CSCI E-124-Spring, 2004 Homework 4

Russell Lowke, April 24th 2004

1. If the social security number consisted of 9 randomly generated digits...

a) How many people would you need to have in a room before it was more likely than not that two had the same last four digits?

Probability that there will NOT be a match is in a room with n people is

$$\prod_{i=2}^{n} (1 - \frac{i-1}{10^4})$$

The moment that this passes below 50 it is more than probable that there will be a match, so:

$\prod_{i=2}^{n} (1 - \frac{i-1}{10^4}) < 50$, which occurs at n = 119, which has a probability of .494163006162 that there will not be two people in the room with the same last four digits of their social security.

b) How many numbers could be issued before it would be more likely than not that there is a duplicate number?

This will happen when $\prod_{i=2}^{n} (1 - \frac{i-1}{10^9}) < 50$, which occurs at n = 37 234,

which has a probability of 0.499986

c) What would the answers for the above questions be if there were 13 digit social security numbers?

The first question would remain the same, with n = 119, as the last four digits are still the last four digits. The second question would change to:

$$\prod_{i=2}^{n} (1 - \frac{i-1}{10^{13}}) < 50$$

I tried to compute this in Mathememática, which only would accept n up to 1,000,000 (for which the probability that the numbers are different was still 0.951229). At 1,100,000 it stopped and would not compute as n was so large.

2. Suppose each person gets a random hash value from the range [1...n] (for the case of birthdays, n would be 365)...

a) Show that for some constant c_1, when there are at least $c_1\sqrt{n}$ people in a room, the probability that no two have the same hash value is at most $\frac{1}{e}$

$$\prod_{i=2}^{c_1\sqrt{n}} \left(1 - \frac{i-1}{n}\right) \leq \frac{1}{e} \qquad e^{-x} \geq 1 - x \quad \text{and} \quad e^{-x-x^2} \leq 1 - x \text{ for } x \leq \frac{1}{2}$$

When n = 365 this is the birthday paradox. I calculated the number 28 to be the first one beyond 1/e (which is about 0.3679). So 28 = c1*19.1050 (which is the square root of 365). When dividing both sides by 19.1050 we get c_1 = 1.46558698325. This works for other values of n as well.

b) Show that for some constant c_2 (and sufficiently large n), when there are at most $c_2\sqrt{n}$ people in the room, the probability that no two have the same hash value is at least 1/2. Make the constants as close to optimal as possible.

$$\prod_{i=2}^{c_2\sqrt{n}} \left(1 - \frac{i-1}{n}\right) \geq \frac{1}{2} \qquad e^{-x} \geq 1 - x \quad \text{and} \quad e^{-x-x^2} \leq 1 - x \text{ for } x \leq \frac{1}{2}$$

When n = 365 we're at the birthday paradox yet again, and we can calculate that 22 is the last values for which the probability that two people don't share a birthday is more than 50%. So we have 22 = c2*19.1050 again, and when dividing we get a value c_2 = 1.1515326297. This also seems to consistently work for other larger values of n.

