DIMACS Series in Discrete Mathematics
and Theoretical Computer Science

Research at ASMSA Based on the DIMACS Biomath Program

Charles Mullins and Daniel W. Cranston

Charles Mullins is a mathematics teacher at the Arkansas School for Mathematics, Sciences, and the Arts; he has taught a variety of courses there during the past eight years. This article is written from his perspective and when the word "I" appears it refers to him.

Daniel Cranston is a postdoc at the Center for Discrete Mathematics and Theoretical Computer Science, at Rutgers, where he got involved with the writing of this article. His research is in graph theory and discrete mathematics.

1. Introduction

Many universities and colleges integrate research into their mathematics curriculums and this is becoming common in high schools, too. For five summers I (Charles Mullins) was part of a program at Rutgers University, sponsored by the Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), that prepared high school teachers to mentor their students in research. During my first four summers at DIMACS, I studied Graph Theory, but in 2004 the focus shifted to Biomath, with an emphasis on the Human Genome Project. I felt confident that, as in previous years, my students at the Arkansas School for Mathematics, Sciences, and the Arts (ASMSA) could adapt this topic for their senior research projects. This paper is a look at my research experience at the 2004 summer DIMACS Connect Institute on Biomath and my mentoring experience with a student researcher at my school.

2. Research at ASMSA

During the summer of 2004 at DIMACS, several other high school teachers and I worked on the Superstring Problem, under the guidance of Elizabeth Z Sweedyk. Since ASMSA requires each senior to complete a substantial research project, I saw the Biomath Institute as an ideal opportunity to do research that I could later revisit with my students.

Each junior at ASMSA chooses a project from a list compiled by all the teachers. The student first investigates topics to develop a research proposal, then meets with potential research mentors to evaluate the feasibility of the proposal. Once a teacher and student agree on a project, they begin meeting for 90 minutes each week, during a class period called FIRM (Fundamentals of Research Methods). I

usually have two or three students, but can meet with one while the others work on their projects. Students occasionally come by for extra help, but this rarely requires much additional time. These weekly meetings begin during the spring of the junior year, and continue until the student completes the project, in early March of the senior year. Successful completion of FIRM is a requirement for graduation.

Once the project begins, each student is expected to make regular progress, both during FIRM and outside of class. The student keeps research notes in a logbook, which is checked regularly in FIRM class. To ensure the student is making good progress, specific deadlines are set for rough drafts of each part of the paper; these parts include Introduction, Methods and Materials, Results, Analysis, and Conclusion. During February of the senior year, the student completes a paper and prepares a poster to present at the regional science fair. If successful at the regional fair, the student then competes in a state science fair. He or she also prepares an oral presentation for the Junior Academy, which is judged separately, by outside judges.

3. The Superstring Problem

The human genome consists of over 3 billion bases and the goal of the Human Genome Project [**5**] was to sequence these. Because current methods for reading the genome are unable to process the billions of digits encoded in the DNA molecule all at once, the most common methods require cutting the DNA molecule into short strands. Until recently, the technology could only process strands of about five hundred bases and even now the limit is less than a thousand. These small strands must then be reassembled into the original larger strand. In other words, the original strand must be a superstrand that contains all the smaller strands. It was this problem that my group at DIMACS studied.

Invoking Occam's Razor [**8**], we assumed that the original strand would be a shortest superstrand that contained all the short strands. Unfortunately, it is easy to find examples where there are many shortest superstrands, but our research did not address this complication. We were merely looking for one that worked. The mathematical approach is to treat the small strands as *strings*, consisting of letters and numbers, then to look for a shortest *superstring* that contains all the smaller ones. The challenge to find a shortest superstring that contains a given set of strings is called the Superstring Problem [**6**].

One attempt to solve the Superstring Problem is the GREEDY Algorithm [**7**], which is used in all sorts of optimization problems, such as the Traveling Salesman Problem. The first step in the GREEDY Algorithm is to consider all possible pairs of strings from the given set. Clearly, there are only two possible superstrings for a given pair. If these two superstrings are the same length, then either superstring is chosen. Otherwise, the shorter superstring is chosen. From this set of pairwise-formed superstrings, a shortest superstring is chosen. Again, the choice is arbitrary if there is more than one superstring of the same length. The pair corresponding to this shortest superstring is replaced by their superstring, to form a new set of strings with one less string. Note that this new set contains all the original strings except for the pair that has been replaced by their shortest superstring. This process is applied to the new set to reduce the number of strings again by one. These steps are repeated until a single superstring remains.

