Chapter 10

2-way Bounding

In high school and early college mathematics, we are often proving equalities and we often prove them by manipulating a sequence of equalities. For example

$$\frac{3x^2 - 5x - 2}{x - 2} = \frac{(x - 2)(3x + 1)}{x - 2} = 3x + 1$$

On more complex problems, it's often necessary to adopt a more flexible approach: bound the quantity of interest from both directions. When our best upper and lower bounds are equal, we've established an equality. Otherwise, we've limited the quantity to a known (hopefully small) range.

Because we need to establish a lower bound and also an upper bound, a 2-way bounding proof contains two sub-proofs. Sometimes the two sub-proofs are similar. However, the true power of the technique comes from the fact that the two sub-proofs can use entirely different techniques. Therefore, the method is widely used when proving more difficult results, e.g. in upper-level computer science and mathematics courses (e.g. real analysis, algorithms).

The most common error building these proofs is to omit one half entirely. For example, someone intending to show that $f(x) = k$ might prove that $f(x) \leq k$ but forget to also prove that $f(x) \geq k$. Do not do this. If one of the bounds is obvious, say that explicitly. In this chapter, we will see how this method works on a variety of examples.

10.1 Marker Making

Suppose that we have a 12" by 15" sheet of cookie dough, from which we'd like to cut maple-leaf cookies. Scraps of dough that have been reclaimed and re-rolled never produce results as nice as those from the original sheet, so we'd like to pack as many cookies into our original sheet as possible. How many cookies can we cut from this sheet?

We could experiment with cutting several sheets of dough, placing our cookie cutters in various different ways. Suppose we managed to fit in 25 cookies on our best attempt. We now know that 25 is a lower bound on the maximum number of cookies we can cut out. But it might be very hard to prove we've found the best layout.

Now, suppose that we put our maple-leaf cutters onto graph paper and work out that the area of each cookie is at least 6 square inches. Since our sheet of dough has area 180 square inches, we know that it's impossible to cut out more than 30 cookies. So we now have an upper bound of 30 on the number of cookies in the optimal layout. In this example, we would normally expect our lower bound and our upper bound to be some distance apart, because it's so difficult to search through all the ways to arrange our cutters.

Home kitchens do not need to worry heavily about efficiency, but industrial plants do. In particular, clothing manufacturers need to create **markers** for cutting clothing parts out of rectangles of cloth. Because the appearance and material properties of cloth are tied to its warp direction, parts must be lined up in specific orientations. Scrap cloth, though recyclable for other purposes, cannot simply be reformed and re-used. Efficient markers are important, because any waste will be replicated many times, but difficult for humans to create. So considerable effort has gone into the development of efficient marker making algorithms.

In both engineering and in theoretical computer science, it is common to encounter quantities that cannot be calculated directly but, rather, must be bounded from above and below. Sometimes we can make the upper and lower bounds meet, giving us a so-called **tight** bound. Often, we can't.

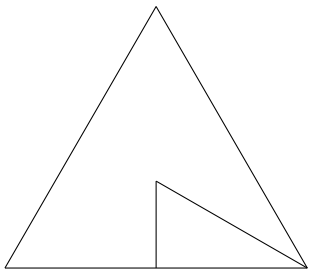
10.2 Pigeonhole point placement

Here's a problem of a very different sort, which uses 2-way bounding toegether with the pigeonhole principle. Like so many pigeonhole proofs, the proof depends on a non-obvious trick. In this case, it's a trick about dividing up a triangle. Once you've seen the trick, however, the proof is a classic example of bounding a quantity from above and below.

Claim 36 *Suppose that T is an equilateral triangle with sides of length 2 units We can place a maximum of four points in the triangle such that every pair of points are more than 1 unit apart.*

To show that 4 is the maximum number of points we can place with the required separations, we need to show that it's possible to place 4 points, but it is not possible to place 5 points. It's easiest to tackle these two subproblems separately, using different techniques.

