A Project in Algorithms based on a Primary Historical Source about Catalan Numbers

David Pengelley
Dept. Math. Sci.
New Mexico State University
Las Cruces, NM 88003
davidp@nmsu.edu

Inna Pivkina
Dept. Comp. Sci.
New Mexico State University
Las Cruces, NM 88003
ipivkina@cs.nmsu.edu

Desh Ranjan
Dept. Comp. Sci.
New Mexico State University
Las Cruces, NM 88003
dranjan@cs.nmsu.edu

Karen Villaverde
Dept. Comp. Sci.
New Mexico State University
Las Cruces, NM 88003
kvillave@cs.nmsu.edu

ABSTRACT

We discuss a project based on an original source from 1838 by Gabriel Lamé, which was used to teach dynamic programming in an Algorithms and Data Structures course for junior level computer science students. The project was developed as part of a group effort at New Mexico State University on using original historical sources in teaching. The project is based on an excerpt from a letter of Monsieur Lamé to Monsieur Liouville on the question: Given a convex polygon, in how many ways can one partition it into triangles by means of diagonals? A variety of tasks in the project, which includes reading, writing, proving statements by mathematical induction, deriving formulas, writing computer programs and analyzing and comparing them for efficiency, help students to develop verbal, analytical and discrete mathematics skills necessary for computer science. We also discuss student reactions to the project and to learning from historical sources.

Categories and Subject Descriptors

G.2.0 [**Discrete Mathematics**]: General; F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General; [**Pedagogy**]

General Terms

Algorithms

Keywords

Pedagogy, algorithms course, dynamic programming, historical sources

SIGCSE'06, March 1–5, 2006, Houston, Texas, USA.
Copyright 2006 ACM 1-59593-259-3/06/0003 ...$5.00.

1. INTRODUCTION AND PEDAGOGY

We are part of a collaboration of mathematics and computer science faculty teaching with student projects based on primary historical sources. Our approach has arisen by combining and extending two successful and well-developed programs in mathematics instruction at New Mexico State University. For many years we have been teaching calculus with substantial student projects, and we have also been developing student learning from primary historical sources in various mathematics courses. Our current team is focused on melding these learning approaches via student projects based on primary sources, specifically in the context of courses with a discrete mathematics component for majors in both computer science and mathematics. The idea is for students to gain the rich advantages that accrue from studying primary sources, an approach common in the humanities, but traditionally uncommon in mathematics and the sciences. The practice of teaching mathematics with primary historical sources is now growing at every level from primary to graduate school, and around the world.

There is increasing recognition that an historical point of view, including studying original sources, provides context, motivation and direction to mathematics teaching [2, 3, 5, 7]. We have a web resource site devoted to this [6], and one specifically devoted to discrete mathematics [4]. For teaching algorithms, while there is an excellent account of their history [1], our aim focuses on undergraduate curricular materials ready for classroom use.

Experience with students shows that reading the original work of some of the greatest minds in history provides improved motivation through understanding the originating problems and driving forces at the conceptual roots of ideas and methods of the subject. Primary sources also foster a broader perspective, reveal mathematical and social context, and hone students' verbal and deductive skills. Struggling with original texts has a flavor of adventure and thrill, displays the primordial creativity of geniuses of the past, and throws light upon the true nature of the subject and the practice of research, and upon the genesis and progress of the subject from then to now. This creative immersion into the difficulties of the past helps students better understand the problems of today. Finally, students gain a more profound technical comprehension, since the initial simplicity of a theory can be a strong help in understanding it.

In the context of computer science, a particular advantage is the rediscovery of the conceptual roots it shares with discrete mathematics. Typically, the perceived history of computer science is very short. This approach allows one to connect computer science to a longer history, to show that computer science did not appear from nowhere, and that its roots were developed throughout the centuries in other disciplines. Additionally, students practice the skill of moving from verbal descriptions of problems to precise mathematical formulations, and must often recognize an organizing concept for a detailed procedure. Such abilities are vital not only for mathematics and the sciences, but especially today for software engineers, who must translate a verbal request into precise code changes, and then realize what effect these changes will have on the global structure of a large program or a body of interacting programs. Applying modern programming techniques to historical challenges and seeing how problems which used to be hard can be solved easily today, stimulates student interest, motivation and general appreciation for science.

