Dr. Richard W. Hamming's Transcript for 21 APR 1995 Error Correcting Codes Lecture held at the Naval Postgraduate School

There are two lectures I particularly like in this course. This is one of them. The other is the last lecture in the course. Both of them are concerned with teaching you how great things are done. Most of the course is on this, but this particularly does it because you know I invented error correcting codes. You know error correcting codes are important and I'm going to tell you, as best as I can, how it happened.

In order to bare some conviction that I'm telling you right, I will back up and tell you at Los Alamos, during the war, I met people like Fineman, Metropolis, Oppenheimer, Beta, Teller, Farame, I saw that I was what you would call a janitor of science. Yes, you have to have people that do these things, but I certainly was not one of the ones who really mattered. I just kept the machines going. I was envious. I went to Bell Labs, I saw the same thing that what I had done had been routine work. Not bad, but not great. I started studying, even at Los Alamos, what was the difference. What was the difference between Fineman and me? Why did he matter and me not? I started studying the subject very carefully. So, when I came to the point where I was creating error correcting codes, I was already extremely interested in the matter. Now, I also knew one other thing; when you are up at bat, you think about hitting the ball, you don't think about how. That's style. That you don't worry about. You practice style other times the same way when I was doing research, I did it. But as so as it was over with, I stopped and asked myself, "What happened?" I was already in that habit. How did it happen? What happened? Now, that does not mean that I tell you the truth. But by putting it into words very soon afterwards, I pretty well froze the situation. So, I can tell you pretty much what happened at the conscious level and a slight probing of the unconscious, but I truly can not tell you what happened inside myself because I haven't any clue as to how the human mind works. So, I'm telling you, in some sense, a superficial view of what happened.

Now I was using this relay computer in New York, two out of five code, built by Snippets, well, by Andrews and Samuels, but Snippets had started this stuff, and this was for Aberdeen, and the machine had a very nice feature since it was error detecting. If an error occurred anywhere, two relays were not up, the machine would halt and retry. And the circuits were built so it would try three times and if it could not get it in three tries, it concluded it couldn't get it and it would drop that problem and automatically pick up the next problem.

Well, in those days, I was low man on the totem pole. I was just a visitor coming in trying to use the machine for some useful work, and I didn't get much done. But, the machine would run all night. So, Friday afternoon late, they would mount a tape of a whole bunch of problems of mine, and the machine would work Friday night, Saturday, Saturday night, Sunday, Sunday night and Monday morning, I'd have all those answers. I'd take 'em back Tuesday to my friends at Murray Hill, and I would give them the answers. So, I was getting a great deal of computing done, even if during the week I didn't get much, I'd have the whole lovely weekend.

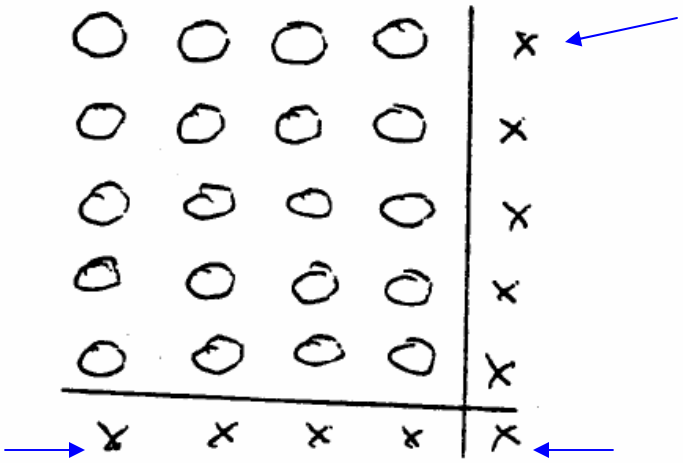
I came in one Monday, something went wrong and all the problems were picked up and dropped and I had no answers. I'd go back Tuesday and tell my friends, "Well, hey, it happened, tough. I'll get 'em next week." Next week the same thing. No answers.

Well, I am, moderately safe, provoked. Strongly, I'm very annoyed! And I say, in slightly more polite words, I'm gonna use, that I'm actually gonna say, I said a little stronger

than this, "If a machine can find out if there is an error, why can't I find out where it is?" Because if it could tell me where it is, I could change the bit from a one to a zero, or, a zero to a one and I'd be in.

I want to stop. Notice one essential feature. I was deeply emotionally involved. I was aroused. It is characteristic of great things that the person involved is deeply emotionally involved. If you are simply, as many of my friends at Bell Labs were, content to just do things well, they never did anything great. The great stuff comes from people who care and care passionately. And, if you're going to be going along, "Well, it's a good career, bing, bing, bing; you probably will not do anything great. If you care greatly you have a chance of doing something. So, the first thing is emotional involvement. Well, let's go on.