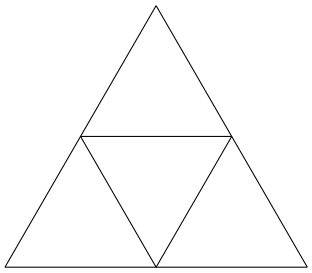
> Proof: To show that the maximum is at least four, notice that we can place three points at the corners of the triangle and one point in the center. The points at the corners are two units apart. To see that the point in the center is more than one unit from any corner, notice that the center, the corner, and the midpoint of the side form a right triangle.



1

> The hypotenuse of this triangle connects the center point to the corner point. Since one leg of the triangle has length 1, the hypotenuse must have length greater than 1.

To show that the maximum number of points cannot be greater than four, divide up the triangle into four smaller equilateral triangles as follows:

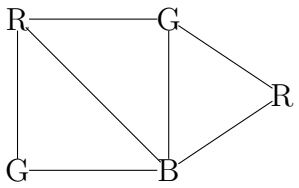


Suppose we tried to place five or more points into the big triangle. Since there are only four small triangles, by the pigeonhole principle, some small triangle would have to contain at least two points. But since the small triangle has side length only 1, these points can't be separated by more than one unit.

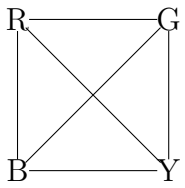
10.3 Graph coloring

A *coloring* of a graph G assigns a color to each node of G, with the restriction that two adjacent nodes never have the same color. If G can be colored with k colors, we say that G is k-colorable. The *chromatic number* of G, written $\chi(G)$, is the smallest number of colors needed to color G.

For example, only three colors are required for this graph:



But the complete graph K_n requires n colors, because each node is adjacent to all the other nodes. E.g. K_4 can be colored as follows:



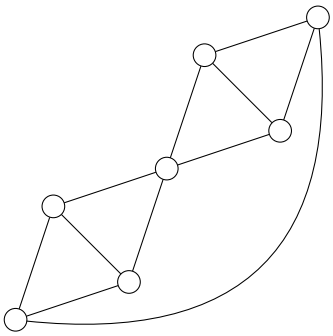
To establish that n is the chromatic number for a graph G, we need to establish two facts:

- $\chi(G) \leq n$: G can be colored with n colors.
- $\chi(G) \geq n$: G cannot be colored with less than n colors

For small finite graphs, the simplest way to show that $\chi(G) \leq n$ is to show a coloring of G that uses n colors. For a larger class of graphs, we could describe an algorithm for doing the coloring. For example, we can color a cyclic graph with an even number of nodes by alternating colors around the circle. A bipartite graph never requires more than two colors.

Showing that $\chi(G) \geq n$ can sometimes be equally straightforward. For example, if the graph has any edges at all, its chromatic number must be at least 2. If G contains a copy of K_n, the chromatic number of G must be at least n because K_n can't be colored with less than n colors.

However, the following example shows why this process can get tricky. It's relatively easy to color with 4 colors. But the largest complete graph in it is K_3, which gives us a lower bound of only 3. Showing that the chromatic number is actually 4 requires carefully stepping through all possible ways to assign three colors to the nodes and explaining why none can end with a complete coloring.



10.4 Why care about graph coloring?

Graph coloring is required for solving a wide range of practical problems. For example, there is a coloring algorithm embedded in most compilers. Because the general problem can't be solved efficiently, the implemented algorithms use limitations or approximations of various sorts so that they can run in a reasonable amount of time.

For example, catalog and on-line retailers frequently organize their clothes offerings into collections, e.g. "Golfing Geezer" or "Tough but Toucheable" or "Uniforms 4 U." All items in a collection are guaranteed to be compatible. This helps retailers market clothes to folks with no dress sense, as well as cope with the fact that people are sensitive to fine color distinctions that aren't reproduced accurately in catalogs and on computer screens. How many collections are required to organize the retailer's inventory of clothes?