For example, suppose the set of strings is $S = \{abc, cab, bca, cba\}$. The shortest superstring for the strings *abc* and *cab* is chosen from among *abcab* and *cabc*; this results in *cabc*. For strings *abc* and *cba* the choice is between *abcba* and *cbabc*. In this case, since the superstrings are of equal length, either superstring may be chosen; we choose *abcba*. By considering each of the $\binom{4}{2}$ pairs of original strings, we reach a set of six pairwise-formed shortest superstrings $\{cabc, abca, abcba, bcab, cabcba, bcacba\}$. This set contains three shortest strings, so again we arbitrarily choose one, say *cabc*. We now form a new set S_1, by adding *cabc* to S and removing both strings in its corresponding pair: *abc* and *cab*. This yields the set $S_1 = \{cabc, bca, cba\}$. Note that when forming the set S_1, it is necessary to keep track of which pair corresponds to each pairwise shortest superstring. If we apply this process to S_1, one possible set of pairwise-formed superstrings is $\{cabca, cabcba, bcacba\}$; here *cabca* corresponds to the pair *cabc* and *bca*. Hence, in this case the new set is $S_2 = \{cabca, cba\}$. The final superstring produced by the GREEDY Algorithm would be either *cabcacba* or *cbacabca*. However, a shorter superstring is *cbabcab*.

As displayed in the example, the GREEDY Algorithm often fails to produce the shortest superstring. Thus, it is important to have a criterion to measure how "good" the GREEDY Algorithm is; this question led to the notion of p-approximation. A *p-approximation algorithm* always produces a superstring of length at most p times the length of the optimal superstring. Blum et al. [**2**] gave the first linear approximation bound; they showed that GREEDY is a 4-approximation algorithm and also gave a variant of GREEDY that yields a 3-approximation. Our DIMACS mentor, Z Sweedyk [**4**], designed an algorithm that gives a 2.5-approximation. Under her guidance, our research group studied a class of the Superstring Problem where each string consists only of 0s and 1s. We concluded that for this class of strings, GREEDY always produces the optimal superstring. We outlined a proof of this result, but two parts of our proof needed more details.

4. Implementing the Biomath research at ASMSA

Upon my return to ASMSA, one of our best students, Johnson Wong, read my description of the Superstring Problem and was quite keen on studying it for his FIRM project. Needless to say, I was delighted and encouraged him to do so. After studying the GREEDY Algorithm, Johnson submitted his proposal. I think it is best summarized by his abstract, which I have reproduced below, in a slightly modified form.

> *To simulate random fragmentation resulting from the reaction of enzymes with the human chromosome, a program was written to randomly generate five pseudo DNA strings. These strings, with lengths ranging from three to seven characters, consisted of random sequences of the letters A, C, G, and T, which represent the bases that make up DNA: adenine, cytosine, guanine, and thymine, respectively. A GREEDY algorithm was written to compare pairs of randomly-generated strings and determine how much they can be overlapped, or superimposed onto each other, so that the sequence of letters in both strings is still present. The algorithm then took the two strings that overlapped the most and merged them together, forming a superstring, which replaced its two substrings.*

> *This merging process continued until one string, the overall superstring, was left. The ratio between the length of the final superstring generated by the GREEDY algorithm and that of the shortest possible superstring, found manually, was calculated.*
>
> *This ratio, denoted by p, has been mathematically proven to be at most 2.5. Examples are known for which p equals 2 and empirical evidence suggests that p is never greater than 2. In other words, GREEDY will never produce a superstring that is more than twice as long as the optimal superstring. Hence, a major goal of studying these randomly generated sets of strings was to search for sets with p greater than 2. Sets with a p-value of 1.0 were also noted in order to find patterns that would help optimize the program to produce the shortest superstring more consistently. All changes made to the algorithm were logged. To compare the efficiency of different versions of the program, one version was selected as a benchmark. Using the benchmark version, the lowest, highest, and average values of p were recorded for each of the many sets examined. All programs were created using AutoIt v3, an open source scripting language based on C++ and QBASIC, which was designed for automation in the Windows graphical user interface.*

Johnson presented his final project to the BIOMATH follow-up conference at DIMACS, in April 2005. It was clearly well-received, justifying my view that he did an outstanding job. In addition to an overview of his project, he produced marvelous animations that illustrated his work. One animation beautifully illustrated an instance when p must be at least 2. I am proud to mention that he took second place in the mathematics section at the Arkansas State Science Fair.

I was particularly pleased that from the very beginning of the FIRM class, Johnson transformed the original problem into a project that interested him and used his strengths to good advantage. Although Johnson was one of our top students in mathematics (he made top grades in Differential Equations, Number Theory, and Vector Calculus), he also studied Computer Science. So in addition to his mathematical perspective, he brought his programming experience to bear. Since none of our team at DIMACS approached the problem with programming, it was not surprising that Johnson's approach put the problem into a context totally different from our work at DIMACS.

Of course, not everything worked as well as Johnson hoped. Generating random strings proved to be much more challenging than he expected, especially "long" strings. Early attempts took an unacceptable amount of computer time, and to the very end Johnson struggled with that problem. However, I think he felt that with more time he could have made significant gains. I believe a large portion of his time was spent tuning his program's performance. However, he still managed to examine a huge amount of data and do an outstanding job of analyzing his results. One surprising observation was that the size of p did not increase with the length of the string or with the size of the alphabet. Indeed, in his results, p tended to decrease as string length and alphabet size increased. However, his conjecture that p increases as the number of strings increases was supported by his results. For his randomly generated sets of strings, the maximum p obtained was 1.5.