2. PROJECT DESCRIPTION

Dynamic programming is an important paradigm for algorithm development. It is usually taught to computer science students in an Algorithms course in the junior/senior year. Several typical examples of problems used in teaching dynamic programming lead to what are called *Catalan Numbers*, e.g. the optimal polygon triangulation problem and the optimal binary search tree problem. One needs to understand that the Catalan Numbers grow very rapidly to appreciate that brute-force algorithms are computationally inefficient to solve these problems. Often the formula for the Catalan numbers is presented to the students as a fact without proof. The project presented here allows the students to learn about the Catalan numbers and an elegant proof to derive the closed formula for the n^{th} Catalan number in a much more interesting fashion. The project makes use of a historical source from 1838 by French mathematician Lamé (translated to English by one of the authors of this paper), which presents a brilliant proof for the formula for the n^{th} Catalan number by comparing two different ways of counting polygon triangulations. The project calls for careful reading of the source and for the students to perform a number of interlaced mathematical and computational exercises based on the paper. The project allows the students to learn/practice a number of important concepts in computer science and discrete mathematics including recursive thinking, induction, counting and enumerating structures, efficient programming, comparison of run times of algorithms, importance of efficient algorithm design, etc.

In Section 2.1 we present the historical source. In Section 2.2 we present the project. In addition to the project discussed in this paper the historical source can be used as a basis for other projects as well. For example, there is a mathematical sciences project based on the same historical sourse ([4]).

2.1 The Historical Source

Excerpt from a letter of Monsieur Lamé to Monsieur Liouville on the question: Given a convex polygon, in how many ways can one partition it into triangles by means of diagonals?[1]

Journal de Mathématiques Pures et Appliquées (Journal de Liouville) **3** (1838), 505–507.
(math-doc.ujf-grenoble.fr/JMPA/)
(gallica.bnf.fr/Catalogue/noticesInd/FRBNF34348784.htm)

"The formula that you communicated to me yesterday is easily deduced from the comparison of two methods leading to the same goal.

"Indeed, with the help of two different methods, one can evaluate the number of decompositions of a polygon into triangles: by consideration of the sides, or of the vertices.

I.

"Let $ABCDEF\ldots$ be a convex polygon of $n+1$ sides, and denote by the symbol P_k the total number of decompositions of a polygon of k sides into triangles. An arbitrary side AB of $ABCDEF\ldots$ serves as the base of a triangle, in each of the P_{n+1} decompositions of the polygon, and the triangle will have its vertex at C, or D, or $F\ldots$; to the triangle CBA there will correspond P_n different decompositions; to DBA another group of decompositions, represented by the product P_3P_{n-1}; to EBA the group P_4P_{n-2}; to FBA, P_5P_{n-3}; and so forth, until the triangle ZAB, which will belong to a final group P_n. Now, all these groups are completely distinct: their sum therefore gives P_{n+1}. Thus one has

$$P_{n+1} = P_n + P_3P_{n-1} + P_4P_{n-2} + P_5P_{n-3} + \cdots + P_{n-3}P_5 + P_{n-2}P_4 + P_{n-1}P_3 + P_n. \quad (1)$$

II.

"Let $abcde\ldots$ be a polygon of n sides. To each of the $n-3$ diagonals, which end at one of the vertices a, there will correspond a group of decompositions, for which this diagonal will serve as the side of two adjacent triangles: to the first diagonal ac corresponds the group P_3P_{n-1}; to the second ad corresponds P_4P_{n-2}; to the third ae, P_5P_{n-3}, and so forth until the last ax, which will occur in the group P_3P_{n-1}. These groups are not totally different, because it is easy to see that some of the partial decompositions, belonging to one of them, is also found in the preceding ones. Moreover they do not include the partial decompositions of P_n in which none of the diagonals ending in a occurs.