I had already, come years before, discussed the question of machines that could correct their errors. Namely, build three machines, and build circuits to compare them and take the majority vote. Now you don't doubt that this is the answer. So, my question is, "Why can't a machine locate where there is an error and correct it." I immediately knew it could. But who wants to build three machines with inter-comparing circuits? Is there no better way? So, that's the first question. How can I build a better method than three copies? Well, you remember last lecture, I talked about parity checks. I had looked into parity checks extensively. I knew them intimately, and it wasn't very long before I said, "Well, if these are the information bits and I put 'em in a rectangle, I put a parity check there ( see Fig. 12-1 ) across these rows and down these columns, and if you want, this one also, it doesn't matter, then if there is one error, I will have the row, I will have the column, I will have the coordinates." Pretty clever, right? Now, the



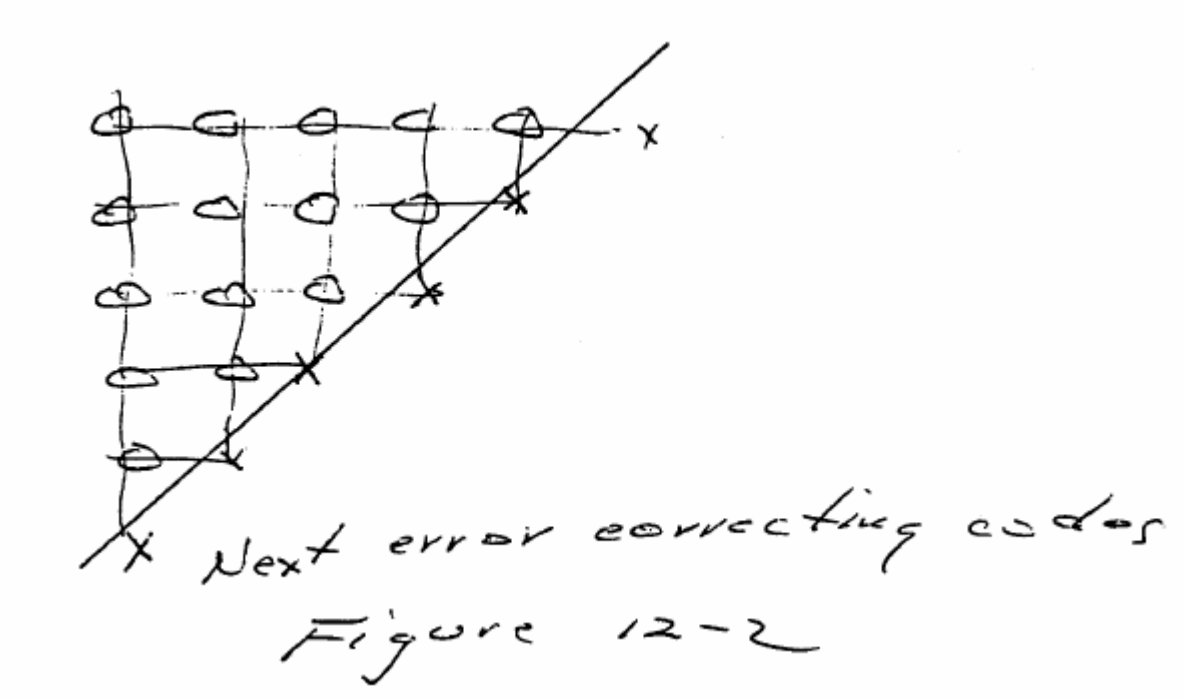

First error correcting codes.

Figure 12-1

closer this is to a square, since you've had calculus, the better. A long lean one will have too much perimeter right? The square will be the best. And, I went through kind of things like,

"Well, if I have forty three and I add a bunch more dummies, I can bring it up to forty nine, that's 7 X 7 and I can get a nice square. So, I was aware of these kinds of things and I was pretty content and I had written what we call a pink, on how one might do these things. Now, notice it fits my rule that I told you several times and I'll repeat again several times, "Luck favors a prepared mind." I had gone through a lot of parity checks and you, I think can see, where a knowledge of binary and a knowledge of parity checks; this is not a profound invention. The calculation of this is the same, so long as I use even parity checks, is not very hard.

