

Programming C++

Project Ulam-Folge

Status:

13. Mai 2022, 17:36

Supervisor: Prof.Dr. G. Haase,

`gundolf.haase@uni-graz.at`

Im Jahr 1964 stellte der polnische Mathematiker STANISŁAW ULAM¹ eine Folge von ganzen Zahlen vor, deren mysteriöse sogenannte Quasi-Periodizität wir uns bis heute nicht erklären können. Diese Zahlenfolge $(a_n)_{n \in \mathbb{N}}$, die in der Fachwelt als (u, v) -Ulam-Folge² bekannt ist, definieren wir rekursiv mit

$$a_1 := u, a_2 := v$$

und a_n als die kleinste natürliche Zahl größer als a_{n-1} , die **eindeutig** als Summe zweier Folgeglieder $a_i + a_j$ mit $i, j < n$ und $i \neq j$ gebildet werden kann.

Für ersten vier Glieder der $(1, 2)$ -Ulam-Folge berechnen wir demnach

$$\begin{aligned} a_1 &= u := 1, & a_2 &= v := 2 \\ a_3 &= a_1 + a_2 = 1 + 2 = \mathbf{3}, & a_4 &= a_1 + a_3 = 1 + 3 = \mathbf{4}. \end{aligned}$$

Da wir 5 sowohl durch $a_1 + a_4 = 1 + 4 = \mathbf{5}$ als auch durch $a_2 + a_3 = 2 + 3 = \mathbf{5}$ darstellen und somit **nicht eindeutig** als Summe zweier bereits berechneter Folgeglieder bilden können, folgt $a_5 = 2 + 4 = \mathbf{6}$ als fünfte $(1, 2)$ -Ulam-Zahl der Folge.

1. Implementieren Sie eine Funktion `count_distinct_sums(summands, sum)`, die die Anzahl der Summanden (s_i, s_j) , für die $\text{sum} = s_i + s_j$ mit $s_i, s_j \in \text{summands}$ und $i \neq j$ gilt, bestimmt und an das aufrufende Programm zurückgibt. 1 Pkt.
2. Nutzen Sie die von Ihnen bereits implementierte Funktion `count_distinct_sums` um eine weitere Funktion `compute_ulam_numbers(upper_bound)` zu schreiben, die beginnend mit $u := 1$ und $v := 2$ alle darauffolgenden Folgeglieder kleiner oder gleich `upper_bound` der $(1, 2)$ -Ulam-Folge berechnet und an das aufrufende Programm retourniert. 2 Pkt.
3. Testen Sie Ihre Funktion `compute_ulam_numbers` indem Sie die für verschiedene Werte von `upper_bound` zurückgegebenen Zahlenfolgen mit der Folge A002858³ der „On-Line Encyclopedia of Integer Sequences“ (OEIS) vergleichen.
4. Erweitern Sie Ihre Funktion `compute_ulam_numbers`, sodass Sie u und v , mit $u < v$, der allgemeinen (u, v) -Ulam-Folge als Argumente `u` und `v` der Funktion `compute_generalized_ulam_numbers(u, v, upper_bound)` übergeben können. Bestimmen Sie mithilfe dieser Funktion in weiterer Folge alle Glieder kleiner oder gleich 100 der $(3, 8)$ -Ulam-Folge. 1 Pkt.
5. Die Methode, sukzessive die Anzahl von eindeutigen Summen potentiell nächster Folgeglieder zu bestimmen, ohne auf Ergebnisse bereits durchgeführte Berechnungen zurückzugreifen, ist nicht sonderlich effizient. Überprüfen Sie, ob

¹https://de.wikipedia.org/wiki/Stanis%C5%82aw_Marcin_Ulam

²<https://mathworld.wolfram.com/UlamSequence.html>

³<https://oeis.org/A002858>

133 325 eine $(1,2)$ -Ulam-Zahl ist, indem Sie Ihre implementierte Funktionen `count_distinct_sums` und `compute_generalized_ulam_numbers` zu einer neuen `compute_generalized_ulam_numbers_fast(u, v, upper_bound)` zusammenfassen, um die Laufzeit Ihres Programms zu verbessern.

2±1 Pkt.