"But if one does the same for each of the other vertices of the polygon, and combines all the sums of the groups of these vertices, by their total sum

$$n\left(P_3P_{n-1} + P_4P_{n-2} + \cdots + P_{n-2}P_4 + P_{n-1}P_3\right)$$

[1] See a Memoir of Segner (*Novi Commentarii Acad. Petrop.*, vol. VII, p. 203). The author found equation (1) of M. Lamé; but formula (3) presents a much simpler solution. Formula (3) is no doubt due to Euler. It is pointed out without proof on page 14 of the volume cited above. The equivalence of equations (1) and (3) is not easy to establish. M. Terquem proposed this problem to me, achieving it with the help of some properties of factorials. I then communicated it to various geometers: none of them solved it; M. Lamé has been very successful: I am unaware of whether others before him have obtained such an elegant solution. J. LIOUVILLE

one will be certain to include all the partial decompositions of P_n; each of these is itself repeated therein a certain number of times.

"Indeed, if one imagines an arbitrary such decomposition, it contains $n-2$ triangles, having altogether $3n-6$ sides; if one removes from this number the n sides of the polygon, and takes half of the remainder, which is $n-3$, one will have the number of diagonals appearing in the given decomposition. Now, it is clear that this partial decomposition is repeated, in the preceding total sum, as many times as these $n-3$ diagonals have ends, that is $2n-6$ times: since each end is a vertex of the polygon, and in evaluating the groups of this vertex, the diagonal furnished a group including the particular partial decomposition under consideration.

"Thus, since each of the partial decompositions of the total group P_n is repeated $2n-6$ times in

$$n\left(P_3P_{n-1}+P_4P_{n-2}+\cdots+P_{n-2}P_4+P_{n-1}P_3\right),$$

one obtains P_n upon dividing this sum by $2n-6$. Therefore one has

$$P_n=\frac{n\left(P_3P_{n-1}+P_4P_{n-2}+\cdots+P_{n-2}P_4+P_{n-1}P_3\right)}{2n-6}. \qquad (2)$$

III.

"The first formula (1) gives

$$P_3P_{n-1}+P_4P_{n-2}+\cdots+P_{n-2}P_4+P_{n-1}P_3=P_{n+1}-2P_n,$$

and the second (2) gives

$$P_3P_{n-1}+P_4P_{n-2}+\cdots+P_{n-2}P_4+P_{n-1}P_3=\frac{2n-6}{n}P_n;$$

so finally

$$P_{n+1}-2P_n=\frac{2n-6}{n}P_n,$$

or

$$P_{n+1}=\frac{4n-6}{n}P_n. \qquad (3)$$

This is what was to be proven."

Paris, 25 August, 1838

2.2 The Project: Counting Triangulations of Polygons

2.2.1 Understanding Triangulations

A *diagonal* in a (convex) polygon is a straight line that connects two non-adjacent points. Two diagonals are different if they have at least one different endpoint. A *triangulation* of a polygon is a division of the polygon into triangles by drawing *non-intersecting* diagonals. For example, the 6-sided polygon $ABCDEF$ below is triangulated into 4 triangles by using the diagonals AD, AE, BD.

Two triangulations are different if at least one of the diagonals in a triangulation is different from all diagonals in the other triangulation.

TASKS:

1.1 Draw a triangulation of $ABCDEF$ that is different from the triangulation in Figure 1. How many diagonals does your triangulation have? How many triangles does it divide $ABCDEF$ into?

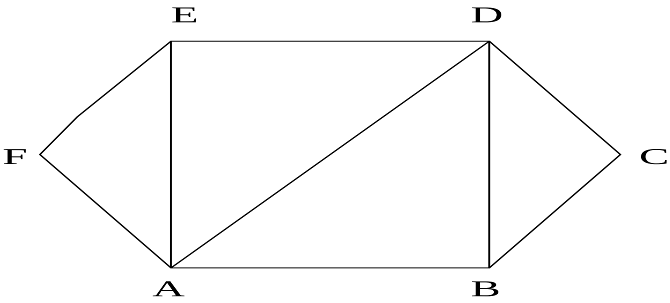

Figure 1: A Triangulation of $ABCDEF$

1.2 Consider an n-sided polygon $A_1A_2\ldots A_n$. How many different possible diagonals does the polygon have? **Note:** We are talking about all possible diagonals, not just diagonals in a triangulation.