Well, next step. In order to get to New York I could either take the train in in the morning, or, I could come to work about a half hour early, having reserved a seat on the company limousine, which went in to transfer mail from one place to another. I got to ride comfortably, using the front seat next to the driver, across north Jersey through the tunnel, up to the West St. building, I'm in. So, I'm doing it. Now, two things. One is, I'll tell you again and again. The story that you should look at mistakes and learn from your mistakes is basically wrong. You should review your successes. Because if you study the mistakes, when your chance comes, you'll know how to make a mistake. If you study success, when your chance comes, you'll know how to make a success. Secondly, there are so many ways of being wrong and so few of being right. It's easier to study success. So, I'm running through my mind, these kinds of things (pointing to parity check blocks on the whiteboard), and the north Jersey scenery isn't exactly something you want to look at. So, I'm running this stuff through my mind, reviewing my successes, and out of nowheres, comes the idea: if I put them in a triangle, and if I put the parity checks just along the diagonal, I will have less perimeter for area.



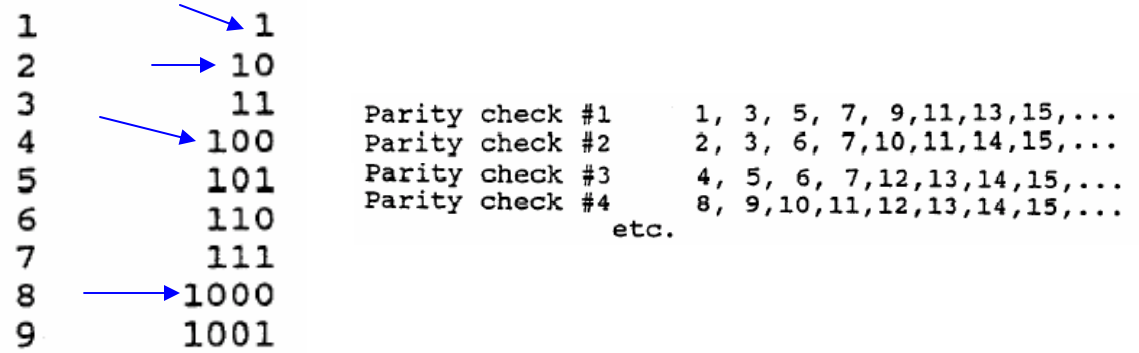

Now, I've looked in vain for whatever made me think of that. Why, reviewing a thing, did I think a triangle would be better? Well, I am humiliated. I had been telling myself how great it was and how a square code would be the best possible. Here is a better one (points to the triangle code). Right then, I say to myself, "Hamming, you're going to have to find the best

possible code. No more finding a good one thinking its real good, the best possible." Well now, if I go to a cube and if I check all the ones across that plane, by this parity check on this line (pointing to the right front vertical of a cube) and I check all the ones on planes that way (waving to and fro) on one edge and the ones that way on one edge (waving back and forth), I will have the three coordinates. And so roughly, a cube n x n x n, I'll have 3n check bits for n^3, which is a hell of a lot better for 2n for n^2.

Hmm, I'm a mathematician. If three dimensions are good, four is better. Now, I don't have to build the thing in four dimensions, I just have to wire it as if it were in four dimensions. Well, four is good, five must be better. And, shall we say, ten miles, and I've come to the realization that 2 x 2 x 2 x 2 is really good. A nice cube. There will be n + 1, one along each edge, on a 2 x 2, and one in the corner and one on each edge. So, it's n + 1 check bits for 2^nth bits in total and n + 1 check bits. Okay, sounds real good. We're now about up to the Lincoln tunnel. How I prove it is best? How I prove it is best?

Well, one of the methods of dealing with those kinds of problems is counting. How many syndromes can I produce with n + 1 check bits? If I got n + 1 check bits, I can produce 2^(n + 1) various numbers from the n + 1 check bits. On the other hand, how many do I need? I need 2^n for each position in the cube plus 1 for the right answer. I'm off by a factor of two approximately, right? I have not got the best code because you could see why if every one of these patterns, which I will call the syndromes, whatever pattern of check bits come up, if one pattern represents one position and one position represents one pattern, that must be best. And if I haven't got it, it can not be best. Well, the car arrives at the door of Bell Labs, I get out, I gotta go in and sign in, and I've got to go to a conference and that's the end temporarily.

