Math Club Notes: 2006 November 20

1 The rearrangement inequality

In our Putnam prep last Thursday, the speaker mentioned the *rearrangement inequality*:

> Let $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$ be n-element sequences of real numbers. Suppose a is increasing. Then, as σ varies over all permutations of $(1, \ldots, n)$, the maximum value of $\langle a, \sigma(b)\rangle$ is attained when $\sigma(b)$ is increasing, and the minimum value when $\sigma(b)$ is decreasing.

Here's a proof.

The case $n = 1$ is trivial. (The case $n = 0$ is even more trivial.)

Consider the case $n = 2$. Let $a_1 \le a_2$ and $b_1 \le b_2$. We wish to show that

$$a_1 b_1 + a_2 b_2 \ge a_1 b_2 + a_2 b_1 \; .$$

Indeed, a little rearrangement yields the equivalent inequality

$$(a_2 - a_1)(b_2 - b_1) \ge 0 \, ,$$

which certainly follows from $a_1 \le a_2$ and $b_1 \le b_2$. That's the case $n = 2$.

Now for the general case. We will proceed by describing an algorithm[1] for sorting a sequence — that is, permuting it so that it becomes increasing — and showing that each step of the algorithm increases the value of the inner product $\langle a, b\rangle$.

Here's the sorting algorithm: If b is sorted, stop. Otherwise choose some k so that $b_k > b_{k+1}$, exchange b_k and b_{k+1}, and repeat.

First we should prove that the algorithm makes sense. The only question here is whether such k always exists. Indeed, if not, that is, if $b_k \le b_{k+1}$ for all k, then b is sorted.

Next, we should prove that the algorithm terminates, that is, that b will become sorted after finitely many steps of this algorithm. For this purpose, consider the pairs (i, j) such that $i < j$ and $b_i > b_j$; call such pairs "unsorts". b is sorted if and only if there are zero unsorts. When the algorithm exchanges b_k and b_{k+1}, these two values' positions relative to other elements in the sequence are unchanged (because b_k and b_{k+1} are adjacent), and the pair $(k, k+1)$ stops being an unsort. Thus this step reduces the number of unsorts by one. Eventually, then, this number will reach zero; at that point b is

[1] Dijkstra uses this proof-by-algorithm approach to prove the AM/GM inequality; see our notes for 2005 May 9.

sorted and the algorithm terminates.[2]

Now, to prove the rearrangement inequality itself, we wish to show that every step of this algorithm increases the dot product $\langle a, b\rangle$. Let τ be the permutation which exchanges b_k and b_{k+1}. If $b_k > b_{k+1}$, then

$$\begin{aligned}\langle a, b\rangle &= a_1 b_1 + \cdots + a_k b_k + a_{k+1} b_{k+1} + \cdots + a_n b_n \\ &\le a_1 b_1 + \cdots + a_k b_{k+1} + a_{k+1} b_k + \cdots + a_n b_n \quad \text{(by the case } n = 2\text{)} \\ &= \langle a, \tau(b)\rangle \,.\end{aligned}$$

And that's that.

Here's another approach that might yield something: start with Lagrange's identity

$$\Big(\sum_{i=1}^{n} a_i b_i\Big)^2 = \Big(\sum_{i=1}^{n} a_i^2\Big)\Big(\sum_{i=1}^{n} b_i^2\Big) - \sum_{i=1}^{n}\sum_{j=1}^{n}(a_i b_j - b_i a_j)^2 \,.$$

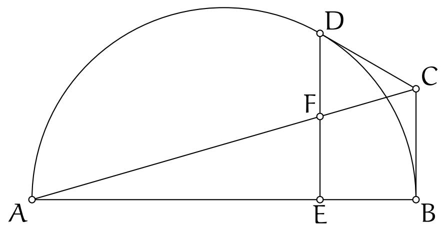
This identity is proven, and used to deduce the Cauchy-Schwarz inequality, in our notes for 2005 May 22. It has the nice property of saying exactly what the "defect" is in the Cauchy-Schwarz inequality — the sum $\sum_{i=1}^{n}\sum_{j=1}^{n}(a_i b_j - b_i a_j)^2$ is exactly the amount by which $\langle a, b\rangle^2$ falls short of $\|a\|^2\|b\|^2$. To show the rearrangement inequality it would suffice to show that the defect is smallest when $\sigma(b)$ is increasing. (That might not be any easier than the original version.)

