

部分和を解く表は、数の個数ではなく数の大きさに育つ

— 桁を一つ足すと表は **9.277988** 倍から **11.053571** 倍になるのに、入力の長さは **62** から **69** ビットしか増えない — 入力の長さに対しては指数で、数の値に対しては多項式である — 分離子は、入力の長さで測るか、数の値で測るか — [論文872]

山岸 勇輝 (Yuuki YAMAGISHI)

Yamagishi Complexity Torus Lab, Japan | 山岸勇輝ラボ

2026年9月

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

— *From thinking to syncing.* —

要旨

部分和の問題は、表を作れば速く解ける、と言われる。本稿は、その表の大きさが数の個数ではなく数の大きさに決まることを、桁数を変えて数える。新しい定理も法則も主張しない。第一に、表は数の大きさに比例する。**20** 個の数で、**1** 桁なら表は **1120** ます、**3** 桁なら **118980** ますである（第2節）。第二に、桁が増えると、表は全部の組を超える。**4** 桁で表は全部の組 **2²⁰** の **1.141605** 倍、**6** 桁で **104.245529** 倍になる（第2節）。第三に、これが本稿の芯である。桁を一つ足すと、表は掛け算で、入力の長さは足し算で増える。表は **9.277988** 倍から **11.053571** 倍になり、入力の長さは **62** から **69** ビット増えるだけである（第3節）。第四に、だから表の手間は、入力の長さに対しては指数である。数の値に対しては多項式でも、値を書く長さは値の対数なので、長さで測ると桁ごとに十倍前後になる（第3節）。第五に、表の答えは正しい。**12** 個の数で **60** 問、表で出した答えと、和を全部並べて出した答えが一致した（第2節）。第六に、分離子は、入力の長さで測るか、数の値で測るかである。同じ表が、値で測れば多項式、長さで測れば指数に見える。これが擬多項式時間という呼び名の中身である（第4節）。キーワード：部分和问题、動的計画法、擬多項式時間、入力の長さ、計算量。

本稿の射程（射程注記）：新しい定理も法則も主張しない。部分和の問題を表（動的計画法）で解くと手間が数の個数と目標の値の積に比例すること、それが擬多項式時間と呼ばれること、この問題が一般には難しい問題の一つとされることは、いずれも既知である。計算量の理論に入らない。P と NP の関係や、難しさの証明には入らない。実在の問題を扱わない。荷物の詰め方や予算の割り振りは主題ではない。速い実装を論じない。表をビット列で持つ工夫は数えるための道具で、速さを比べていない。無作為の数に限った。数は種を固定した乱数で作る、目標は合計の半分とした。既刊との関係：論文843 は、比べる回数の下界が入力ではなく答えの通り数で決まることを示した。本稿はその隣に置く。あちらは下界が何で決まるかを分けた。ここでは表の手間が数の個数でも入力の長さでもなく、数の値で決まることを分ける。加えたのは、桁数 1~6 で表の大きさと入力の長さを並べたこと、表が全部の組を超える桁 4 を測ったこと、桁ごとの倍率 9.277988~11.053571 と増えるビット 62~69 を測ったこと、分離子を「入力の長さか、数の値か」に置いたことである。

1. 記法と問い

設定： d 桁の正の整数 $a_1, \dots, a_{20}$ を種を固定した乱数で作る、目標 T を合計の半分（切り捨て）とする。和がちょうど T になる組があるかを、「和 t に届くか」を $t = 0, \dots, T$ について数ごとに更新する表で解く。

$$\text{表のます} = n(T+1), \quad \text{入力の長さ} = \sum_{i=1}^n \lceil \log_2(a_i + 1) \rceil, \quad \text{全部の組} = 2^n$$

$n = 20$ とした。正しさは、12 個の数で、和を全部並べる方法の答えと 60 問で比べた。

問い：表を作れば、部分和の問題はいつでも速く解けるのか。

2. 桁数ごとの表の大きさ

計算 2.1：桁数 d を 1 から 6 まで変えて、表のます、入力の長さ、全部の組に対する表の比を出した。

d	入力の長さ（ビット）	表のます	表 $\div 2^{20}$
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