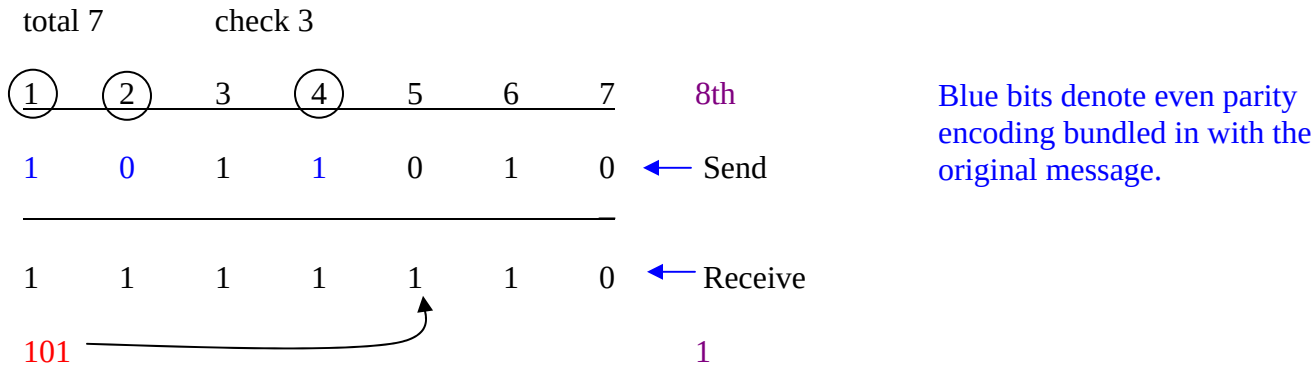
Some days later, I get back to it. What am I gonna do? Obvious, after some thinking! If I make that syndrome tell me what position it is in, the syndrome shall name the position. The syndrome, remember, is a pattern of illness, it's the pattern of bits coming up with all zeros, meaning, the correct answer. Otherwise if an error in position six, the binary number six (110) shall come up. You can't beat it, right? How do I do it? You think a bit and it's perfectly easy. Look at the binary numbers:




Every time there is a one is this column, the first parity bit must come up. In other words the first parity bit must be 1, 3, 5, 7, 9… n. The second parity bit must be where one is in that column (points to the next one from the left) 2, 3, 6, 7, 10, 11… n. The third bit, the third parity check: 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 and so on. Is it clear, that if I'm going to have the

syndrome name the position, those positions that have a one must trip that one (the first column), those positions that have a one (points to the second column) must trip that one, and so on? So that must be the rule.

Alright, let's do one. Take it out of empty space and look at a careful case. If I choose the most degenerate case, three, I simply send a bit three times and take the majority vote. So, I'll take the next one, seven. Check bits, three. ( He drew on the whiteboard: )




Those are the seven positions. Now, my check bits. These are my parity checks right over here. ( Points to the block code – cube ). 1, 2 and 4, I'm going to pick as the check bits, because they have the property that they are set independently of anything else. Any other choice, one of the numbers may occur in several places, right? And that may cause some trouble. Okay, I've got four message positions. What messages do you want? Four bits. What will you have? Give me, 1010. That's the message, okay. I've gotta get it to you. First, I've got to put the check bits in. 1, 3, 5 and 7, I've got to put a one in, right? Correct? 2, 3, 6 and 7, I've got to put a zero in. 4, 5, 6 and 7, I've got to put a one in right? Correct? Huh? (Student asked a question about even parities). Yeah, even parities.

Now, I'm gonna change, say, the fifth bit. So, what is going to be received by you is 1111110. What do you do? You calculate 1, 3, 5 and 7? No! 1, 3, 5 and 7 is odd. 2, 3, 6 and 7, yes. 4, 5, 6 and 7, no. What binary number is that (101)? Change the fifth position. Strip off the check bits and you got the correct message. Right? If you look at whether it's these bits or this or this or that, it does not matter in difference at all what place you put the error. I will always be able to correct the error. I send it to you along with the parity checks. You receive them, you apply the parity checks. If there is a single error, that's where it is. Simple? Obvious?

Now, let's say some things. Whether I use zeros or ones, in any code developed, whether I replace ones by zeros or zeros by ones in any one position, it's the same code. If I change the columns around, so long as you and I agree on the change, it's the same code, right? So you might very well want the 1, 2 and 4 to leave first and the 3, 5, 6 and 7 to be sent later on. This particular code is one realization with a very acute property of this: the syndrome names the position. But there's a whole bunch of equivalent codes which are the same code with trivial changes. And, you might very well want, for practical engineering, a slightly different code.

Now, this is about as best as you can do. Now, that doesn't look good there, but if you look at a message of total length 2^n – 1, I have n bits. And so I have a 2^n – 1 – n message. Alright, now if I take n = 10, I have (1024 – 11 = 1013) for n = 10. For 10 bits, I can send 1000 message positions. For three and four, it's unfavorable, but I have to write smaller codes. For a big code it's evidentl; I get a very favorable excess redundancy. I've only added 10 extra bits and covered a thousand. Right? So, the one I put here is misleading. It's too easy and too simple, but I don't want to write out 1023 positions. I'm not about to do it in class. But you see why it supposedly works. But, if I go to 1023, of course, a double error.