3. For the document similarity scheme... it would be better to store fewer bytes per document...

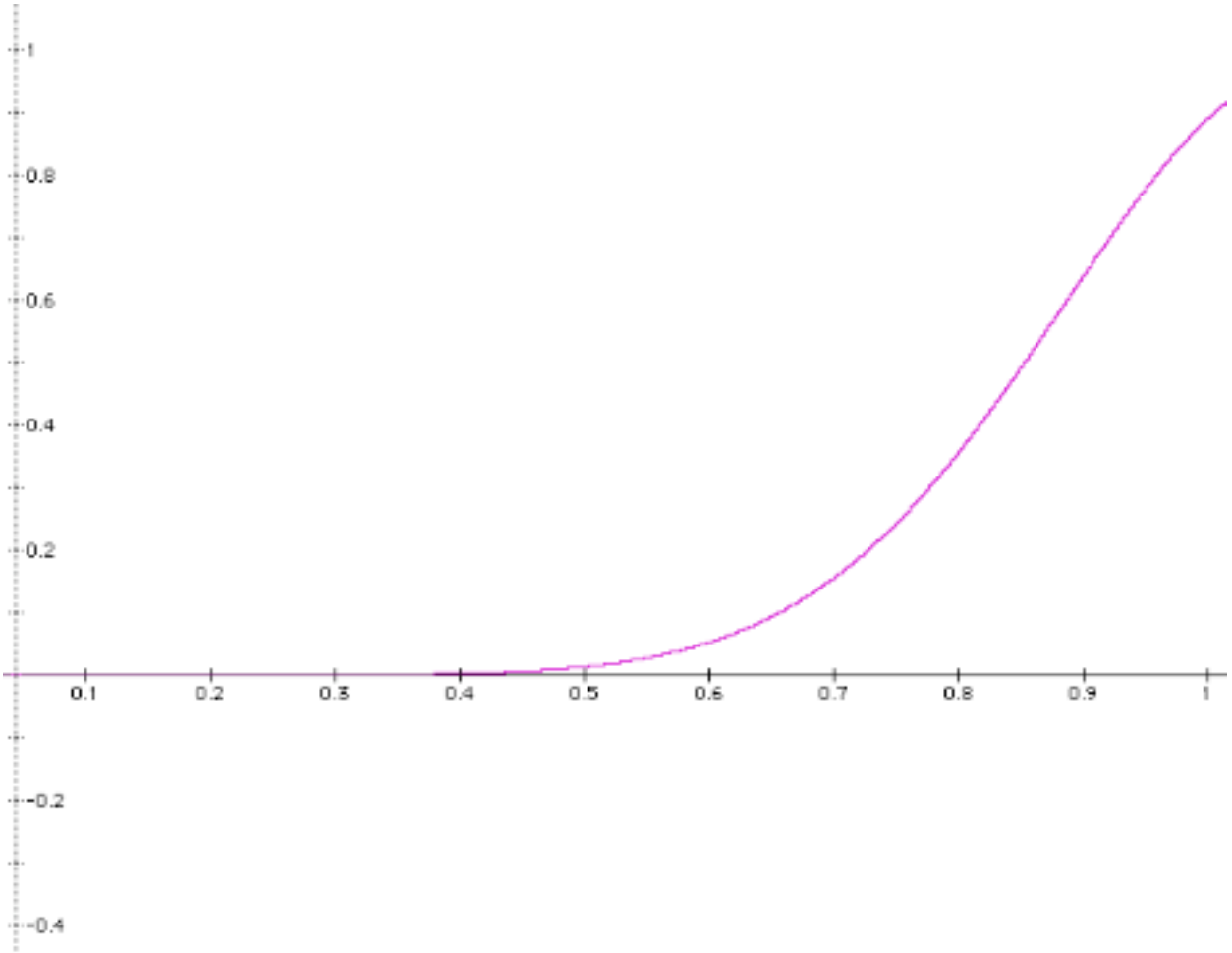
I had some ambiguity in deciphering the meaning of the problem. In particular, the "rehash each group of 15 values to get 6 new 64 bit values" part that was confusing. By definition, I thought that rehashing wasn't a procedure that got rid of values, just mapped them into some other, smaller set of numbers. So I figured that you meant to treat these new values like the original data, shingle them, then use the minimum values again at the values in the super-sketch.

In taking this approach, I found that we ended up with the same situation in which we were trying to predict the probability that a match will occur between two sets. The matches would only occur this time if (with a shingle size of 4) we get a sequence of 4 matches in a row. Since a single match occurs with probability r, the string of 4 matches will occur with probability r^4. So the chance that two shingles will match is r^4. When we hash our shingles down to values and take the minimum again, the chance that this minimum will match is equal again to the intersection of A with B divided by the union of A and B. The number of elements in the intersection should be r^4*|A| (or |B|, because they are both 11 in this case) because r^4 is the chance we'll get a match and thus r^4*|A| is the average number of elements that should match from the set. So we have r^4*(11)/(11 + 11) = (r^4) / 2.