[2]This proof of termination is typical. Note that the only reason we chose to exchange adjacent pairs in our algorithm (instead of arbitrary pairs in the wrong order) was to make this proof simpler.

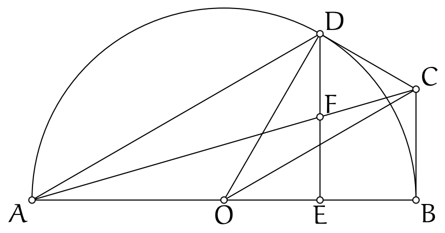
2 Archimedes' weird bisection lemma

We began looking at the following proposition from Archimedes' *Book of Lemmas*:

> Let AB be the diameter of a semicircle, and let the tangents to it at B and at any other point D on it meet in C. If now DE be drawn perpendicular to AB, and if AC, DE meet in F, then $DF = FE$.



Here's a hint. Let O be the centre of the circle, and add the lines shown here:



With these auxiliary lines, we can describe the relevant segments trigonometrically. For example, $DE = \sin\angle DOB$ (if we, wlog, take the circle to have radius 1), and $EF = AE\cos\angle CAB$. Continuing in this manner — and figuring out the relationships between the angles — we can reduce the problem to proving a certain trigonometric identity.

3 Making not

We also looked a little at our outstanding problem of expressing $\vee$ using only $\neg$ and $\veebar$ (that is, expressing inclusive-or using only negation and exclusive-or). Or rather, proving that it is not possible to do so.

When proving that a certain operation *can* be expressed using others, we always have the option of just writing down the relevant expression. For example, to show that $\vee$ can be expressed using only $\neg$ and $\wedge$ (and), we just write down a variant of De Morgan's laws:

$$A \vee B \equiv \neg(\neg A \wedge \neg B)\ .$$

To show that $\vee$ can be expressed using only $\wedge$ and $\veebar$, we write down

$$A \vee B \equiv A \veebar B \veebar (A \wedge B)\ .$$

The verification of these identities is routine.

But how can we prove that a certain operation *cannot* be expressed using others? Here's an example, showing that $\neg$ cannot be expressed using only $\wedge$, $\vee$, and $\veebar$.

We are interested in functions taking (some number of) boolean arguments, and returning a boolean value. For concreteness, we suppose the functions take two arguments, A and B. We then define the *expressible* functions recursively as follows:

1. The function $(A, B) \mapsto A$ is expressible.
2. The function $(A, B) \mapsto B$ is expressible.
3. If f and g are expressible, then the function $f \wedge g$ is expressible.
4. If f and g are expressible, then the function $f \vee g$ is expressible.
5. If f and g are expressible, then the function $f \veebar g$ is expressible.

These rules capture the notion of using $\wedge$, $\vee$, and $\veebar$ to build up larger expressions. For example, the function $(A, B) \mapsto A \veebar (A \vee B)$ is expressible, since A and $A \vee B$ are expressible.

Now, let us say that f *fixes zero* if $f(0, 0) = 0$. Obviously the projection functions $(A, B) \mapsto A$ and $(A, B) \mapsto B$ fix zero. Moreover, if f and g both fix zero, then $(f \wedge g)(0, 0) = f(0, 0) \wedge g(0, 0) = 0 \wedge 0 = 0$, so $f \wedge g$ fixes zero too. Similarly, $f \vee g$ and $f \veebar g$ fix zero. Therefore (by what they call "structural induction") any expressible function fixes zero.

The function $(A, B) \mapsto \neg A$ does not fix zero; therefore it is not expressible.

A similar style of proof can be given to show that $\vee$ cannot be expressed using only $\neg$ and $\veebar$. The trick, of course, is to find the property that all expressible functions have but $\vee$ doesn't.