THE BEST 9x9 SQUARE YET

ERIC ALBERT
Auburndale, Massachusetts

On December 28, 1897 the first 9x9 word square in history was published. It was constructed by Arthur F. Holt, the master formist of his time, and appeared in the Chicago Inter-Ocean, back when newspapers were enlightened enough to carry that sort of thing. Given the incredible difficulty of building such a form, it is not surprising that the square contained words taken from a variety of sources.

In the November 1980 issue of **Word Ways**, Jeff Grant discussed a 9x9 square, based on the word SLESTERED, that he considered "probably the finest" constructed up to that time, a square attributed to Wayne Goodman of Chicago. Grant pointed out that the square contained seven words in Webster's New International Dictionary, Second Edition (Web 2), a word from the Century Dictionary Supplement (1909), and the name of a small community in Yorkshire, listed in most large British gazetteers.

Grant also noted "Of course the dream of the 9x9 constructor has always been to find a square using only solid-form, uncapitalized words, drawn from a single reference." He then displayed an impressive square, culled from "many hours of wearying research" in the Oxford English Dictionary (OED). Unfortunately, the form had one flaw: it contained an inferred word. Grant's argument for the word's historical existence is convincing, but the fact that the word is not actually found in the OED weakens the square's claim to come "from a single reference."

In the February 1988 issue of **Word Ways**, Dmitri Borgmann confirmed Grant's observations, commenting that "... no fully satisfactory 9x9 word square has ever been constructed" and stated that the Goodman square on SLESTERED was "the best such square known."

On June 27, 1989, at 11:29 a.m., a better 9x9 square was discovered. Here's the story.

While chatting on the phone with Murray Pearce on December 3, 1988, I idly observed that the construction of large word squares seemed like a task tailor-made for a personal computer because you could run such a machine day and night with no expense except that of electricity. Murray, not one to let a potential logological gold mine go unexplored, suggested that I actually make an attempt to find 8x8 squares in this manner. He picked 8x8 squares because several were known to exist that used only words from Web 2, but finding such squares was extremely difficult by hand.

We made a back-of-the-envelope estimate of how long such a program would take to run to completion. Based on the most straightforward programming algorithm we could think of, our calculations showed that it would take the fastest supercomputer in the world 100,000,000,000,000,000,000,000 years to check for all the 8x8 squares in my database. The bad news was that I didn't have access to this supercomputer. He suggested that I not use the most straightforward approach.

The next day I decided to at least make a stab at writing the program. I began considering various algorithms, concentrating on ways to increase speed. I immediately decided to follow the formist's basic rule, well-known for over a century: start from the bottom of the square and build up. This rule makes eminent sense because English is ending-poor; that is, there are far more letter combinations that begin words than that end words. You want to attack the hardest part of the square first because this allows you to more quickly weed out unworkable combinations.

This early weeding out, called "pruning" by computer-science types, was clearly the key to any hope of success. I wanted to find all the 8x8 squares, but our initial calculation had already shown that there was no way for the program to actually examine every possible combination of words. How could we most quickly eliminate the chaff from the wheat?

The final approach I chose used a variety of pruning techniques. I'll describe one technique to give the general flavor. Suppose that we've chosen a bottom word to try to build a square on, say FLAPJACK. Before even considering the word that goes above it, we can make sure that there exists at least one database entry that ends with each letter in the bottom word. In this case, for example, if there is no database entry that ends with J, then there is no point in continuing; we already know that we will not be able to build a square on FLAPJACK. We can go on and try the next possible bottom word, thanks to some heavy pruning.

The same logic can be used higher up in the square. Suppose that we do have database entries that end with every letter in FLAPJACK. Then we can begin to look for a word to go above it, building this word one letter at a time. Say that we start with the letter B. We can immediately check to see if there are any database entries that end with the combination BF. If there are none, then we know that any word that begins with a B will not form a square when placed in second position above FLAPJACK. You can see how this approach allows us to quickly throw out an enormous number of fruitless beginnings.

Besides considering algorithms, I also put together a database of 50,000 eight-letter entries by combining various dictionaries and lists that I had on my computer. I say "entries" because many of the items in the database were multiple-word phrases such as COOL JAZZ. The entries were actually entered backwards (for example, FLAPJACK was in the database as KCAJPALF) to make things conceptually simpler. This way, the program could act as if it

were building the square from the top down.

Working part-time, I finished the program four days later, on December 7th. It was written in the C programming language and was about 1,000 lines long. Much of the programming involved working around the hardware limitations of my personal computer. This machine, an 8-megahertz IBM PC/AT clone, had only 640 kilobytes of memory, some of which was used by the operating system and my program. This meant that I could not have the entire database in memory at one time. On the other hand, the machine's 20-megabyte hard disk was very slow, and I had known from the start that the program could not afford to read from the disk very often. Coming up with an acceptable solution to these conflicting demands required some fairly sophisticated techniques that I had learned in my career as a computer scientist.

When I finished my program, I gave it a test spin on a ten-word database, and was immediately rewarded with the 8x8 square I had planted. This was a good sign. I tried the program on several more difficult situations and found some minor errors, which I fixed. That night I let it loose on the full 50,000 entries and went to bed. I was so excited that I woke up in the wee hours and went to check on the program's progress. It had processed the first forty bottom words but had found no squares. This was disappointing but hardly surprising.