I asked Johnson to comment on what background he felt a student needed to undertake his project. Below is his response:

> *When I started the project, I was taking Differential Equations and Vector Calculus later in the year. I already had pre-AP Biology and was taking AP Bio. I had written a paper on the Human Genome Project in 11th grade, so I was familiar with the terms and the idea of sequencing DNA. I felt that I sort of avoided the biology aspect of DNA other than explaining the idea behind shotgun sequencing.*
>
> *I think a good Computer Science (CS) student or a good math/decent CS student would be able to do a project like this. Mainly, it just requires someone with good logic, as there isn't any advanced math. As you know, the algorithm for calculating the solutions took a long time to finish. I enjoy programming, but I wouldn't say that I'm especially good at it. However, my logic is good enough to solve problems. Although I did manage to find some points where shortcuts could be taken in certain cases, I believe a very good CS student would have been able to make the program run much faster by writing it in a way that the computer likes better. The project is not very feasible unless the student has access to fast computers and/or has plenty of time to run the program.*

5. Reflections and Conclusions

It was an absolute pleasure to mentor Johnson and I'm sure I will enjoy working with other students on this problem in the future. It has been interesting to see students make a project their own as Johnson did. I actually expected him to take a completely different approach. Since he was such a strong math student, I thought he would be more attracted to the theoretical aspect of the problem than the applied. Although Johnson was an excellent biology student (he is currently majoring in Bioengineering at Johns Hopkins University), his approach was primarily one of applied math. Besides Johnson's research in Biomath, several of my students have done research in Graph Theory, with excellent results. While working with these students, I have often been surprised by the direction their research took. In some cases, as with Johnson, it was applied versus theoretical; in other cases, it involved which aspects of the problem interested them most.

Regardless of the topic, it is very important that the student have a strong understanding of the problem and the ideas on which it is based. This foundation need not be deep, but must be very clear. Many of the difficulties that students encounter might have been avoided if they had become more familiar with the problem earlier on; however, it is often a challenge to convince students of the importance of background research. Eventually they realize this, but often only after much unnecessary effort.

While Johnson was certainly a very strong student, I believe that most mathematically mature students could undertake a project on this topic. By *mathematically mature* I mean a student who will take AP Calculus or beyond; at ASMSA, that is most of the students. I believe the project can be adapted to many levels, depending on the student's background. For example, a student with a strong mathematical background could certainly follow the approach our DIMACS group

took and perhaps extend it. In fact, certain aspects of our research could be done in different and probably better ways by a good high school student.

An average student could investigate superstrings for various alphabets. It's always important for the student to do lots of examples before setting off on too much research. Initially, the student could investigate many different examples of combining strings and looking for a best superstring. Surely I would have to spend more time helping some students understand the problem's background, but most students should grasp it without a great deal of Biology. Because the Biomath option is on the cutting edge of genetics, I believe it may be more attractive to many students than other topics, and I envision the research going in a variety of directions.

I hope it is clear that I felt especially privileged to participate in the DIMACS Connect Institutes. It was extremely helpful to me in implementing the FIRM component of the ASMSA curriculum. I certainly think my DIMACS experience has been good for me, good for my students, and good for ASMSA.

References

[1] A. Biggs, W.C. Hagins, C. Kapicka, et. al., *Biology, The Dynamics of Life*, Glencoe, New York, 2004.

[2] A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis, *Linear Approximation of Shortest Superstrings*, Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, New Orleans, LA, May 6–8, 1991, pages 328–336. ACM Press, New York, 1991.

[3] D. Gusfield, *Algorithms on Strings, Trees and Sequences*, U of Cambridge, New York, 1997.

[4] Z. Sweedyk, *A* $2\frac{1}{2}$*-Approximation Algorithm for Shortest Superstring*, SIAM J. Comput. 29(3): 954-986, 1999.

[5] U.S. Department of Energy Office of Science, *Human Genome Project Information*, `http://www.ornl.gov/sci/techresources/Human_Genome/home.shtml`, retrieved 1 Sept. 2004.

[6] M. Weinard and G. Schnitger, *On the Greedy Superstring Conjecture*, University of Frankfurt. `http://www.thi.informatik.uni-frankfurt.de/~weinard/Presentations/fsttcsshow.pdf`, retrieved 22 Jan. 2005.

[7] E.W. Weisstein, *Greedy Algorithm*, Wolfram Research. `http://mathworld.wolfram.com/GreedyAlgorithm.html`, retrieved 1 Sept. 2004.

[8] Wikipedia, *Occam's Razor*. `http://en.wikipedia.org/wiki/Occam's_Razor`, retrieved 1 Sept. 2004.

Arkansas School for Mathematics, Sciences, & the Arts; Department of Mathematics, 200 Whittington Avenue, Hot Springs, AR 71901

E-mail address: `mullinsc@asmsa.org`

DIMACS Center/CoRE Building/4th Floor, Rutgers University, 96 Frelinghuysen Road, Piscataway, NJ 08854-8018

E-mail address: `dcransto@dimacs.rutgers.edu`