Now a double error in the rectangular code, let me show you what happens. Now if I had a double error here and here (points to middle of rectangular block code), I would get those two, or those two (points to vertical and horizontal pair). I would not know if it were that pair, or this pair. Now if two of them were aligned, I would know the two rows, but I would not know the column. So, this could not cope with a double error. Nor can this (points to working example). If there's a double error, it will fix the wrong position and put in a triple error. So, you ask yourself what you could do and Hamming says, "Well," being familiar with the binary system he say, "put in an eighth position which is a parity check over everything. Well, now the situation is very easy. If the syndrome is all zeros and this is zero (points to overall parity check), the parity check is zero, you're in. If this is all zeros meaning that no error occurred there, but if this is a one, the error must have occurred in the eighth position. If there's a pattern here and a one there (points to the eighth position parity check), there must be one error. If there's a pattern here and a zero there, it must have been a double error. Therefore, for one extra bit I can detect double errors, and, it's a natural thing to add on. Same way over here (points to the block code). The same argument will apply. One parity check over everything and I will know whether it is a double error or a single error. So, one extra bit and I buy double error detection, which is probably a prudent thing.

Now, let's not get confused about how fast I did these things. I'm going to give you some more about what I did. I do not know length of time, although I suppose it could be dug out if you really wanted to. My impression is that something I could read six months time. Remember, I was part of Bell Labs team doing things. I was running computing numbers for other people, I was programming; I was doing all kinds of other things. I had a job as a member of a team to do. This work was by and large done at home, at odd moments, and that's how you first get started. When you hire a plumber, you don't hire the plumber to learn on your time, you hire a plumber who knows. And, when you're first hired as a researcher you're told pretty much what to do. When you've demonstrated your ability to do research, then they give you freedom. And, it's in some sense, ridiculous. Again, jumping out of order, consider the situation of myself. During this time, I was loaded with many many other tasks. By the time the error correcting codes appeared which was delayed for eighteen months or so by patent troubles, and I gradually became famous, management gave me a bigger, freer hand. And when, after 30 years at Bell Labs, I had the corner office with windows in two directions, a rug on the floor, a secretary, unlimited travel expense and no assigned duties. But, I didn't have anymore ideas!

When you're young is when you have the ideas and that's when you are burdened with other things. The ground rule, as I say, is very simple. You must, on your own time, demonstrate greater ability and when you have demonstrated that, they will give you the freedom

to do it. But they won't, by and large, give you the freedom to do. For example, it's no use going to your boss and say, "Hey, I want some time to do some research," because I remember what happen when I was a graduate student at Nebraska. An instructor went to the head of the department said he'd like to be relieved of some teaching so he could do some research in mathematics. And the department looked him in the eye and says, "When you've done the research, I'll relieve you of the teaching." That's the way things are. And that's the way you want them, don't you?

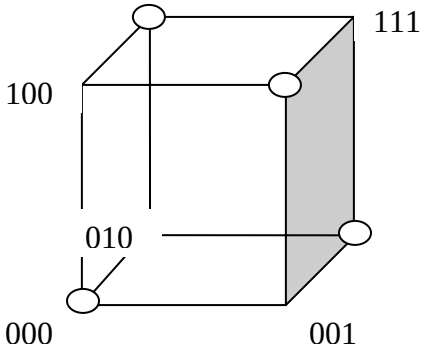
So, your first successes must be done by this extra effort. It's doing more than is needed at all times and finding when you do get some successes, then you get more leeway apparently. Of course, as you rise up the organization chart, you get more duties. That's where I was smarter than most people. I saw very early that Bell Laboratories would ultimately have a vice president in charge of computing. When I said that, people said, "Na, you're crazy because computing's just a minor thing that we're able to do." But, when I left, there was a vice president. And somewhere along the way I said to my wife, "Eh, you know dear, I could be vice president at Bell Labs if I wanted to," and you know all the perks that comes along with being a vice president, you know, limousines and all kinds of other things and fancy this and that. I said, "You really think this is going to matter?" She said, "No." I said, "Neither do I. I'm gonna avoid promotion." And so I spent my life avoiding promotion. It was a bit of a struggle, but I did it pretty well, although, several times they promoted me in emergencies, and I had to get out of it somehow or other when the emergency was over and remind everyone, "Hey, the emergency is over, I want out," cause, they were gonna leave me there.

