**Accounting Dilemma**

1. **Introduction**

Accordingly to the description, the given problem belongs to classical [Subset sum problem](http://en.wikipedia.org/wiki/Subset_sum_problem). Accordingly to the theory and the provided limitations, the practical solution must be based on a dynamic programming algorithm[[1]](#footnote-1).

1. **Proposed solution. Explanatory notes**

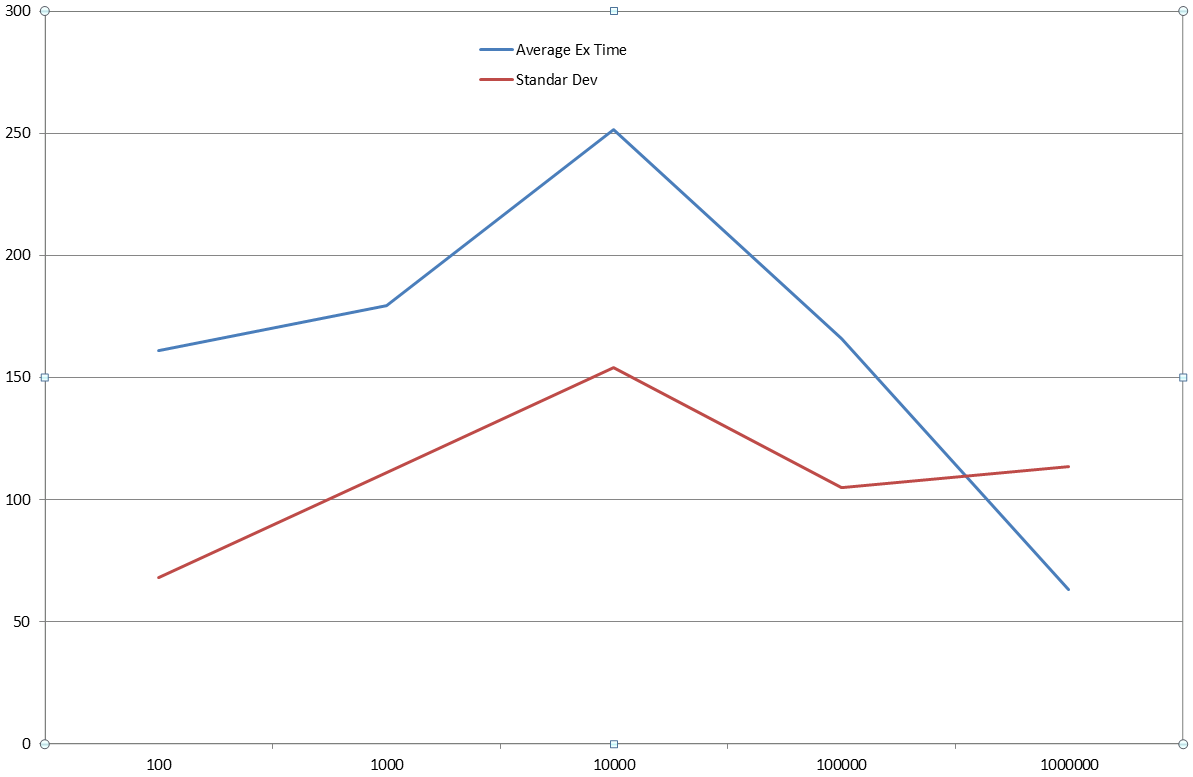
My algorithm (jar) performs calculation in the following manner:

1. It reads the file name as a string and considers the parameter as a correct one without extra checks.
2. It reads the data from file and checks if the given data represents float values. Note that it doesn’t also conduct an extra checks that number belongs to 2 place decimal format, firstly, because it doesn’t require and because in the accounting the rounding is common issue, so my algorithm rounds it automatically.
3. It converts the float data to integer in order to optimize calculation.
4. It screens out the data in order to do a computation for the suitable candidates that lay in specific range bounded by target sum. If by chance we can catch a solid one-to-one transaction then the algorithm stops.
5. The next step is an optimization of data (it is preferred to use integer array instead of Vector)
6. Then initializing the well-known dynamic programming algorithm and running it.
7. The last step is a storing the results.
8. **Algorithm analysis. Empirical evidence from simulations**

