Algorithmic Games ***Berkeley Math Circle***
November 21, 2004

We take a look at a few games that can all be described as "algorithmic", that is, they have a few common characteristics. First, a "situation" is involved (which we will see as as "starting state"). Then, there is a set of operations one can perform, which changes the configuration of the problem (that is, modifies the situation). Finally, there must be a "target" or "final" state, which will be desirable (in some context or another). These three we regard as "defining characteristics" of an algorithmic game.

The types of problems we will deal with involve the following:

Type 1: showing that there is a way to reach the target state, by performing a sequence of operations,

Type 2: showing that *every* sequence of operations we perform will eventually lead to the target state (if we don't stop on the way),

Type 3: showing that *every* sequence of operations we perform takes us to the target state *equally fast*,

Type 4: showing that the target state is unreachable.

Note that *Type 3* is a special case of *Type 2*, which in its turn is a special case of *Type 1*, while *Type 4* is complementary to *Type 1*.

In each one of these problems, we will define some parameter P (as a function of the current situation/configuration), and we will look at the way in which performing one operation or a sequence of operations modifies P. Typically, we will want to look at the "improvement" (or lack thereof) in P which is brought about by an operation. For each of the 4 types of problems we describe above, we will define P and the show

Type 1: that there is a way (or a sequence of operations) which "improves" P until it reaches the desired value (which will be somehow *unique* to the target state/configuration),

Type 2: that every operation improves P, until the value of P which is specific to the target state is achieved,

Type 3: that every operation improves P *by the same amount*, until the value of P which is specific to the target state is achieved,

Type 4: that *no* operation improves P, and that reaching the target state cannot be done without improvement.

Note that some of the problems may have more than one solution – we will only present the one that fits the pattern of the algorithmic game we describe here. Many times, different solutions will be considerably more laborious than the ones we present (think of Problem 3). =)

I'd like to thank all those who have contributed in today's class, many of which I remember by name: Aviv, Victoria, Erin, David, Curtis, Kenneth, Spencer – and I apologize to those whose names I forgot. Thanks.

Problem 1. Given an $m \times n$ array of integer values (which we can think of as indexed by pairs (i, j) with $1 \leq i \leq m$ and $1 \leq j \leq n$) , we define the following operation: pick a row or column with negative sum and flip the signs of all the values in that particular row/column. Is there a sequence of operations which will make all row/column sums non-negative?

Solution 1. Yes. Moreover, we will show that this problem is of Type 2, i.e., any sequence of operations will have to eventually produce a table of values with all row/column sums non-negative.

The parameter P here is the sum of all entries in the table. We can write

$$P = \sum_{i=1}^{m} \sum_{j=1}^{n} eij \ ,$$

where e_{ij} is the entry that sits at the intersection of row i with column j.

The key fact here is that we can sum the elements *either on rows or on columns.*

W.l.o.g., assume we choose a row with negative sum $-S$, and call that row i. If we flip the signs in row i, the sum on row i becomes now $2S$, and the sums of all the other rows are left unchanged. Hence P becomes $P + 2S$! Thus, we increase the parameter.

We keep on performing operations for as long as there is a row or column with negative sum; each time we do, we increase P. Can we increase it indefinitely? No, because there is a finite number of configurations which are possible (of size 2^{m+n}, since we can assign a sign =) to each row and each column); hence there is only a finite number of values for P! And so, it means that any sequence of operations must have an end. But the only way in which we can end it is if we can perform no more operations! And if we can perform no more operations, we have reached the target state: no row or column has negative sum. =) □

Problem 2. Given an $m \times n$ chocolate bar (which we can think of as a rectangle of area $m \times n$), we define an operation as follows: choose a chunk of area at least 2 and break it on a row or column.

We perform any sequence of operations until only individual squares remain.

Show that no matter what our choices are, it will always takes us $mn - 1$ breaks to reduce the chocolate to individual pieces.