I can not answer the question you may ask about that point. Would I have been more valuable to society had I risen to be vice president, since I thought then, I still do, that I was practically the only one who understood the role of computers? Was I shrinking my responsibility, or was I playing to my greatest strength, namely, originality. I don't know the answer, but, occasionally, once in a while I think, "Uh, maybe I should have done the other one." Not that I want to be vice president, but maybe I had some obligations to do that. And, the same way with you. As you go up the line, although you think you have more freedom, it seems like it, there's more darn things to be done and it's not clear you have more freedom at the top as you did at the bottom, it just looks like that. Well, let's get back to this business.

Now, I've got the parity check alright. Now, you learned one other thing. When you took analytic geometry you learned that you could look at a problem algebraically or geometrically, and what I've given here is basically an algebraic approach. The fact is that after a little while they were done somewhat in parallel. So, let me draw some of the geometric aspect. A distance function $D(x, y) >= 0$ between two points has got to be positive. If the distance function $D(x, y) = 0$, then $=> x = y$. If there is no distance between two points, they must be the same point. $D(x, y) = D(y, x)$. The distance from here to there is the distance from there to here. And, lastly, $D(x, y) + D(y, z) >= D(x, z)$. If instead of going there, I go someplace else, and there, it has to be at least as far as going there directly, right? This is what you mean by a distance function. This is something that I learned while taking abstract algebra.

Okay, I am gonna look at a geometric picture now. Here's a cube. Now, I've actually drawn a two dimensional figure, but you think it's three, cause you can see it. You assign the

coordinates all the way around. Now going from one point to another is making one error (from 000 to 001). If I go along this edge and that edge (from 000 to 001 to 111) I have made two

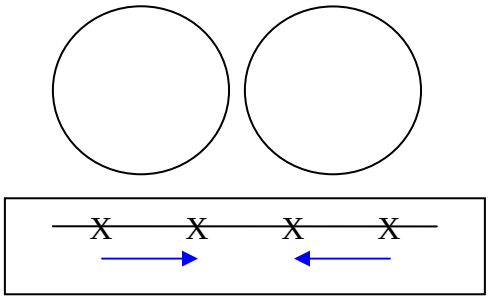



cube with vertices labeled accordingly

| Min. dist | |
|---|---|
| 1 | unique |
| 2 | error detection |
| 3 | 1 error correct |
| 4 | 1 correct + 1 detect |
| 5 | 2 correct |
| 6 | 2 correct + 1 detect |
| 7 | 3 correct, etc. |
| | \| |
| | \| |
| 2k + 1 | k error corr. |
| 2k + 2 | k error corr. and k + 1 detect |

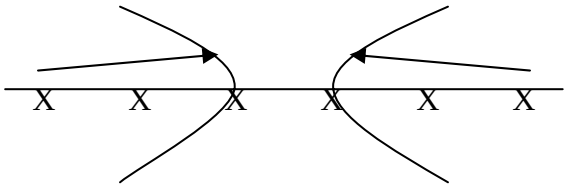
changes, right? Well, Hamming says, "The distance is the sum of the differences (Dist = sum of the differences). Not the sum of the squares, ala Pythagoras." Pythagoras says the sum of the squares of the differences on two sides is the sum of the square of the total distances. Hamming says, "No," in what we call L1. It's a well known mathematical formulation which is abstract. This is the true difference. Well, if I put a message here (at 000, 011, 101 and 110), then, I say, it is evidence that every point is two away from every other point, and I can draw a sphere of radius one. What do I mean by a sphere of radius one? All points at distance one away. A sphere of radius two, all points two away. A sphere of radius three, all points three away, right? Well, if I had the property that the minimum distance, if the minimum distance is not one away from two points, I have two different meanings for the same point and I'm stuck, right? If the minimum distance is two, I have error detection. Because if the distance is two away, if a message comes to you with one mistake, it's no longer a legitimate message, correct? Right? Now, two of 'em, you're dead. But one, you're right.

Okay, now we go to three dimensions. Minimum distance three. If the minimum distance is three, and the spheres don't overlap, one two three is the distance across, I've spread out the distance so along the line, one error will carry me there, or one error will carry me there.



If the minimum distance is three, and you receive a message with one error, it is closer to one particular message where it came from than any other one. It is error correcting, correct? If the minimum distance is three, one move away will leave it inside the sphere of radius one. From the other one, one sphere of radius one, in or on the third. Therefore, a minimum distance of three tells me that if I can pack the spheres in "n" dimensions, with that distance function, so the spheres don't overlap, I have error correcting code, if, and only if, if the distance isn't that much, then one would carry another one and you wouldn't know which one it was.

Well, distance four. (Notice the min. distance graph on previous pate.) Five, two correct, because if the things are five positions away, a sphere of radius two and a sphere of radius two, will have no points in common. If you get a message from me and there are two errors and they're all five away, you can determine which sphere it lies in and which the message was, correct? Right?