1.3 In Section II of Lamé's paper there is a statement that any triangulation of an n sided polygon has $n-2$ triangles and $n-3$ diagonals. Prove that these statements are true using mathematical induction.

2.2.2 Optimal Triangulation and Counting Triangulations

The *Optimal Polygon Triangulation* Problem is the following: Given an n-sided polygon $A_1A_2\ldots A_n$ and a weight $w_{i,j}$ for each diagonal A_iA_j, find a triangulation of the polygon such that the sum of the weights of the diagonals in the triangulations is minimized. A naive way to solve the problem is to generate all possible triangulations one by one, calculate their weight (i.e. sum of weights of all the diagonals in the triangulation) and keep the best. The efficiency of this naive method depends on the number of possible triangulations of a polygon with n sides. Thus, we would like to count how many different triangulations an n-sided polygon has. The problem of counting the number of polygon triangulations of an n-sided polygon was first proposed by Leonhard Euler in the mid 18th century and studied by many famous mathematicians including Segner and Eugene Charles Catalan. In a short paper, in 1838, Gabriel Lamé provided an elegant method for this counting.

TASKS:

2.1 Read Section I in Lamé's paper. Explain what Lamé is saying in your own words and derive the general recursive formula for P_{n+1} (formula (1) in the paper).

2.2 Use the recursive formula to calculate P_i for $i=2,3,4$, 5, 6,7, 8 by hand and display it as a table.

2.3 Draw all triangulations of polygons with n sides for $n=4,5$.

2.4 Lamé's recurrence relation in his section 1 for $n=5$ yields

$P_6=P_5+P_3P_4+P_4P_3+P_5.$

Draw all triangulations of a 6-sided polygon classified into groups according to the idea of the recurrence relation, i.e., the triangulations should be classified into four groups with each group corresponding to a term on the right-hand side of the recurrence above.

2.5 Write a simple recursive function $SRCAT(n)$ (stands for "Simple Recurrence CATalan") in Java that given an input n calculates P_n using the recurrence relation (1) in Lamé's paper directly.

2.6 Write another Java program that repeatedly uses function $SRCAT$ to calculate P_i for $i = 3, 4, 5 \ldots$. Restrict the total time your program uses to 10 minutes. What is the largest value N_0 of i for which your program calculates P_i? Print out a table with i and the time required in seconds by $SRCAT$ to calculate each of the P_i values. Your table should have a row for each $i = 3, 4, 5 \ldots N_0$.

2.7 From your calculations you may observe that it seems that for all $n \geq 3$ $P_{n+1} \geq 2 * P_n$. Give a simple mathematical argument that establishes the truth of this statement.

2.8 Prove that for all $n \geq 3$, $P_n \geq 2^n/8$. **Hint:** Mathematical induction.

2.9 What does this tell you about the efficiency of the naive algorithm for solving the optimal polygon triangulation problem?

2.10 Write a Java program that repeatedly uses the recurrence given in formula (1) in Lamé's paper to calculate P_i for $i = 3, 4, 5 \ldots$ but that stores the computed values in an array systematically and uses them as needed. Restrict the total time your program uses to 10 minutes. What is the largest value M_0 of i for which your program calculates P_i?

2.11 Extend your program to print out a table of values of i and time required in seconds to compute P_i for $i = 3, 4, \ldots M_0$.

2.12 Graph the tables obtained in 2.6 and 2.11. Analyse these graphs and write down your observations.

2.2.3 *Lamé's Method for deriving a formula for* P_n

Section II of Lamé's paper gives an alternative way of counting triangulations of a polygon. Read this section carefully.

TASKS:

Consider a 6-sided polygon $ABCDEF$.

3.1 Draw all triangulations of the polygon where:

- AC is one of the diagonals in the triangulation.
- AD is one of the diagonals in the triangulation.
- AE is one of the diagonals in the triangulation.

