Welcome to the Math CountS “summer 2003” column. This column I tried to start early, but alas summer activities got in the way. I would like to hear from the readers about their view of why they think math counts in CS education. Email me and I will publish a summary in the next column.

Here is a summary of some of the recent events and activities relevant to mathematics in CS and SE education.

1. Discrete Mathematics: An Early Foundation for the Study of Computer Science, a Mathematical Association of America Professional Enhancement Program workshop organized by Bill Marion was held June 2-6, 2003 at Valparaiso University primarily for mathematics faculty teaching introductory discrete mathematics courses for computer science and software engineering majors. Fifteen participants heard presentations by Susanna Epp, Peter Henderson and Henry Walker, and developed new material, curricula, etc. for introductory discrete math courses for our majors.

2. Computing Curriculum Software Engineering – the first public draft of the CCSE volume was released on July 17, 2003 (see http://sites.computer.org/ccse/) with reviews by the public requested by September 19, 2003. The core mathematical foundations consists of 58 hours (see page 24 of the volume) with two recommended courses CS 105 (Discrete Structures I) and CS 106 ((Discrete Structures II) to be taken as early as possible in the curriculum (Quote from Page 53 “It is highly recommended that the discrete mathematics courses be taught starting in first year in lieu of any other mathematic course requirements since it is more important that a strong discrete mathematic foundation is made than, for example, calculus.”). The ‘missing piece’ is integrating this mathematics into the foundational computer science and software engineering courses. However, this will come as computer science software engineering education mature.

3. SIGCSE Committee on the Implementation of a Discrete Mathematics Course – this is the first such SIGCSE committee and was approved by the SIGCSE board in July 2003. I am thrilled that the topic of mathematics in our computer science curricula was selected to spearhead this effort. Doug Baldwin and Bill Marion are the co-facilitators for this committee. The charge of the committee can be found at http://www.sigcse.org/topics/committees.shtml. I encourage you to join the committees discussion listserver by sending the message ‘SUBSCRIBE SIGCSE-MATH-COMM your name’ to LISTSERV@ACM.ORG.

4. Mathematics in CS Education - The September 2003 issue of the Communications of the ACM, with guest editor Keith Devlin, will be devoted to the topic of the role of
mathematics in computer science education. There will even be an article by yours truly entitled “Mathematical Reasoning in Software Engineering Education.” I anticipate that this issue will further increase the awareness of the relevance of mathematics in the education of computer science and software engineering students, and will stimulate valuable discussions.

Again, as you can see there is significant activity relating to the topic of mathematics in the education of our students. However, there is still much to be done – specifically, integrating the mathematics into computer science and software engineering courses so students understand its importance and relevance, introducing discrete mathematics and logic early in the curricula and improving the quality of the instruction of the basic mathematics courses for computer science and software engineering students.

Continuing the topic of inductive reasoning, recall from the previous column that ‘inductive reasoning’ is “reasoning from detailed facts to general principles”. Inductive reasoning plays a very important role in understanding, specifying, designing, implementing and validating software. Below is the illustrative problem considered in the previous column.

**Problem:** Consider a sequence of $n > 0$ integers $i_1, i_2, i_3, \ldots, i_n$ that we wish to find the maximum value of.

Here are generalized patterns discovered using the inductive reasoning from the previous column - assume one has computed the maximum of the first $k$ integers in the sequence $(i_1, i_2, i_3, \ldots, i_k)$ and wishes to compute the maximum of the first $k+1$ integers in the sequence $(i_1, i_2, i_3, \ldots, i_k, i_{k+1})$. Let $\text{maxOfSeq} : \text{integer sequence} \rightarrow \text{int}$; now, for each $k \geq 1$

$$\text{maxOfSeq}( i_1, i_2, i_3, \ldots, i_k, i_{k+1} ) = \max ( \text{maxOfSeq}( i_1, i_2, i_3, \ldots, i_k ), i_{k+1} )$$

$$\text{maxValue} = \max ( \text{maxOfSeq}( i_1, i_2, i_3, \ldots, i_k ), i_{k+1} )$$

$$\text{maxValue} = \max ( \text{maxValue}', i_{k+1} )$$

where function $\text{max} : \text{int X int} \rightarrow \text{int}$ returns the maximum of its two argument values, and $\text{maxValue}'$ represents the previous value of the variable $\text{maxValue}$ in an iterative algorithm.