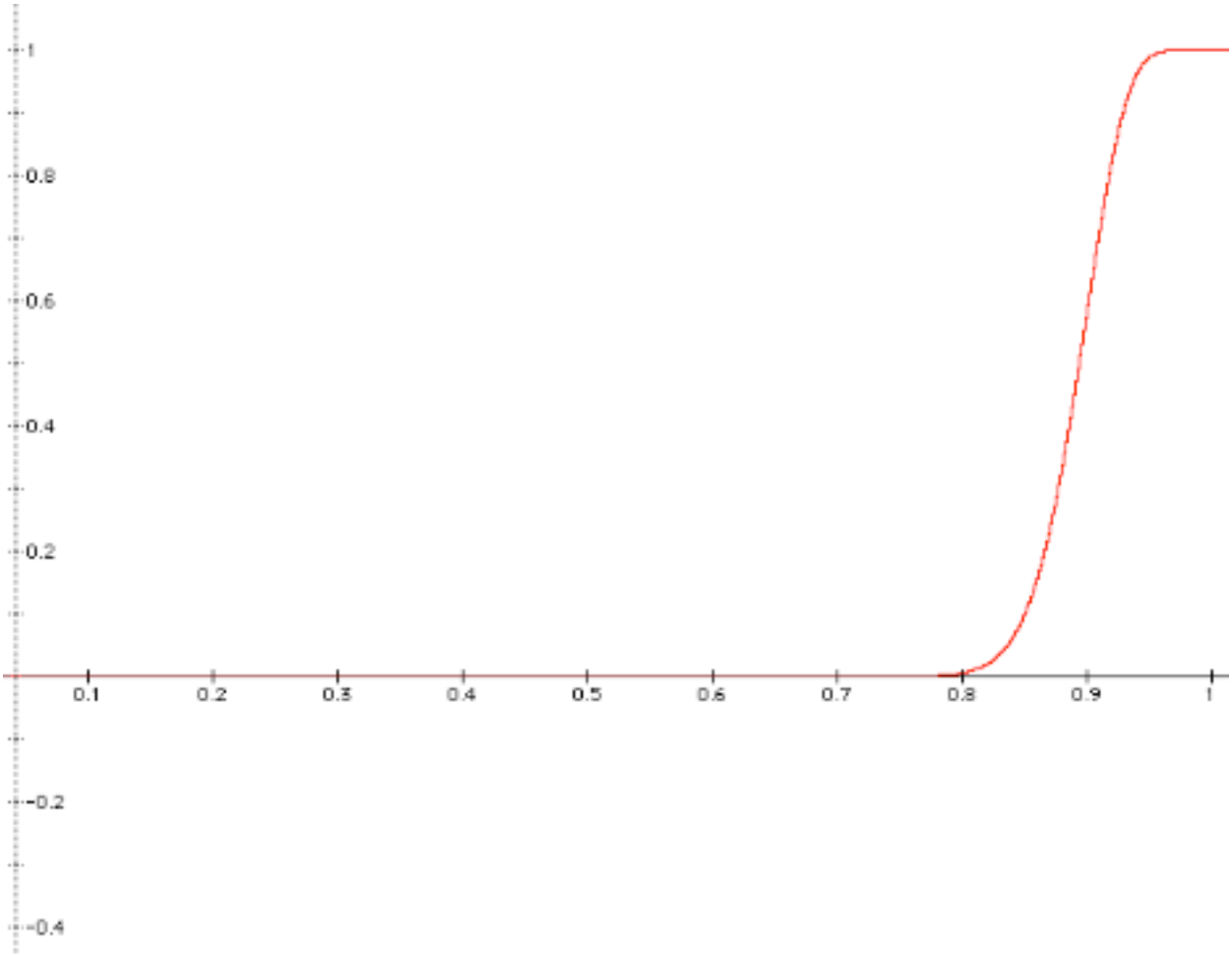
So it seems that, in separating this second hash from the first one, the term that came out as r now comes out as (r^4) / 2. Fitting this back into the equation given in class, we have the likelihood that 2 or more matches will occur out of 6 to be:

$$\sum_{k=2}^{6} \binom{6}{k} \left(\frac{r^r}{2}\right)^k \left(1 - \frac{r^r}{2}\right)^{6-k}$$

