

1. Non-Adaptive Weighing

1.1 Introduction

The first algorithm we cover in this class determines, given a scale and a set of coins in which one is 'bad' and weighs less or more than the others, which coin is bad and how its weight compares to the rest of the coins. The algorithm is non-adaptive, meaning that our procedure doesn't change to accommodate various intermediate results. Thus, given a number of coins, we can explicitly construct a series of weighings that is guaranteed to determine which coin is bad, and whether it is heavier or lighter than the rest.

1.2 Algorithm

A handy way to describe the weighing procedure is to have an i by j matrix, where i is the number of coins given and j is the number of weighings. The value of each cell is limited to -1, 0, or 1, which represents which side of the scale the coin is on. A value of -1 means one side of the scale, 1 means the other side, and 0 means the coin is not being placed on the scale for that weighing.

Populating the matrix with the weighing sequence is a simple problem, as the values can be chosen at random assuming four constraints are followed:

- 1) Each row must sum to zero. This rule represents that an equal number of coins must be placed on each side of the balance.
- 2) No two columns can be the same. If two columns were the same, that would mean that two coins were always on the same side whenever they were on the scale. If that were the case, determining which coin was the bad one would be impossible.
- 3) No two columns can be the inverse of each other (where -1 and 1 are inverses, and 0 is its own inverse). If this were the case, that would mean that two coins would always be on opposite sides of the scale at all times, and if one of those coins were bad it would be impossible to tell if one was heavy or the other was light.

Assuming these three conditions are met, we will be guaranteed to find our bad coin. The way we find our coin is by taking the series of scale results, where a -1 represents that the -1 side of the scale is heavier, a 1 represents the opposite, and a 0 represents that the scale balances. We take this sequence of numbers and look for a column in our matrix with either an identical or inverse sequence of numbers. If the sequence in the matrix is identical to the results sequence, then that means the coin represented by the column is heavier. Conversely, if the results sequence is the inverse of a certain column, that coin is then lighter than the rest.

One possible wrinkle that can arise is if one of the columns is comprised completely of 0's, meaning that the coin is never weighed, and then the scale balances in every weighing. Obviously the bad coin is the one not weighed, but we have no way of knowing if it's heavier or lighter than the rest. There's no solution to this except to add a

constraint that no column can be entirely 0's, although this reduces the possible number of coins that can be handled by a given number of weighings.

If a matrix cannot be created that satisfies the constraints for a given i by j matrix, then i must be reduced or j increased, corresponding to either adding more weighings or reducing the number of coins.

Let's look at the case with four coins and two weighings. A sample matrix could look like this:

	Coin 1	Coin 2	Coin 3	Coin 4
Weighing 1	1	-1	0	0
Weighing 2	1	0	-1	0

For example, if the weighting sequence is -1, -1, then we know that Coin 1 is lighter than the rest.

If we wanted to extend our scheme to handle more coins, an easy way is to add another round of Weighing. We have a very easy way of doing this: simply add another row and triplicate each column, adding a 1, 0, and -1 for the bottom cell.

	1	2	3	4	5	6	7	8	9	10	11	12
Weighing 1	1	1	1	-1	-1	-1	0	0	0	0	0	0
Weighing 2	1	1	1	0	0	0	-1	-1	-1	0	0	0
Weighing 3	1	-1	0	1	-1	0	1	-1	0	1	-1	0

Note that doing so creates a constraint violation in coins 10 and 11. This can be solved by removing them, which still satisfies the row sum condition. We are left with:

	1	2	3	4	5	6	7	8	9	10
Weighing 1	1	1	1	-1	-1	-1	0	0	0	0
Weighing 2	1	1	1	0	0	0	-1	-1	-1	0
Weighing 3	1	-1	0	1	-1	0	1	-1	0	0

1.3 Good Coins

A variant in Non-Adaptive Weighing is the introduction of a 'good coin', or a coin that is known at the outset to not be the bad coin. The addition of these coins changes the problem slightly. Say for example we had two coins. Determining which coin is bad would be impossible, with any number of weighings. This is reflected by our constraints, as putting both coins on each side of the scale would lead to two inverse columns, leaving one off would cause the row to not add up to zero, and leaving both off would once again result in the columns being the inverse of each other. Additionally, we can use our physical intuition: if you have two coins and a scale, whichever way the scale

tips you won't know if one is light or the other is heavy. However, by adding a known good coin, the problem becomes solvable.

	Good	1	2
Weighing 1	1	-1	0

The problem now is simply a matter of checking to see which direction the scale goes: -1 means a heavy first coin, 1 means a light first coin. In the case of 0, i.e. the second coin is the bad one, we will not be able to distinguish its weight. Note that the addition of a good coin leads to inverse columns, but that this is not a problem. The physical intuition behind this is that weighing two coins will prevent us from knowing if one coin is light or the other is heavy, but the addition of the good coin removes that confusion and helps us construct an optimal weighing scheme. An optimal weighing scheme is one that handles the maximum number of coins in a given weighing, where the maximum number of coins for n weighings is equal to $(3^n + 1)/2$.

With one weighing and one good coin, two coins is the maximum number of coins that can be tested. We can extend this to the next maximum by using the same procedure we used last time, of tripling each column.

	G	1	2	3	4	5	6	7	8
Weighing 1	1	1	1	-1	-1	-1	0	0	0
Weighing 2	1	-1	0	1	-1	0	1	-1	0

This time, the matrix produced violates the three constraints. Coins one and three are inverses, as are six and seven, and two and five. We have to remove one from each pair, while ensuring that the rows still sum to 0. In our example, that would mean to remove coins one, five, and six, leaving us with:

	G	1	2	3	4	5
Weighing 1	1	1	-1	-1	0	0
Weighing 2	1	0	1	-1	-1	0

Note that we've moved from the maximum number of coins that can be handled by one weighing to the maximum number of coins that can be handled by two weighings. To construct the optimal weighing matrix (i.e. for a given number of weighings handles the maximum possible number of coins) for k weighings, all that's needed is to:

- 1) Take an optimal matrix for $k-1$ weighings
- 2) Make three copies of each column
- 3) Add bottom cells alternating between 1, -1, and 0
- 4) Remove the three conflicting columns

Exercises

Exercise 1 Verify this last claim for $n = 9, 10$, and 11 ; by producing three-row matrices without a good coin having the said number of columns, obeying the conditions above.

- Scotty B. Ostler