How many total triangulations did you draw?

3.2 Repeat the same with vertex B as the "special" vertex, i.e., draw all triangulations where:

- BD is one of the diagonals in the triangulation.
- BE is one of the diagonals in the triangulation.
- BF is one of the diagonals in the triangulation.

How many total triangulations did you draw?

3.3 Do the same with vertices C, D, E, F being "special".

3.4 Consider the triangulation of $ABCDEF$ in figure 1 (of section 1). How many times is that triangulation repeated in all the triangulations that you drew for polygon $ABCDEF$ in this section? Identify the diagonals in whose group it was drawn.

3.5 Do the same for the different triangulations of polygon $ABCDEF$ that you drew in section 1.

3.6 What would you guess about the number of times any triangulation of $ABCDEF$ is repeated? Argue why your guess is correct.

3.7 Consider the n-sided polygon $A_1A_2 \ldots A_n$. Let P_i denote the number of different triangulations of a polygon with i sides.

(a) Calculate, in terms of P_i's, the number of triangulations of this polygon that have A_1A_3 as a diagonal, that have A_1A_4 as a diagonal, that have A_1A_j as a diagonal.

(b) Consider drawing triangulations treating A_1 as the "special" vertex. That is, draw all triangulations where A_1A_3 is a diagonal, then draw all triangulations where A_1A_4 is a diagonal, etc. all the way up to where A_1A_{n-1} is a diagonal. What is the number of triangulations you draw (in terms of P_i's) when A_1 is treated as a special vertex?

(c) Suppose we repeat the above process with another vertex (say A_2) being the special vertex instead of A_1. What can you say about the number of triangulations drawn as compared to the number of triangulations drawn when A_1 was chosen as the special vertex? Explain in your own words why this is true.

(d) Consider doing what you did for A_1 in (b) successively for each vertex. That is, enumerate all triangulations treating A_1 as a special vertex, treating A_2 as a special vertex, ...treating A_n as a special vertex. Now consider the specific triangulation of $A_1A_2 \ldots A_n$ obtained by drawing the diagonals $A_1A_3, A_1A_4, \ldots, A_1A_{n-1}$. How many times is this triangulation enumerated? What about the triangulation obtained by drawing the diagonals A_1A_4, A_1A_5, ..., A_1A_{n-2} and the two diagonals A_2A_4, $A_{n-2}A_n$? Justify your answer.

(e) What is your guess as to how many times any specific triangulation is enumerated? Explain in your own words why this is the case.

3.8 Combine (b) and (e) to derive the formula (2) in Lamé's paper. Explain in your own words how this formula is obtained.

3.9 Combine formulas (1) and (2) in Lamé's paper to obtain the formula (3) in Lamé's paper. Show all the steps in your calculation. Explain why this formula is better in terms of calculation of values of P_n.

3.10 Using formula (3) in Lamé's paper, show that $P_{n+2} = \frac{1}{n+1}\binom{2n}{n}$ where $\binom{2n}{n} = \frac{(2n)!}{n!n!}$. **Hint:** Mathematical Induction.

3.11 Write a simple recursive function $ASRCAT(n)$ (for "Another Simple Recurvise CATalan") in Java that given an input n calculates P_n using the recurrence relation (3) in Lamé's paper directly.

3.12 Write another Java program that repeatedly uses function $ASRCAT$ to calculate P_i for $i = 3, 4, 5 \ldots$. Restrict the total time your program uses to 10 minutes. What is the largest value L_0 of i for which your program calculates P_i?

3.13 Extend your program to print out a table of values of i and time required in seconds by $ASRCAT(n)$ to compute each of the P_i values for $i = 3, 4, \ldots L_0$. Your table should have a row for each $i = 3, 4, 5 \ldots L_0$.

3.14 Write a better Java program using the ideas from dynamic programming ("store and re-use") that repeatedly calculates P_i for $i = 3, 4, 5 \ldots$. Restrict the total time your program uses to 10 minutes. What is the largest value L_1 of i for which your program calculates P_i.

