The Binomial Theorem

Callie Wurtz

1 Motivation

Most people associate the binomial theorem with its ability to shorten the painstaking process of expanding binomials. While this is one of its uses, the importance of this theorem extends beyond algebra. The binomial theorem for the case when $n = 2$ was known as early as 300 BC by Euclid. It was further developed by Pascal and Fermat and eventually generalized by Newton in 1676 [Wei05]. Now a cornerstone for combinatorics, it is used to prove and derive numerous other identities. One of the appeals of the binomial theorem is the variety of ways it can be proven, arguably the most elegant is its combinatorial proof.

2 Preliminary Ideas

You should be familiar with the definitions of *independent* and *mutually exclusive*, the notation for *combinations*, and both the *Rule of Sum* and *Rule of Product* counting principles. If these terms are unfamiliar to you, please refer to the "Common Definitions" file. That file also contains a brief review of summation notation.

3 The Problem Presented

Theorem 1 *The Binomial Theorem*

Let n be a nonnegative integer. Then

$$(x+y)^n = \sum_{r=0}^{n} \binom{n}{r} x^r y^{n-r}.$$

4 The Solution by Counting

Imagine a class of n computer science students. Each student is given the choice of coding either 1 of x different projects in the computer language C++ or 1 of y different projects in the Java language. Since each person must choose one or the other, the tasks are mutually exclusive. Therefore, every student is choosing 1 out of $x + y$ possible projects. Also, what each student chooses is independent of the other students' choices. Thus, there are $(x+y)^n$ possible outcomes for the projects that the students choose.

Now, consider the number of students choosing to code in C++. Let this number be r. There are $\binom{n}{r}$ ways to determine which r students use C++. (The remaining $n - r$ students will write in Java.) Because there are x C++ projects, there are x^r ways for the r students to decide which projects to do. Likewise, there are y^{n-r} ways for the remaining students to decide which Java projects they do. Since each of these options is independent and different values of r are mutually exclusive, there are $\sum_{r=0}^{n} \binom{n}{r} \cdot x^r \cdot y^{n-r}$ total ways for the computer science projects to be chosen by the students.

Equating our two answers completes the proof.

□

5 Visual Example

Suppose we have a class of only three computer science students. (Note that this is a particular example. The visualization does not prove the general identity. It only relates to the case when $n = 3$.) These students: Jake, Karl, and Ryan must choose to work on either one $(x = 1)$ computer project in C++ or one of two $(y = 2)$ computer projects in Java. These projects will be represented as C, J_1, and J_2, respectively.

The visualization uses *decision trees* to depict the identity. Decision trees branch at nodes (vertices) where a choice is being made. The branches (edges) of a decision tree represent the available choices. For this example, the nodes represent the students making the decisions and the branches represent the possible computer projects.

The first part of the visualization corresponds to the left-hand side of the theorem. This counting method is represented by one large tree that contains all of the students' possible choices. Note that each individual has three choices. Remember that the choices are independent, so no matter what Jake chooses, both Karl and Ryan still have the same three choices. Thus, there are $3 \cdot 3 \cdot 3 = 27$ possibilities. The visualization then highlights each path in the tree and lists the projects that correspond with that pathway. The list of decisions from this tree is a list of all possible project combinations.

The second part of the visualization uses trees to display the right-hand side of the identity. Here, we are concerned with how many of each project is chosen. The visualization looks separately at each the terms of $\sum_{r=0}^{3} \binom{3}{r} 1^r 2^{3-r}$. When $r = 0$ or $r = 3$, only one tree is needed because only one project type is chosen. When $r = 1$ or $r = 2$, three trees are needed. As an example, consider the case when $r = 1$, which produces the summand $\binom{3}{1} 1^1 2^2$. The single C++ project could be chosen by any of the three students. If the C++ project is chosen by Jake, a different tree results than if the C++ project is chosen by Ryan. Again, the visualization highlights every pathway. The projects that correspond to that route are bolded in the comprehensive project list.

Notice that eventually every project combination is highlighted.

References

[BQ03] Arthur T. Benjamin and Jennifer J. Quinn. *Proofs that Really Count: The Art of Combinatorial Proof*. Number 27 in The Dolciani Mathematical Expositions. The Mathematical Association of America, 2003.

[Wei05] Eric Weisstein. *Mathworld*. Wolfram Research, http://mathworld.worlfram.com/, 2005.