Once we have gained a good understanding of the problem, we can identify appropriate pre and post conditions. Some may argue that we should have done this first; but without ‘playing’ with the problem, it is hard to ensure we understand it completely as there are often unclear or conflicting aspects and/or hidden nuances,

Precondition: \{ a sequence of $n > 0$ integers $i_1, i_2, i_3, \ldots, i_n$ \}

Postcondition: \forall j \in \{1,2,3,\ldots,n\}, \text{maxValue} \geq i_j \text{ and } \exists j \in \{1,2,3,\ldots,n\} \text{ such that } \text{maxValue} = i_j$

---

1 ‘$\text{maxOfSeq}$’ is a function representing a solution to the problem we are trying to solve.
It is interesting to observe that many novice students identify only the first universal quantification, not realizing the second existential one is required. A simple example such as the sequence 1,2 with maxVal = 3 suffices to jog their thinking.

A potential iteration invariant for an algorithmic solution can be discovered from the previous columns generalized inductive reasoning and the above generalized patterns. Basically, in English, the maximum value of the first $k$ values in the sequence has been correctly computed. More formally, the proposed iteration invariant is a logical predicate

\[ I(\text{maxValue}, \{ i_1, i_2, i_3, \ldots, i_k \}) = \text{maxValue} = \text{maxOfSeq}(i_1, i_2, i_3, \ldots, i_k) \]

Below is a skeleton of an algorithmic solution illustrating key logical assertions based upon this iteration invariant.

\[
\begin{array}{l}
\{ \text{precondition} \} \\
\quad k \leftarrow 1 \\
\quad \text{maxValue} \leftarrow i_1 \\
\{ \text{maxValue} = \text{maxOfSeq}(i_1, i_2, i_3, \ldots, i_k) \} \\
\quad \text{while } <\text{condition}> \text{ do} \\
\quad \quad \{ \text{maxValue} = \text{maxOfSeq}(i_1, i_2, i_3, \ldots, i_k) \} \\
\quad \quad \quad k \leftarrow k + 1 \\
\quad \quad \quad \text{maxValue} \leftarrow \max(\text{maxValue}, i_k) \\
\quad \quad \{ \text{maxValue} = \text{maxOfSeq}(i_1, i_2, i_3, \ldots, i_k) \} \\
\{ \text{maxValue} = \text{maxOfSeq}(i_1, i_2, i_3, \ldots, i_k) \text{ and not } <\text{condition}> \Rightarrow \text{post condition} \} \\
\end{array}
\]

Upon termination, the invariant and negation of $<\text{condition}>$ must imply the truth of the post condition. Note that the invariant must be true at the four locations shown. Students can see the truth of the invariant after the initialization. Using forward logical reasoning, if the invariant is true before the body of the iteration, students can see (with your guidance) that it is true after the body. Backward reasoning can also be used. If $<\text{condition}>$ has no side effects\(^2\), then it is easy to argue, using forward and backward logical reasoning, that if the invariant is true at one location then it must be true at another ‘adjacent’ location (e.g., if the invariant is true before the while $<\text{condition}>$ do and the value of all variables remain the same, then the invariant must be true immediately after the while $<\text{condition}>$ do )\(^3\). Also, it is easy to see that the $<\text{condition}>$ must be designed to ensure that $k = n$ upon termination.

---

\(^2\) Creates no changes to the value of any variables.

\(^3\) In general, this is a mathematical induction based argument.
This example was designed to illustrate the power of inductive thinking when discovering and/or reasoning about algorithms, and the important connections with mathematical logic. I hope you find it useful.