

The Table That Solves Subset Sum Grows with the Size of the Numbers, Not with Their Count

— adding one digit multiplies the table by **9.277988** to **11.053571**, while the input length grows by only **62** to **69** bits — it is polynomial in the values of the numbers but exponential in the input length — the separator is whether one measures by input length or by the values of the numbers — [Paper 872]

Yuuki YAMAGISHI (山岸 勇輝)

Yamagishi Complexity Torus Lab, Japan

September 2026

YOI: <https://www.yamagishiyuuki.com/yoi/872>

— *From thinking to syncing.* —

Abstract

Subset sum is said to be solved quickly by building a table. This paper counts, varying the number of digits, that the table's size is set by the size of the numbers, not by how many there are. No new theorem or law is claimed. First, the table is proportional to the size of the numbers: with 20 numbers, the table has 1120 cells for 1 digit and 118980 for 3 digits (Section 2). Second, with more digits, the table outgrows all subsets: at 4 digits it is **1.141605** times all 2^{20} subsets, and at 6 digits **104.245529** times (Section 2). Third, and this is the core. Adding one digit multiplies the table but only adds to the input length: the table grows **9.277988** to **11.053571**-fold while the input length grows by only **62** to **69** bits (Section 3). Fourth, so the table's work is exponential in the input length. It is polynomial in the values, but the length needed to write a value is its logarithm, so measured by length it grows roughly tenfold per digit (Section 3). Fifth, the table's answers are correct: on 60 problems with 12 numbers, answers from the table matched answers from listing all sums (Section 2). Sixth, the separator is whether one measures by input length or by the values of the numbers: the same table looks polynomial by value and exponential by length; this is what the name **pseudo-polynomial time** means (Section 4). **Keywords:** subset sum problem, dynamic programming, pseudo-polynomial time, input length, computational complexity.

Scope of this paper (scope note): No new theorem or law is claimed: that solving subset sum with a table (dynamic programming) takes work proportional to the number of items times the target value, that this is called pseudo-polynomial time, and that the problem is counted among the generally hard ones, are all known. Complexity theory is not entered: the relation of P and NP and proofs of hardness are not entered. Real problems are not treated: packing goods or allocating budgets is not the subject. Fast implementations are not discussed: keeping the table as a bit string is a counting tool; speeds are not compared. Only random numbers were used: numbers came from seeded random draws, and the target was half the total. Relation to earlier papers: Paper 843 showed that the lower bound on comparisons comes from the number of answers, not from the input. This paper stands beside it. There, what sets the bound was separated. Here, the table's work is separated as set neither by the count of numbers nor by the input length, but by their values. What is added is laying out table size and input length for 1 to 6 digits, measuring 4 digits as where the table outgrows all subsets, measuring per-digit factors 9.277988 to 11.053571 and added bits 62 to 69, and placing the separator on input length versus values.

1. Notation and the Question

Setting: positive d -digit integers $a_1, \dots, a_{20}$ are drawn with a fixed random seed, and the target T is half their total (rounded down). Whether some subset sums exactly to T is decided with a table of “can the sum t be reached” for $t = 0, \dots, T$, updated number by number.

$$\text{table cells} = n(T + 1), \quad \text{input length} = \sum_{i=1}^n \lceil \log_2(a_i + 1) \rceil, \quad \text{all subsets} = 2^n$$

We took $n = 20$. Correctness was checked on 60 problems with 12 numbers against listing all sums.

Question: does building a table always solve subset sum quickly?

2. Table Size by Number of Digits

Computation 2.1: varying the digits d from 1 to 6, the table cells, the input length, and the table relative to all subsets were computed.

d	input length (bits)	table cells	table $\div 2^{20}$
1	59	1120	0.001068
2	125	12380	0.011806
3	190	118980	0.113468
4	259	1197060	1.141605
5	325	11781580	11.235790
6	387	109309360	104.245529

Observation 2.2

with more digits, the table outgrows all subsets. Up to 3 digits the table is smaller than all subsets; at 4 digits it is 1.141605 times, and at 6 digits 104.245529 times: the count of numbers stayed at 20. Only their size changed.

Observation 2.3

the table's answers are correct. On 60 problems with 12 numbers, answers from the table matched answers from listing all sums: the table gives up no correctness for speed. Only the work grows.

3. Growth per Added Digit

Computation 3.1: for each added digit, the factor on the table and the bits added to the input were computed.

digits	table factor	bits added
1 → 2	11.053571	66
2 → 3	9.610662	65
3 → 4	10.061019	69
4 → 5	9.842096	66
5 → 6	9.277988	62

Observation 3.2

this is the core. Adding one digit multiplies the table but only adds to the input length. The table grows 9.277988 to 11.053571-fold, while the input length grows by only 62 to 69 bits: when a value grows tenfold, the length needed to write it grows by only $\log_2 10$ bits. The table is proportional to the value, so measured by length it grows exponentially.

4. The Separator

Observation 4.1

the separator is whether one measures by input length or by the values of the numbers. Measured by value, the table's work is the count times the target, a polynomial: measured by input length, the same work is exponential in the length. Polynomial by value, exponential by length: this is what the name pseudo-polynomial time means, and the table is fast only while the numbers are small.

Observation 4.2

placed beside Paper 843.843 showed that the lower bound on comparisons comes from the number of answers, not from the input. There, what sets the bound was separated. Here the table's work was separated as set by the values, not by the count or the input length. Both are divides over what the work should be measured by.

Observation 4.3

what is not said. The relation of P and NP, real packing problems and fast implementations are not treated. What is shown is that the table solving subset sum grows with the size of the numbers.

5. Census and Closing

1. table cells $n(T + 1)$ (Section 1): **count times target.**
2. **1120 cells at 1 digit** (Computation 2.1): **small.**
3. **1.141605 times all subsets at 4 digits** (Computation 2.1): **where the table loses.**
4. **104.245529 times at 6 digits** (Computation 2.1): **still 20 numbers.**
5. table factor **9.277988 to 11.053571** (Computation 3.1): **the core.**
6. bits added **62 to 69** (Computation 3.1): **addition.**
7. **polynomial in value** (Observation 4.1): **count times target.**
8. **exponential in length** (Observation 3.2): **roughly tenfold per $\log_2 10$ bits.**
9. **answers agree on 60 problems** (Observation 2.3): **two ways of solving.**
10. **input length, or values** (Observation 4.1): **the separator; Paper 843.**

Closing: The table that solves subset sum grows with the size of the numbers, not with their count. With 20 numbers, the table is 1.141605 times all 2^{20} subsets at 4 digits and 104.245529 times at 6, because the table is proportional to the target value, and it grows with the values alone while the count stays fixed. Adding one digit multiplies the table by 9.277988 to 11.053571 while the input length grows by only 62 to 69 bits: polynomial in the values, exponential in the input length: that is what pseudo-polynomial time means. Placed among the earlier papers: beside Paper 843's "the bound comes from the number of answers, not the input", we placed that the table's work comes from the values of the numbers. To be honest: each digit count used one seeded random set, and the target was fixed at half the total.

References

1. Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.
2. Yamagishi, Y. (2026). *The Lower Bound on Comparisons Comes From the Number of Answers, Not From the Input* [Paper 843]. YOI: <https://www.yamagishiyuuki.com/yoi/843>

— *From thinking to syncing.* —

Yuuki YAMAGISHI (山岸 勇輝)

© 2026 Yamagishi Complexity Torus Lab, Japan · Licensed under Creative Commons Attribution 4.0 International (CC BY 4.0)