Solution 2. This is a Type 3 problem. Each time we make a break, we increase the number of chunks by *exactly* 1. Since we start off with a single chunk, and the target state contains mn chunks, we must perform exactly $mn-1$ breaks each time, no matter how we choose them. =)

This easy problem has left many mathematicians of repute temporarily stupefied... =) □

Problem 3. Given an $n \times n$ board, we"infect" some $n - 1$ squares on it. The infection spreads as follows: at every step, a square becomes infected if it shares at least 2 edges with neighbors that are already infected. Prove that one cannot spread the infection to the whole board (i.e., after a while, no more infected squares will appear, and at least one square will still be "healthy").

Solution 3. This is a Type 4 problem. The quantity P here is the *perimeter* of the infection. In the beginning, P is at most $4(n-1)$ (as we have $n-1$ infected squares, and the maximum perimeter is achieved when none of them share a common edge).

Note that, as infection spreads to a new square, *the perimeter decreases.* Indeed, though perhaps counter-intuitive: if a square becomes infected, the edges that it shares with the infection will cease to be part of the perimeter, and the edges that it doesn't share with the infection will be added to the perimeter. Since there are at least 2 of the former and at most 2 of the latter (for a total of 4 edges), *the perimeter does not increase!*.

However, if the infection were to spread to the entire $n \times n$ board, it would have to cover a total perimeter $4n$. But since it cannot increase in perimeter, the infection will always be restricted to a perimeter less than (or equal to) $4(n-1)$. Hence it will NEVER spread to the entire board. =)

Note that if we started off with n squares instead, a diagonal configuration would do. =) □

Problem 4. Suppose we play the following game: starting at the origin of the plane, at each step, we are allowed to place a vector (line segment) of length 1 at the "current point", at any angle we wish. Then a computer opponent chooses whether to let it stay as is, or flip it by 180 degrees (see Figure 1). Can we eventually exit any large square about the origin?

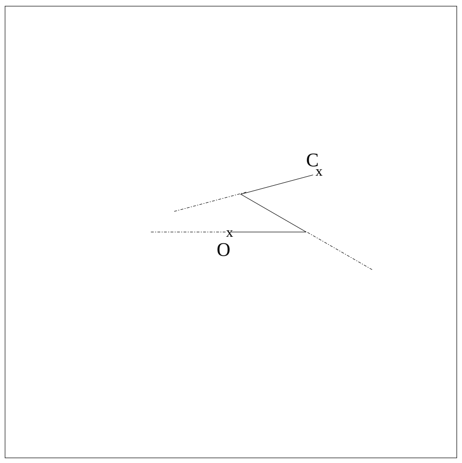

Figure 1: Can we exit the square? O is the origin, C is the current point, and the dotted lines represent options not taken by the computer.

Solution 4. Yes. This is a Type 1 problem. We can always exit the square by *always* increasing the distance at the current point from the origin. By choosing the line segment to always be perpendicular to the line OC, the distance OC (which in this case is our parameter P) will always increase, by the Pythagorean theorem! Moreover, this way we will be at distance 1 from the origin after the first step, at distance $\sqrt{2}$ after the second, $\sqrt{3}$ after the third, ..., $\sqrt{n}$ after the nth. Since the square is fixed, we will eventually exit the square. See how much better we are in Figure 2 than in Figure 1?...

Question: in how many steps will we exit the square? =) □

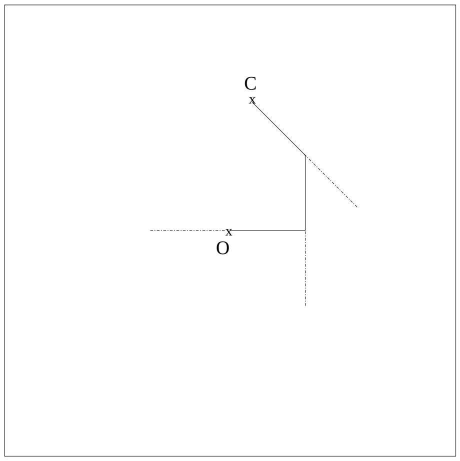

Figure 2: By always going perpendicular to OC with our next choice of angle, we increase the distance OC from the origin at every step!