By morning the program had gone through the first sixty words, with no squares found. A quick calculation showed that, at 200 words a day, the program would take 250 days to complete. This sure beat the original estimate of 100,000,000,000,000,000,000,000 years, but it was still a long time, and I would need the computer for other tasks in the meantime. That very evening, though, the program found its first square. I was hooked.

I ended up running the program most nights over the next few months. My friend Alan Frank suggested some algorithmic improvements which I implemented, and these, plus a few of my own additions, sped up the program a lot. The search would have been completed sooner but my computer started crashing every couple of nights because of hardware problems. I modified the program to make sure that no results would be lost at these times, but this still meant that many nights were wasted.

It was worth it. All in all, the program discovered a total of 749 8x8 squares, many of them based on words which would never have been tried by human formists. The experiment was clearly a success.

In March 1989, I modified the program to search for 7x7 squares. The results were dramatic. Squares poured out on the screen in overwhelming profusion. A very rough calculation showed that the program would find tens or perhaps even hundreds of thousands of the forms. There was no point to this; even if I had the disk space to store them, I'd never have the time to look at them.

I modified the program again, this time to only look for squares whose bottom word contained many unusual letters. Even with this

constraint the program still found hundreds of squares, but at least they were a lot of fun to look at. I also added a feature that let me type in a base word to use for a square. My friends with seven-letter names started getting unique presents.

7x7 squares were clearly too easy, and I'd already done the 8x8s. My thoughts turned to 9x9 squares. I knew how had been constructed throughout history. I knew that only three people then alive had ever successfully constructed one. I knew that the average 9x9 square used terms from many different, obscure sources.

It was this last fact that worried me. I had a database of 63,000 nine-letter entries, but a large majority of them were from one work: Web 2. Even worse, I by no means had all of the entries from Web 2 -- my database was woefully deficient in many regards. Finally, Web 2 is a big dictionary, but it is just one of many big dictionaries, and the nine-letter entries in almost all of these dictionaries had been added to formists' lists decades before. The formists had also gone through medical dictionaries, legal dictionaries, gazetteers, and so on and on. Their lists made mine look puny by comparison.

I was left with two burning questions. Could there be a 9x9 square hiding in my subset of Web 2? If so, why had the formists not found it in a hundred years of searching? For the first question, I turned again to Alan Frank, who is a brilliant mathematical estimator. Using some arcane techniques, which I suspect involved guessing, he informed me that there was a fifty percent chance that my database contained a 9x9 square.

This was psychologically a very different situation from searching for 8x8 squares. In that case, I had known that I would find some, if I was willing to wait long enough. This time, I didn't know if there would be any reward at all. I did not want to baby my lemon of a personal computer through a half year of nightly searches. I was not looking forward to modifying the program again, either. Going to 7x7 squares had been easy; the database was smaller and thus required less disk space and memory. Going to 9x9 squares would be the reverse, and that meant struggling once again to get the program's compromises just right.

What finally convinced me to take on the task was my new computer-science job. Or, to be more precise, the 20-megahertz IBM 386 clone with a quick 90-megabyte hard disk that sat on my desk. I figured that by running the program all the time at home, and every night and weekend at work, I could drastically reduce the total time this project would take.

I began modifying the program once again on May 31st. On June 4th, I started up the new version at home, and the next evening, as I left work, I started up another copy of the program there. Each copy of the program was given a different place in the database at which to start looking for possible bottom words.

Weeks passed. My work computer performed flawlessly, but my home computer moved even further along the path to total flakiness.

It failed now in the majority of nights, sometimes only an hour after I had gone to sleep. I spent many hours patching together the data files that resulted from these crashes, files that kept track of what work had been completed in the computer's increasingly-brief moments of sanity. I began to threaten the machine with physical violence, but to no avail.

Two years have passed, and that personal computer has long since gone to the great hardware museum in the sky. I don't lament its passing, but I find I retain some affection toward the old thing. For on June 27, 1989, when I returned home from work, the following 9x9 square was displayed on its screen:

N E C E S S I S M
E X I S T E N C E
C I R C U M F E R
E S C A R P I N G
S T U R N I D A E
S E M P I T E R N
I N F I D E L I C
S C E N A R I Z E
M E R G E N C E S

Adrenalin coursing in my veins, I plowed through the three volumes of my Web 2 until I had no doubts: all the words were there! A passage from a book by Nobel-prize-winning physicist Richard Feynman ran through my thoughts: 'I went on and checked some other things, which fit, and new things fit, new things fit, and I was very excited. It was the first time, and the only time, in my career that I knew a law of nature that nobody else knew.' I did not remain alone in my knowledge for long; later that night I called Murray Pearce and we talked for hours.

One look at the square makes it clear why it had remained hidden in Web 2 for so long. MERGENCES is not FLAPJACK, but it is unlikely to show up in any formist's list of the top 50 words to try to build a 9x9 square upon. Even those who are uncomfortable with the use of computers in logology agree that no human would have found this square.

Is the MERGENCES square the best possible 9x9 square? Perhaps not. It meets two of Grant's dream criteria: all of the words are solid-form and all are drawn from a single reference. STURNIDAE is capitalized, though. I know that no other 9x9 squares can be constructed from my current database, but this database is incomplete. My arcane mathematical techniques lead me to believe that there is at least a fifty percent chance that another 9x9 square can be found from Web 2 words. Perhaps this square would meet all three criteria.

If any reader has Web 2 or other large dictionary on computer, please let me know. This time I won't have to suffer through long months of tension and ennui. Technology marches on, making pioneering efforts seem crude and faintly amusing. My current home computer can do the complete 9x9 search in one weekend.