Double error correction requires a minimum distance five. It requires it and it is sufficient. Six, two correct plus a detect. Seven, three correct, and so on. Whatever amount of correction you want, if you want "k" error correction, I must have those spheres of size 2k + 1 about the message points with no overlap. It's a problem of finding non-overlapping spheres in the space with that crazy distance function, correct? So you see, the higher order correction codes can exist up to any amount you want. How to find them is another question. I answered the question for those (points to 1 – 4 in the minimum distance list). For minimum distance three and minimum distance four, that's one correction and one detection. That was the extra bit added on with this distance function, L1, the sum of the differences. So you see the geometry and the algebra is there. Now I can give you how many spheres there are. The whole space has 2^nth points, there's a center; there's a binomial coefficient n1 + n2 + nk, that's the volume of a sphere of radius k, that's the total one. Divide 2^n by C(n, 1) and you must have an upper bound on the number of spheres. Because C(n, 1) is the volume of one sphere, 2^n is the total volume, divide

$$\frac{2^n}{1+C(n,1)+C(n,2)+\ldots+C(n,k)} \geq \#\,of\,spheres$$

the total volume by the number of spheres. I can not do better than that.

Now a perfect code is one in which every point is in some sphere and once you get beyond the ones I gave you there, there are very very few, in fact, only a couple of perfect codes where you get every point. So, there are some losses as you go higher. On the other hand, you do better. Higher error correction, shall we say, it defeated me. I tried and couldn't find any regular method. I could create isolated codes with a minimum distance, but I could find no

regular method of doing it, but then, let me remind you of another thing I'm gonna tell you. I told you frequently I managed my career. I had observed, already, people like Einstein. He had a lot of good ideas. He had a unified field theory, and for the last half of his life he did nothing. I saw Sharon, other people. I saw that a great many great people do something great and they spend the rest of their life on that thing, and they do nothing creative after that. They add, elaborate, and so on, and they become the great famous name for that, but they really do nothing else. Knowing these things, once error correcting codes were reasonably launched, I told myself, "Hamming, you are not going to read papers on the subject of error correcting, you're not going to write papers, you are not going to do anything. You are going to try and go out and do something else." I did that. I had the nerve to try that.

Now it takes nerve. You're a great expert in a field. You're the name and you're going to abandon the field to somebody else and to start back where you know nothing. It's hard to do, but if you don't do that, you will then do one great thing and that will limit you, and it's very very characteristic of great scientists. They do a great many good things and they finally do one great thing and they spend the rest of their life elaborating that. So, I didn't. I managed my life by constantly and deliberately changing. I'm telling you how to do it.

Well, now let me talk a little bit more about this L1. There's another one, L(infinity), which is in (min, max) difference. Take the maximum difference of any two coordinates and take the least of that and that's the maximum difference. I wanna be outrageous. I said the other day exactly the same thing. I'll be outrageous again on the same speech. Pythagoras was the first great physicist. He found we live in L2; that the sum of the squares of the sides gives you the square of a diagonal of a rectangle. Hamming, in some vague sense, is Pythagoras junior. He says, "Yes! Pythagoras is right about the physical world. No, about the mental world." The difference between two strings of bits is the sum of the differences. It's not the sum of the squares. This is the correct way, or, that ( points to L1 and L(infinity) ) and I'll give you some examples.

Somebody is coming down the street. It's a child. Is it your child? You may say, "No, she's too tall." You are using L(infinity). One difference is sufficient. Alternately you may say, "Look at all these differences. There's six different differences. That why she's not my daughter." You're using L1. We have found, particularly in artificial intelligence that L1, the sum of the differences of two patterns, is a much better measurement than L2. Sometimes L(infinity). One single feature is sufficient to tell you it isn't that. So, what had happened here is quite simple. Now, I leave it to you to think about the story. How difficult was it for me to say, "Well, I don't believe in Pythagoras. I believe in this crazy theory I heard about in some algebra course that I really want to work in L1, the sum of the differences." Well, when you say it; obvious. But, before you say it, how obvious? Well, not so easy.

There were some other things I told you; not so easy. But, don't get the impression that what I've told you in, well, actually self, without the frills of how I did it, something like twenty five minutes. And, that's shall we say, took uh fifty days, or maybe one hundred days. Thinking odd moments here and there and odd blundering this way and that way, but coming back, again and again to the idea until I had what I considered is the best possible proof. But I also had a failure. I could not find systematic methods of creating two and three distance codes. Now

those are ones that are used heavily now. Now I did find, although I didn't publish, what are called burst codes. You simply break the bits up into a burst because lightening strikes and a whole bunch of bits are out, or there's a scratch on the disc; the whole bits are out. Well, what you do is separate the bits far apart and you put them in codes separately and you encode them error correcting codes and any single one of those far separated bits will be corrected. So, there's a string of them and as long as the string of errors is not bigger than what you are breaking up, you'll be able to find all the errors.