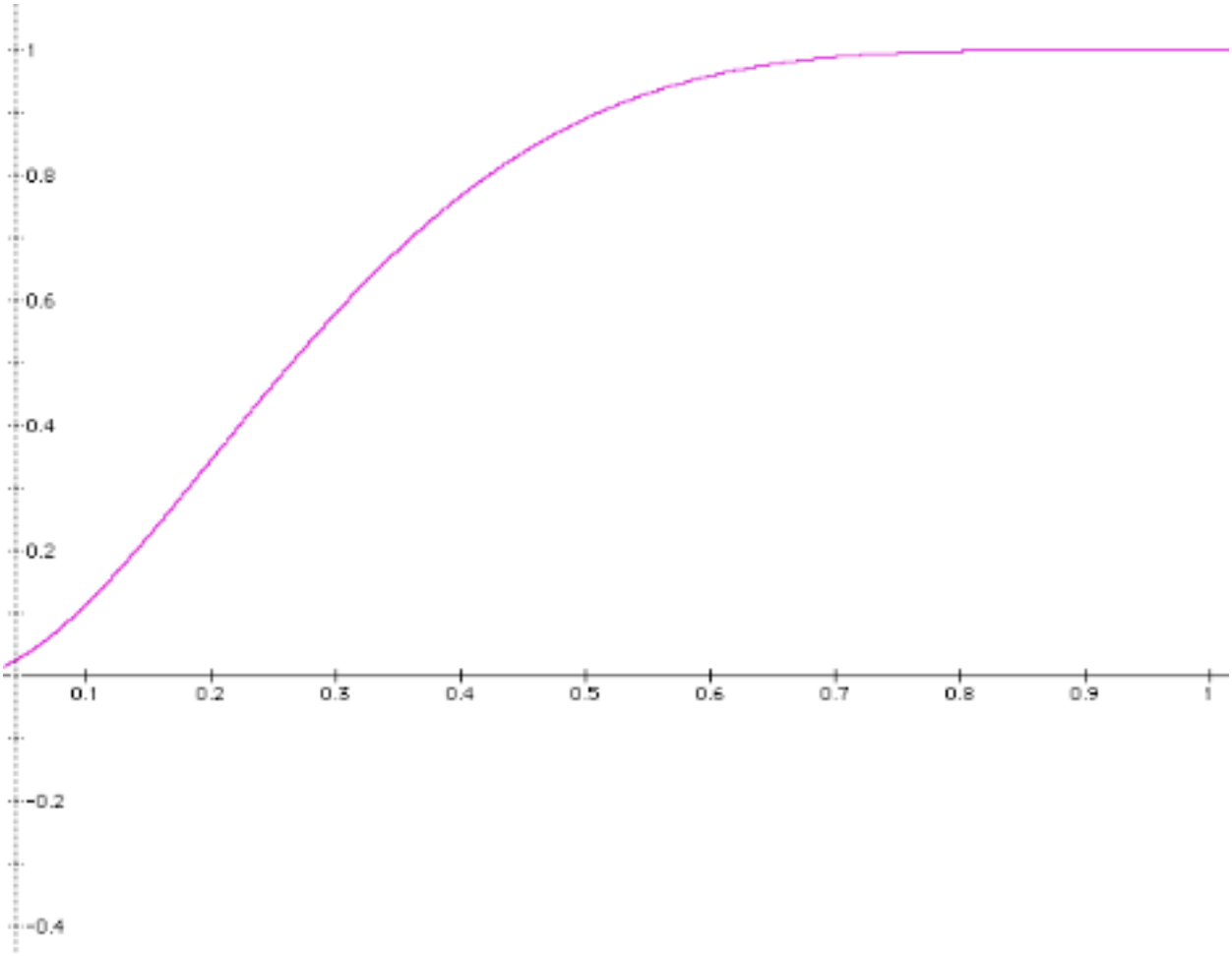
which, when graphed looks like the following: [see next page]