We can model this problem as graph coloring. Each graph node is an item of clothing. Edges join pairs of items that aren't compatible, e.g. the bright green pants don't go with the grey-orange shirt, the business formal blouse doesn't go with the golfing pants. The "color" on each node is the name of a collection. If we find that too many collections are required, we might want to remove selected items of clothing (e.g. the bottle green bell bottoms that match nothing) from our inventory.

We can model a sudoku puzzle by setting up one node for each square.

The colors are the 9 numbers, and some are pre-assigned to certain nodes. Two nodes are connected if their squares are in the same block or row or column. The puzzle is solvable if we can 9-color this graph, respecting the pre-assigned colors.

We can model exam scheduling as a coloring problem. The exams for two courses should not be put at the same time if there is a student who is in both courses. So we can model this as a graph, in which each course is a node and courses are connected by edges if they share students. The question is then whether we can color the graph with k colors, where k is the number of exam times in our schedule.

In the exam scheduling problem, we actually expect the answer to be "no," because eliminating conflicts would require an excessive number of exam times. So the real practical problem is: how few students do we have to take out of the picture (i.e. give special conflict exams to) in order to be able to solve the coloring problem with a reasonable value for k. We also have the option of splitting a course (i.e. offering a scheduled conflict exam) to simplify the graph.

A particularly important use of coloring in computer science is register allocation. A large java or C program contains many named variables. But a computer has a smallish number (e.g. 32) of fast registers which can feed basic operations such as addition. So variables must be allocated to specific registers.

The nodes in this coloring problem are variables. The colors are registers. Two variables are connected by an edge if they are in use at the same time and, therefore, cannot share a register. As with the exam scheduling problem, we actually expect the raw coloring problem to fail. The compiler then uses so-called "spill" operations to break up the dependencies and create a graph we can color with our limited number of registers. The goal is to use as few spills as possible.

10.5 Proving set equality

Finally, 2-way bounding proofs are often used to prove that two sets A and B are equal. That is, we show that $A \subseteq B$ and $B \subseteq A$, using separate

subproofs. We can then conclude that $A = B$. This looks superficially different from the previous examples, because the relation is $\subseteq$ rather than $\leq$. But the main idea is the same: we first show that A is no larger than B, and then show that A is no smaller than B.

As an example, let's look at

Claim 37 *Let* $A = \{15p + 9q \mid p, q \in \mathbb{Z}\}$ *Then* $A = \{$*multiples of 3*$\}$.

Before you try to prove this, first put some sample integer values into the formula $15p + 9q$. See if you can convince yourself informally that this formula really generates all and only multiples of 3.

Using the bounding method, we would write the proof as follows:

> Proof:
>
> (1) Show that $A \subseteq \{$multiples of 3$\}$.
>
> Let x be an element of A. By the definition of A, $x = 15s + 9t$, for some integers s and t. But then $x = 3(5s + 3t)$. $4s + 3t$ is an integer, since s and t are integers. So x is a multiple of 3.
>
> (2) Show that $\{$multiples of 3$\} \subseteq A$.
>
> Notice that (*) $15 \cdot (-1) + 9 \cdot 2 = 3$.
>
> Let x be a multiple of 3. Then $x = 3n$ for some integer n. Substituting (*) into this equation, we get $x = (15 \cdot (-1) + 9 \cdot 2)n$. So $x = 15 \cdot (-n) + 9 \cdot (2n)$. So x is an element of A.
>
> Since we've shown that $A \subseteq \{$multiples of 3$\}$ and $\{$multiples of 3$\} \subseteq A$, we can conclude that $A = \{$multiples of 3$\}$.

10.6 Variation in terminology

When coloring graphs, we have placed colors on vertices. A closely-related set of problems involve placing colors on edges. The terms "vertex coloring" and "edge coloring" are used when it's necessary to distinguish the two.