Problem 5. Given a 7×7 checkerboard from which we cut out two opposite 2×2 corners, can we tile it with 1×2 (or 2×1) dominoes so that all of the board is covered exactly (i.e. nothing can be "sticking out")?

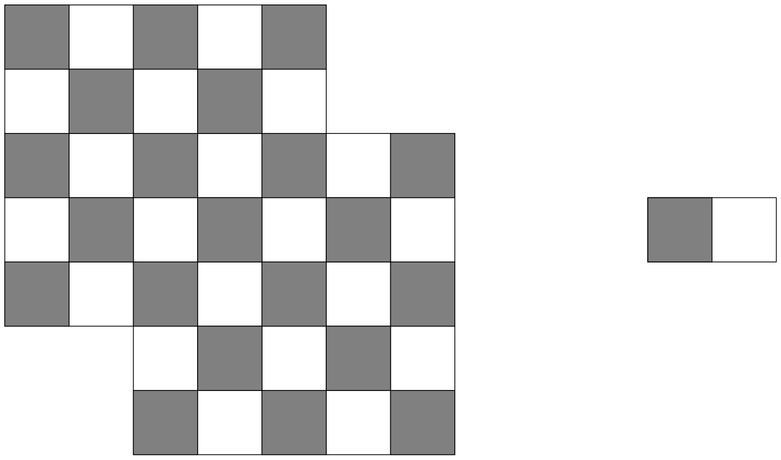

Figure 3: Checkerboard cut-out and the domino type we can use to tile it.

Solution 5. No. This is a Type 4 problem. To begin with, there are $49 - 2 \cdot 4 = 41$ squares, 21 of which are black, and 20 of which are white. Let P be the difference between the number of black squares and the number of white squares; in the beginning $P = 1$. By adding one tile we cover one black square and a white one, thus not changing P; however, if we were to tile the figure perfectly, we would have $P = 0$ (since it would be perfectly covered by dominoes). This is impossible, hence we cannot tile the checkerboard cut-out. =) $\square$

Problem 6. We are given 3 very large buckets, each of which has an integer number of ounces of fluid in it. Call the buckets A, B, and C, and the quantities in them a, b, and c (and assume safely that each bucket can hold more than $a + b + c$ ounces of fluid in it). At each step we can double the quantity of a bucket by pouring into it (from another bucket which contains at least as much fluid). For example, if $a \leq b \leq c$, we can pour a ounces of liquid into A from B, which leaves us with $2a$ ounces of liquid in A, $b - a$ ounces of liquid in B, and c ounces of liquid in C.

Show that we can always empty a bucket.

Solution 6. This is another Type 1 problem, with a slightly technical solution. Assume $a \leq b \leq c$, and let P be the minimum number of ounces of fluid in one of the three buckets. At first, $P = a$.

We describe here a sequence of operations which leads to a situation with a (strictly) smaller P. Let $b = na + r$, with $n \geq 1$ and $0 \leq r < a$. Let

$$n = \sum_{i=0}^{M} x_i 2^i \ ,$$

where each x_i is either 0 or 1 (we write n in base 2), and $x_M = 1$. Do $M + 1$ moves, as follows: at move i, if $x_i = 1$, pour from B into A; otherwise pour from C into A. The content of A is doubled each time, so that at the end A has $2^{M+1}a$ ounces of fluid in it. Moreover, since $c \geq b$, it follows that $c \geq b \geq 2^M a$, so it was always allowed to pour from B or C into A (because before the last move, A had $2^M a$ liquid ounces in it).

But then, at the end, we have r ounces of liquid in B, and since the new $P = r < a$, we have decreased P.

If the new P is 0 we're done, otherwise we relabel the buckets and repeat. □