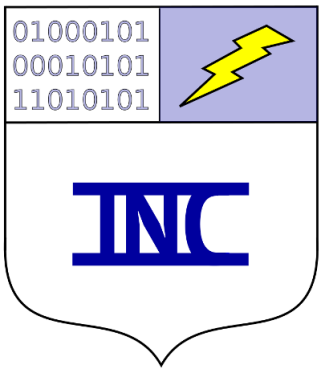
Expenditure Accounting Algorithm

Brent Kirkpatrick

December 20, 2023

Intrepid Net Computing



www.intrepidnetcomputing.com

Document Revision History

March 27, 2023 Conceived.

May 20, 2023 Revised.

June 29, 2023 Sketched.

December 20, 2023 Published original on www.bowtiecomputing.com

Abstract

Accounting algorithms have to be convenient and hand-computable. Given two account statements with the expenditures in different orders and the balances at different times, check for agreement. In this paper, we present a simple algorithm for this problem.

Algorithms for accounting are simple and straight-forward. To be useful, an accounting algorithm needs to be hand-computable. The problem that we look at here is one of two different accountings of the same expenditures, and we want to balance the accounts. The expenditures would be in a different order, and running balances given at different points.

An algorithm for this problem should be useful in the problem context and be simple. The beautiful idea of bipartite matching is useful for this problem. The simple act of matching up expenditures can help us balance the accounts.

Algorithm

We want to arrange the account data by date of the expenditure being recorded. This data can be imagined on a line. There are two accountings, so we have two sets of account data arranged by date. The expenditures are recorded in a different order.

We take a sliding window of each of accountings and compare them. These sliding windows should be either a span of dates or a span of a specific number of expenditures. Once the windows are chosen, we compare the expenditures from both accountings. This lets us check the balances at the beginning and end of each window in each account. See Figure 1 for the arrangement of the account data and the selection of sliding windows.

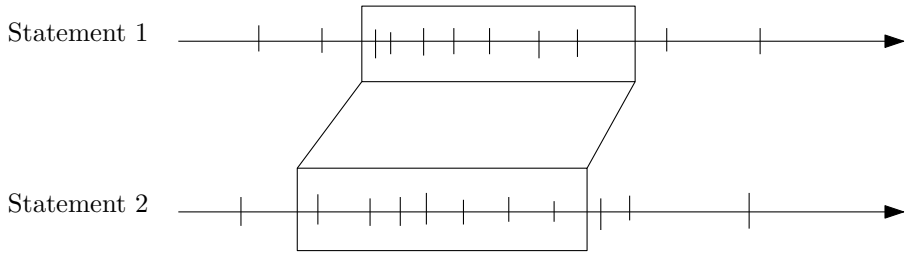


Figure 1: **Sliding window.** The accounting data are arranged on two lines for time. The expenditures are the marks on the lines. Each data set records the expenditures at different times. We select two sliding windows, on in each data set, to compare.

Once making the selection of sliding windows, the algorithm precedes by comparing the expenditures in the two windows with bipartite matching. The expenditures that have the same value are paired in the matching. The balances at the beginning and end of the sliding windows are compared. See Figure 2 for the matching of expenditures. The variables in the figure help with balancing the accounts.

The balances for the accounts in the sliding windows are given carefully. The x_i, y_i, z_i, w_i variables are the expenditures. The beginning and end of each window has a balance. These variables are A, B, C, D. We give the balances as

$$A - \sum x_i = p - B + \sum z_i$$

$$C - \sum y_i = p' - D + \sum w_i$$

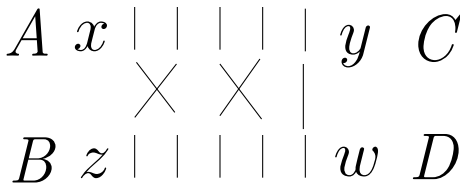


Figure 2: **Matching.** The slashes are the expenditures. The matching is given between expenditures that have the same value. The variables help with balancing the accounts.

a balanced statement means $p = p' + v(G)$ where $v(G)$ is the value of expenditures in the bipartite graph G. This means that $v(G)$ is the sum of all the matched expenditures in one half of the bipartite graph. Put again, $v(G) = \sum_{matched} x_i$ where the x_i are summed only if they match. A balanced statement means that all the expenditures are matched and the balances agree.

Correctness

The proof of correctness for the account expenditure algorithm relies in the properties of bipartite matching. This proof is instructive, as the problem essentially reduces to itself.

Lemma 1. *In the accounting data, when you remove the matched expenditures from the bipartite matching, the remaining data presents the same original problem.*

Proof. Take the data from the two accountings. Use any bipartite matching algorithm on the data. Remove the matched expenditures from both accountings. The remaining data contains expenditures in different orders, and only the expenditures that were not matched. This problem reduces to itself. □

This construction assumes that the same expenditures appear in both accountings. This means that both the accountings are faithful and correct.

A different circumstance happens if there is fraud or incorrect charges in the accountings. Such a circumstance could be addressed with algorithms. If we forgo the sliding windows and do a matching of the entire data sets, we can check for incorrect charges. When we match months of data and find unmatched charges with no pair in the bipartite matching, these charges might be fraud. Running balances can be compared to see if there were any other unmatched charges.

Computing By Hand

Accounting algorithms are only useful if they can be done by hand. This algorithm can be computed by hand. They key idea for computing by hand is Lemma 1 which tells us this can be done recursively. This time, we line up the figures from the expenditures in two columns. We check off in both columns the figures that match. Once the figures are checked off, they can be removed from the data set, and we can continue matching.

The power of this accounting algorithm is that it reduces to itself. Each time a pair of expenditures are matched, they can be removed from the data set and the matching can continue. We see this in Figure 3 where there are two columns of figures each from different accountings and the checked figures are matched in the other column. We can simply remove all the checked figures and continue looking for matches.

These accounting data can be compared by hand. Checking off the figures for the expenditures removes them from being matched. We only match un-checked figures for expenditures. We can match until no more figures can be matched. The unmatched figures are hopefully on a different statement or are fraud.

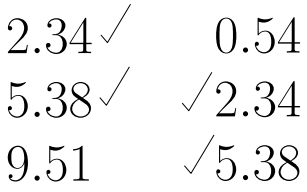


Figure 3: **Computing by hand.** The values of the expenditures are given in two columns. The values that match are checked off. Any pair of matched expenditures that are checked can be removed from the data set while matching continues.

The running balances can be compared after all the figures are matched. When we compare the running balances, we have to check carefully that all the same expenditures were matched for the balances on the two different accountings. It requires care to compare the balances. The running balances are not shown in the figure, but they have different values because of the different order of the expenditures. To compare the running balances, we do it like with the above algorithm where we use the matching expenditures. Once we compare the running balances, we can know that the accounts are balanced.

Conclusions

This paper gives an algorithm for balancing accounts. The algorithm is motivated by the problem of having two different accountings with expenditures in a different order. The computer algorithm uses sliding windows and checks the adjusted account balances. This is done with a bipartite matching. The algorithm is easily done by hand in columns of figures. This approach requires careful matching of the expenditures. The running balances are comparable. This paper gives a useful and simple algorithm for balancing accounts.