観察 2.2

桁が増えると、表は全部の組を超える。3 桁までは表が全部の組より小さいが、4 桁で 1.141605 倍、6 桁で 104.245529 倍になる。数の個数は 20 のまま変えていない。変えたのは数の大きさだけである。

観察 2.3

表の答えは正しい。12 個の数で 60 問、表で出した答えと、和を全部並べて出した答えが一致した。表は速さのために正しさを捨てていない。大きくなるのは手間だけである。

3. 桁を一つ足したときの増え方

計算 3.1：桁を一つ足したとき、表が何倍になり、入力の長さが何ビット増えるかを出した。

桁	表の倍率	増えたビット
1 $\rightarrow$ 2	11.053571	66
2 $\rightarrow$ 3	9.610662	65
3 $\rightarrow$ 4	10.061019	69
4 $\rightarrow$ 5	9.842096	66
5 $\rightarrow$ 6	9.277988	62

観察 3.2

これが本稿の芯である。桁を一つ足すと、表は掛け算で、入力の場合は足し算で増える。表は **9.277988** 倍から **11.053571** 倍になり、入力の長さは **62** から **69** ビット増えるだけである。値が十倍になっても、それを書く長さは $\log_2 10$ ビットずつしか伸びない。表は値に比例するので、長さで測れば指数で育つ。

4. 分離子

観察 4.1

分離子は、入力の長さで測るか、数の値で測るかである。数の値で測れば、表の手間は個数と目標の積で、多項式である。入力の長さで測れば、同じ手間が長さの指数になる。値では多項式、長さでは指数。これが擬多項式時間という呼び名の中身で、表が速いのは数が小さいときに限られる。

観察 4.2

論文843の隣に置く。843は、比べる回数の下界が入力ではなく答えの通り数で決まることを示した。あちらは下界が何の量で決まるかを分けた。ここでは表の手間が、数の個数でも入力の長さでもなく数の値で決まることを分けた。どちらも「手間を何で測るか」の分け目である。

観察 4.3

言わないこと。PとNPの関係も、実在の詰め込み問題も、速い実装も扱わない。示したのは、部分和を解く表が数の大きさに育つということである。

5. 点呼と結び

1. 表のまず $n(T+1)$ (第1節) : 個数と目標の積。
2. 1桁で **1120** ます (計算 2.1) : 小さい。
3. 4桁で全部の組の **1.141605** 倍 (計算 2.1) : 表が負ける桁。
4. 6桁で **104.245529** 倍 (計算 2.1) : 個数は **20** のまま。

5. 表の倍率 **9.277988**～**11.053571** (計算 3.1) : 本稿の芯。
6. 増えたビット **62**～**69** (計算 3.1) : 足し算。
7. 値に対して多項式 (観察 4.1) : 個数と目標の積。
8. 長さに対して指数 (観察 3.2) : $\log_2 10$ ビットごとに十倍前後。
9. **60** 問で答えが一致 (観察 2.3) : 別々の解き方。
10. 入力 of 長さか、数の値か (観察 4.1) : 分離子・論文**843**。

結び : 部分和を解く表は、数の個数ではなく数の大きさに育つ。**20** 個の数で、**4** 桁なら表は全部の組 2^{20} の **1.141605** 倍、**6** 桁なら **104.245529** 倍になる。表は目標の値に比例し、数の個数は変えていないのに、値だけで大きくなるからである。桁を一つ足すと表は **9.277988** 倍から **11.053571** 倍になり、入力の長さは **62** から **69** ビットしか増えない。値に対しては多項式でも、入力の長さに対しては指数であり、これが擬多項式時間の中身である。既刊との位置 : 論文**843** の「下界は入力ではなく答えの通り数で決まる」の隣に、表の手間が数の値で決まることを置いた。正直に言えば、数は種を固定した乱数一組ずつで、目標は合計の半分に限った。

参考文献

-
1. Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.
 2. 山岸勇輝 (2026). 『比べる回数の下界は、入力ではなく答えの通り数で決まる』[論文**843**]. YOI:
<https://www.yamagishiyuuki.com/yoi/843>
-

— *From thinking to syncing.* —

山岸 勇輝 (Yuuki YAMAGISHI)

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