You can't remove items from a Bloom Filter and you can't actually say look at what say it attached that information you are checking for. Like with a node you can have information attached the node that you can access when you find the node, with a Bloom Filter you cannot do this sort of thing. This is why a Bloom Filter is a probabilistic data structure and can only be used in certain applications.


(Running) - Microsoft Visual Studio
Project Build Debug Team Tools Test Analyze Window Help
Release x86 Continue
CS210 Prj4.exe
Lifecycle Events Thread:
BloomFilter.hpp MurmurHash3.cpp MurmurHash3.h
(Global Scope)
double rate = 0.25;
//Loop to get the hashes from 1 to hashes 10;
while (rate != 1 / pow(2,11))
{
//Loops from 2^4 to 2^23 to then add those elements to the Bloom Filte
for (long size = LOWER_SIZE; size <= UPPER_SIZE; size *= 2)
{
std::cout << "Bloom Filter characteristics:" << std::endl;
std::cout << "Flase Positive Rate: " << std::fixed << rate << std:
//calculate bits you have to find the numerator and denom the put
//you dont divide
double num = -(size * log(rate));
double denom = pow(log(2), 2);
int bits = num / denom;
//Calculatesthe hash num by doing bits divided by size, then multi
uint8_t hashNum = (bits / size) *log(2);
//Creates a Bloom Filter
BloomFilter filter(bits, hashNum);
//Adds elements from 0 to the size of the number of elements
for (long i = 0; i <size; i++)
{
filter.add((const uint8_t*)(&i), sizeof(int));
}
std::cout << "Elements (N): " << size << std::endl;
std::cout << "Bits (b): " << bits << std::endl;
std::cout << "Hash Functions (h): " << unsigned(hashNum)<< std::e
double numOfFP = 0;
//Checks from size to 2 *size to see if it is contains in the Bloo
//If it isnt contained, adds 1 to numOfFP, if not does nothing
for (int i = size; i < size * 2; i++)
{
if (!filter.possiblyContains((const uint8_t*)(&i), sizeof(int)
{
numOfFP++;
}
}
Value Type
Error List
Entire Solution
Search Error List
Code
C4244 'argument': conversion from 'uint64_t' to
CS210 Prj4
BloomFilter.
Quick Launch (Ctrl+Q)
C:\Users\Nick\documents\visual studio 20
Bloom Filter characteristics:
Flase Positive Rate: 0.0009765625
Elements (N): 16384
Bits (b): 236371
Hash Functions (h): 9
Rate of false positives: 0.0009924175
Bloom Filter characteristics:
Flase Positive Rate: 0.0009765625
Elements (N): 32768
Bits (b): 472742
Hash Functions (h): 9
Rate of false positives: 0.0009926141
Bloom Filter characteristics:
Flase Positive Rate: 0.0009765625
Elements (N): 65536
Bits (b): 945484
Hash Functions (h): 9
Rate of false positives: 0.0009931058
Bloom Filter characteristics:
Flase Positive Rate: 0.0009765625
Elements (N): 131072
Bits (b): 1890969
Hash Functions (h): 9
Rate of false positives: 0.0009914805
Bloom Filter characteristics:
Flase Positive Rate: 0.0009765625
Elements (N): 262144
Bits (b): 3781938
Hash Functions (h): 9
Rate of false positives: 0.0009926844
Bloom Filter characteristics:
Flase Positive Rate: 0.0009765625
Elements (N): 524288
Bits (b): 7563876
Hash Functions (h): 9
Rate of false positives: 0.0009924878
Bloom Filter characteristics:
Flase Positive Rate: 0.0009765625
Elements (N): 1048576
Bits (b): 15127753
Hash Functions (h): 9
Rate of false positives: 0.0009926533
Bloom Filter characteristics:
Flase Positive Rate: 0.0009765625
Elements (N): 2097152
Bits (b): 30255507
Hash Functions (h): 9
Rate of false positives: 0.0009926223
Bloom Filter characteristics:
Flase Positive Rate: 0.0009765625
Elements (N): 4194304
Bits (b): 60511015
Hash Functions (h): 9
Rate of false positives: 0.0009925992
Bloom Filter characteristics:
Flase Positive Rate: 0.0009765625
Elements (N): 8388608
Bits (b): 121022031
Hash Functions (h): 9
Rate of false positives: 0.0009927290
Press any key to continue . . .