Topics... What the Computer *Can* and *Cannot* Do

by Frances Marcello

In this age of seemingly limitless technology, one might assume that there is no job a computer can't do. But the real question is ''Can we wait for the computer to finish?''.

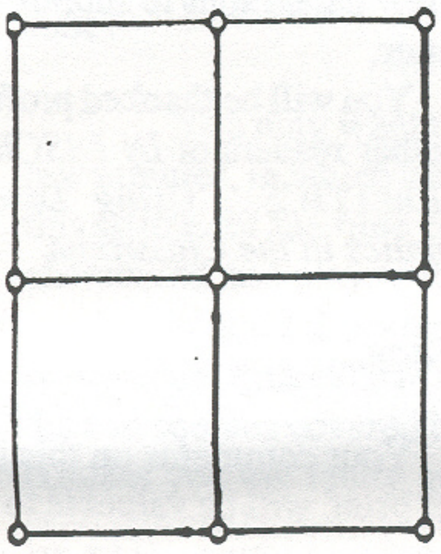
The problem used to discuss this question appears in *Dynamic Programming -- An Elegant Problem Solver*, by Cliff Sloyer et al., Janson Publications, Inc. (1987).

PROBLEM: You are given a 30 x 30 grid with a number on each edge representing the time required to travel that edge. Find the fastest path (consisting of North and East directions only) to get from point A to point B.

When I presented this problem to my students and asked for a possible solution, they immediately gave me the brute force approach -- simply find the lengths of *all* paths and then choose the smallest.

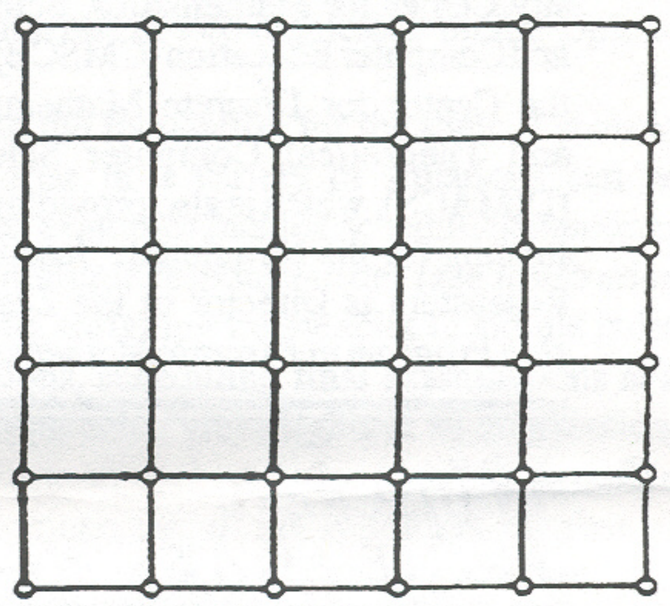
The question ''How much time would you need to calculate the solution?'' resulted in answers from 30 minutes to as long as 2 weeks. ''How much time would a computer operating at 100,000 operations per second require?'' ''No time at all!!'' was the unanimous opinion.

We decided to analyze the problem to confirm their opinion. Here are three simplified versions of the problem:



Consider the 2 x 2 grid at the left. A path here requires traversing 4 edges, e.g., ENEN. Since each path consists of 4 edges with 2 N's and 2 E's, there are ${}_4C_2 = 6$ paths to calculate. How many additions are necessary to determine the length of each path? E + N + E + N requires 3 additions. We need to perform ${}_4C_2 \cdot 3 = 18$ additions.

Now consider the 5 x 5 grid at the right. Here we traverse 10 edges, e.g., NEENNENNEE. The 10 edges of 5 N's and 5 E's produce ${}_{10}C_5 = 252$ paths. Each path requires 9 additions for a total of ${}_{10}C_5 \cdot 9 = 2268$ additions.



Now consider a 10 x10 grid. We are up to 20 edges of 10 N's and 10 E's resulting in ${}_{20}C_{10} = 184{,}756$ paths. With 19 additions per path we would have ${}_{20}C_{10} \cdot 19 = 3{,}510{,}364$ additions.

By this time students were astonished to see the number of paths and operations skyrocket as we went from 2 to 5 to 10 unit square grids. And we weren't finished. Now we had to select the shortest path!

''How many comparisons are necessary to find the shortest path?'' Again we use brute force. We compare the 1st path to the 2nd, choose the smaller, compare that to the 3rd, choose the smaller, ... continuing until all paths have been compared.

Our 2 x 2 grid requires ${}_4C_2 - 1 = 5$ comparisons.
Our 5 x 5 grid requires ${}_{10}C_5 - 1 = 251$ comparisons.
Our 10 x 10 grid requires ${}_{20}C_{10} - 1 = 184{,}755$ comparisons.

Finally we sum the total number of operations required and find the time required for our computer to finish its job.

Total operations = (number of additions) + (number of comparisons)

Time required = (total operations)/100,000 seconds

The 2 x 2 grid requires a time of .00023 seconds, the 5 x 5 grid a time of .02519 seconds, and the 10 x 10 grid a time of36.95119 seconds. At this point I can hear a sigh of relief; after all, even though the number of operations seemed large, the job can still be done in a feasible amount of time.

But then the class returned to the original problem, looking at the 30 x 30 grid. We have established some patterns we can use to help in the calculations. The grid has ${}_{60}C_{30}$ paths with 59 additions per path and will require ${}_{60}C_{30} - 1$ comparisons to find the shortest path. Carrying out the usual calculations yields the result that the 30 x 30 grid requires about 7.0×10^{13} seconds.

(Continued on Page 9)