Some of the consequences. Take the problem of digital music. A Japanese president of a company said to his employees, "I want digital music because it can correct errors and get it high and what I demand is this particular symphony that my wife and I like. It runs one hundred and twenty minutes. There shall be no errors because one error could produce a blip. No errors in a hundred and twenty minutes." They put in a Hamming type code with frills to get around burst because there might be a scratch locally, and that's why you have digital music. Simply, the idea of keeping the signals far apart and coping with noise.

It has had profound representation everywhere. Your discs write in error correcting codes and pull them back off in error correcting codes. You don't know it. You don't need to know it. We've built machines error correcting so they can correct their own errors and go ahead, which is what you want. When you put a vehicle out on Mars, to run it with a computer, you can't send a repair man out. You want these kinds of things and this is how you do it. And the underlying theory is this geometric picture which is easy to understand. Those minimum distances are necessary and sufficient. How to find them is a very elaborate and difficult process which there is not time to go into. But, I say again, I did it in three to six months. I don't remember just how long. I could look it up, but let's say it took me six months is odd moments. What the hell? There's weekends, there's evenings… there's lots of odd moments. While you're cleaning your teeth. While you're walking to work. There's all kinds of odd moments.

It isn't so much to have done that like the time. As I've told you, I was gonna do. I willing to stick my fingers in any one of your faces and say, "Are you prepared to stand up and say that given the position I was in, and given the extra effort I made, that you could not have done it? Are you willing to point to any one thing you could not have done?" Bunch of chickens. That is saying, effectively, that you are just a capable as I am. The difference is that this phrase I've used several times, luck favors a prepared mind. I was prepared. I had looked into parity checks. It was not surprising I could think of this rectangular one. When I found it wasn't best, by that chance, I can not account for it, why the triangle? I can not account. But once that was out, then, only honesty is all. What will be the best? It's not sufficient to find a better one. "Hamming, you've got your pride. What will be the best code?"

Well, once you phrase it that way, you were driven right down the path that I want to cross here to exactly those parity checks and this method of doing it. Now, the method is not exactly new. It was, in fact, on a top of some breakfast food cards. The same kind of a code. You pick out the card which had your birth date of the month on 'em, and you hand it to the other person. Its various dates around here, lots and lots of dates. All he did was look at the binary encoding and give you the date. That's what I did. Same kind of code. So, it really was not new, but I didn't know it and I had to, sort of, invent them. I had to do these things. It was,

I'll summarize the points: Emotional involvement first. If you don't care strongly, you're not likely to do really great things. Secondly, prepared mind. You did more than the minimum so that when luck came around, you knew how. Another rule I told you, study successes. Not only your own, but why did Galileo do what he did? How was Newton led get to it? How did General so and so do such and such? The examination of successes will prepare you to succeed. The examination of failures, well, I don't say that you should never look at failures, will, by and large, prepare you to fail. So, furthermore, as I said, there's so many ways of being wrong and so few of being right, it's so much more economical to study success.

Now I'll give you part of the same lecture; the last one in the class. Meanwhile, I will tell you a very simple thing and we'll call it class off. It's a very simple observation. Nobody ever told me the things that I've just told you. I had to find them for myself, because I was aroused that I was just a janitor of science who would never amount to anything beyond, eh, routine. Well, I published twenty papers, or something else, but none of them would have mattered. I decided that I wanted to be something different. I set out to do it; I did it. And, if it wasn't error correcting codes, there's a Hamming window out and there's various other things lying around that were named after me, so, it was not a matter of luck. Yes it is, but, unfortunately for your argument is luck, all luck. Einstein did too many great things. Hamming did too many good things. Various other people did too many good things. Shannon did Boolean algebra as well as information theory. Most great scientists did more than one thing, although, some of them that I told you, sterilized themselves by once they had out a good idea pursuing it forever. It wasn't that they couldn't; it was they did not manage themselves. What I am preaching is, me having told you how to be great, you have no excuse! But you must manage yourself. I have an excuse for not to amounting to anything because nobody ever told me. I've told you in detail how it was done, and you've been afraid to say, "No, I'm that smart." What you have to say is, "Either I didn't have the energy; I didn't do this or that." But luck, there's too many opportunities around everyone that you will not find one or two to seize and become great. And once you become great, you get the corner officer, the rug on the floor, the secretary, the unlimited travel and everything else. But, of course, I told you, in research, by the time you got it, it's too late.