3.15 Extend your program to print out a table of values of i and the time required in seconds to calculate each of the P_i values. Your table should have a row for each $i = 3, 4, 5 \ldots L_1$.

3.16 Graph the tables obtained in 3.13 and 3.15. Analyse all four graphs obtained and write down your observations. How do the results for the second two programs compare with your first two programs? How fast does the running time of the last two programs grow?

3.17 Discuss how the choice of Lamé's formulas (1) or (3), or using dynamic versus naive recursive programming influences the effectiveness of computation.

3. CLASSROOM EXPERIENCE AND CONCLUSIONS

The project has been used twice in a junior level computer science course "Algorithms and Data Structures" at New Mexico State University, a 4 credit course with lab). In Fall 2004 the project was worked on by students during the lab while dynamic programming was being covered during lectures. The project, as shown above, was 15% of the course grade and had four parts with about 8 problems each. The problems involved proof by induction, programming, enumerating structures, drawing conclusions, quantitative analysis and comparison of algorithm speeds, etc. Students worked individually on their project. Each part of the project lasted one week. During the lab, the students asked questions, developed proofs, made observations, etc. Students felt after completing their project that they really appreciated the benefits of dynamic programming. Some of the students thought the project was too hard (especially the proofs) and that it should have counted for more than 15% of their grade.

To address students' concerns a modified version of the project was used in Spring 2005. The modified version had significantly fewer questions requiring proofs, and fewer programming tasks. The revised project was in two parts, with 10 questions and 8 questions. Students had a week to do each part. No other assignments were given to students during these two weeks. Students did the project in teams of 1–2. The project was assigned when dynamic programming was introduced in lectures. It counted for 15% of the course grade. Some of the students again perceived the project as hard. However, everybody performed much better on the project than on other assignments in the course. This can be explained by the increase in the students' motivation and interest due to the use of the historical source. Pre-course and post-course questionnaires sought to evaluate the usefulness of the project. Out of nineteen students, seventeen said that there were benefits in learning mathematics from historical sources. Some of the benefits which students listed are: "context provides a hook to hang the information on", "helps to understand where other mathematics concepts come from", "learn by example", "I think it makes it a little more interesting", "understand how to reinvent certain ideas when necessary", "to know how easy it is now". After the course students' confidence with their mathematical skills increased (from 26% being very confident to 32%). The course also helped students understand the importance of mathematics for computer science. In pre-course questionnaires 47% of the students thought that mathematics was very important, 47% somewhat important, and 6% not important. In the post-course questionnaires, 58% said it was very important, 42% somewhat important, and none said that it was not important. One of the students wrote about the use of historical projects: "it really expands one's horizons and you come away more knowledgeable and more informed".

We conclude that this project based on a historical source can be an effective and interesting way of teaching dynamic programming, and increases students' appreciation of the connection between mathematics and computer science.

4. ACKNOWLEDGEMENTS

Supported by grant DUE-0231113 from the Division of Undergraduate Education of the National Science Foundation.

5. REFERENCES

[1] J.-L. Chabert (ed.). *A history of algorithms : from the pebble to the microchip.* Springer, New York, 1999.

[2] J. Fauvel and J. Van Maanen. *History in Mathematics Education.* Kluwer, Boston, 2000.

[3] V. Katz, (ed.). *Using History to Teach Mathematics.* Mathematical Association of America, Washington D.C., 2000.

[4] J. Lodder, et al. Teaching Discrete Mathematics via Primary Historical Sources. `http://www.math.nmsu.edu/hist_projects/`, 2003–.

[5] R. Laubenbacher and D. Pengelley. *Mathematical Expeditions: Chronicles by the Explorers.* Springer-Verlag, New York, 2000.

[6] R. Laubenbacher and D. Pengelley. Teaching with Original Historical Sources in Mathematics. `http://www.math.nmsu.edu/~history`, 1999–.

[7] F. Rickey. The Necessity of History in Teaching Mathematics. In R. Calinger (ed.), *Vita Mathematica: Historical Research and Integration with Teaching,* pages 251–256. Mathematical Association of America, Washington D.C., 1996.