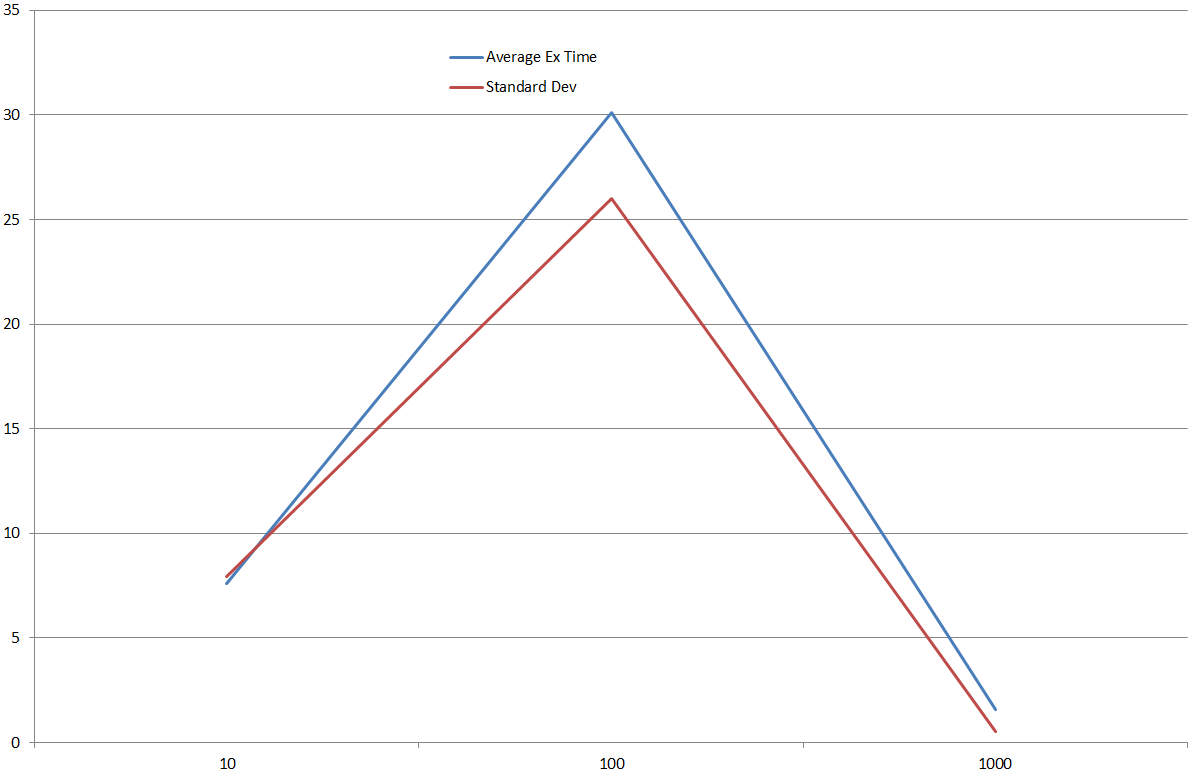
I have tested my program using data simulation. Normally accordingly to the central limit theorem the transaction data in the accounting general journal should represent a normal distribution. It is difficult to say something certain about its parameters (the mean and the standard deviation). Moreover, I’d rather say that these parameters a typical firm and industry based. Therefore, no hypothesis one can use here. That’s why to simulate the data I have used uniform distribution assumption about the transaction in the input journal.

Using 8GB RAM Dell PC under 64-bit Win7 which is equipped with Intel® Core™ i7-2600 CPU@ 3.4GZ and the simulated data, my performance analysis is shown on the next two Figures.

**There is a solution case**

****

**No solution case**

****

**Figure description**

**X axis – overall number of transaction in the file (note that it’s log-scale axis)**

**Y axis – time (measured in nanoseconds)**

**Blue line – average execution time**

**Red line – standard deviation of the execution time**

I start to measure a time just before converting the data to integer array and before initializing dynamic table. However, for the pure the case when we have a one-transaction match I consider the spent time equal zero. Each sample contains of 30 trails.

Note, that it naturally expects that for the bigger data the probability that there is matching a one-transaction case goes up, because the distribution is the uniform and the initial range data limit

1. **Concluding remarks**

Any developer may agree with Voltaire that *“Le mieux est l'ennemi du bien”* (The perfect is the enemy of the good)*,* although let’s think about some possible ways of the enhancing:

1. The combination of the greedy recursive algorithm with the DP[[2]](#footnote-2)-algorithm (a programming trick that similarly used to combine bubble sort algorithm and quick sort one). Thus, I can believe that it’s possible that for small data and for some architecture a greedy algorithm can be faster, but the switcher (a certain number of transactions starting from which the greedy algorithm starts underperform the DP-one) is a machine depended and must be chosen using empirical simulations.
2. Concerning the possibility of using Pisinger’s algorithm[[3]](#footnote-3) and others heuristic approximated solutions[[4]](#footnote-4)[[5]](#footnote-5). In case, when the transaction matching should be exact for our Accounting Problem first of all the need of such solution must be justified before. The second, in general, the good approximation or heuristic solution comes from the pre-analysis of the given data, without it it’s difficult to provide a solution, especially when it can guarantee the required result: a prefect accounting transaction matching.

Dear Sir or Madame thank you for paying attention to reading this report and playing with my jar and code. I also had a fun.

1. If *P* (the number of place values) is a small fixed number, then there are dynamic programming algorithms that can solve it exactly <http://en.wikipedia.org/wiki/Subset_sum_problem#Complexity> [↑](#footnote-ref-1)
2. Dynamic Programming [↑](#footnote-ref-2)
3. <http://www.diku.dk/~pisinger/95-1.pdf> [↑](#footnote-ref-3)
4. <http://www.academia.edu/4112866/Modified_Genetic_Algorithms_Based_Solution_to_Subset_Sum_Problem> [↑](#footnote-ref-4)
5. <http://www.cs.bham.ac.uk/~xin/papers/RohlfshagenYaoSoftComp10DOP.pdf> [↑](#footnote-ref-5)