Which is by no means the behavior we saw in the function given in class (that was estimating at least 90 out of 100 hits), which looked like: [see next page]



And while it may seem as though the new function is much worse, if we take a look at the old function plotted against a data set of the same size (2 or more out of 6) it looks like this: [see next page]



Note: the preference that is given to the upper end of the resemblance scale for the function we gave for small samples sizes. The original function, which was dependent upon r, just could not hold up to small sizes well. The one after the second hash is much less likely to accept more than two values for anything except a fairly high resemblance.

If one was to hash values to a 16 bit hash table or an 8 bit hash table there there would be a higher chance of various numbers hashing to the same hash value [numbers that aren't the same] and this would produce more false matches. Going from 64 bit to 16 bit hash is a factor of $1/2^{48}$ as much information. Going from a 16 bit hash to a 8 bit hash is 1/256 as much information. As such, there will be a dramatic increase in the probable errors and false matches with each smaller size hash table.

4. a) Prove that 636127 is composite by finding an appropriate witness.

The easiest way to see if this is composite, assuming that it's not a Charmichael number, is to test with Fermat's little theorem for various values of a. When testing with a = 2, we immediately failed because

$$a^{636126} = 195504 \bmod 636127$$

Therefore 2 is a witness to 636127's compositeness.

b) The number 294409 is a Carmichael number. Prove that it is composite by finding a witness. Briefly explain why Fermat's little theorem won't help.

For 294409, Fermat's little theorem will fail because it is a Charmichael number and, by definition,

$$a^{n-1} = 1 \bmod n$$

for all a when n is a Charmichael number. Therefore, we try to look for nontrivial square roots of 1.

So letting,

$$294408 = 2^i \bullet u = 2^3 \bullet 36801$$

gives us a scheme. We can, for values i = [1, 3] test to see if this is ever not equal to 1 when done mod 294409. Sure enough, for i = 1, we find

$$2^{2^1 36801} \bmod 294409 = 2^{73602} \bmod 294409 = 147204$$

so 2 is also a witness to 294409's compositeness.

5. RSA public key is: (469478487497204305296287390 81, 372674862636792350620645369 73)

Convert the message Give me an A into a number, using ASCII in the natural way. Encode the message as though you were sending it using the RSA key, and write the corresponding encoded message in decimal.

m = "Give me an A"
n = 469478487497204305296287390 81
e = 372674862636792350620645369 73

Converting message to bit stream,

'G' = 71 = 01000111
'i' = 105 = 01101001
'v' = 118 = 01110110
'e' = 101 = 01100101
' ' = 32 = 00100000
'm' = 109 = 01101101
'e' = 101 = 01100101
' ' = 32 = 00100000
'a' = 97 = 01100001
'n' = 110 = 01101110
' ' = 32 = 00100000
'A' = 65 = 01000001

So message converted to bit stream is,

b = 01000111011010010111011001100101001000000110110101100101001000000110000101101110001000000010000001

The bit stream converted to a number is,

x = 221009320130778009770266542 73

Which, being less than n, means it doesn't have to be broken up into smaller pieces.

Encrypt with RSA

$(x)^e$ = $(22100932013077800977026654273)^{372674862636792350620645369 73}$

$(x)^e$ % n = $(22100932013077800977026654273)^{372674862636792350620645369 73}$ % 469478487497204305296287390 81

= 270167643401181923957124923 78

So the corresponding encoded message for "Give me an A" in decimal is.

270167643401181923957124923 78