Project Euler 23. Non-Abundant Sums

hiragn

2024 年 12 月 23 日

1. 問題の概要

> 真の約数の和がその数よりも大きい数を過剰数という。
> 28123 より大きい任意の整数は 2 つの過剰数の和（同じ数 2 つを足す場合も含む）であらわせることが知られている。
> 2 つの過剰数の和で書き表せない正の整数の総和を求めよ。
>
> https://projecteuler.net/problem=23

2. 二分探索する解法

20161 より大きい数はすべて過剰数の和であらわされるらしいので，28123 を 20161 に置き換えた問題を考えます。[*1] 20161 以下の過剰数は 4994 個あります。

```
In[]:= Length@Select[Range@20161, DivisorSigma[1, #] > 2 # &]
Out[]= 4994
```

2 つの過剰数の和で表せる数の集合 st を作って，その総和を 1〜20161 の総和から引く方針で解きます。

abundant の i 番目の要素 x と j 番目の要素を足すとします（$i \leqq j$）。

x に足せる数は $20161 - x$ 以下の数です。過剰数のリスト abundant がソート済みなことを利用して境目となる数を二分検索してそのインデックスを $j_{\max}$ とします。

j の範囲は $i \leqq j \leqq j_{\max}$ です。x に abundant の i 番目から $j_{\max}$ 番目までを足した数のリストを Union で集合 st に入れると自動的に重複が取り除かれていって，最終的に st は 2 つの過剰数の和のリストになります。

その総和を 1〜20161 の総和から引いたものが答えです。

[*1] http://mathworld.wolfram.com/AbundantNumber.html

```
In[]:= Clear["Global`*"];
RepeatedTiming[
 nmax = 20161;
 abundant = Select[Range@nmax, DivisorSigma[1, #] > 2*# &];
 st = {};
 Do[
  x = abundant[[i]];
  If[2 x > nmax, Break[]];
  j = ResourceFunction["BinarySearch"][abundant, nmax - x];
  st = Union[st, Take[abundant, {i, j}] + x], {i, Length@abundant}];
 ans = Total@Range@nmax - Total@st]

Out[]= {1.14685, 4179871}
```

3. 二分探索しない解法

二分探索しない解法もためしてみました。2 つの過剰数の和は「自分自身との和」「自分と違うものとの和」にわけられます。

これらを For ループや Do ループを使わずに求めると[*2]，意外に速く終わります。

```
In[]:= Clear["Global`*"];
RepeatedTiming[
 nmax = 20161;
 abundant = Select[Range@nmax, DivisorSigma[1, #] > 2 # &];
 st = Select[2*abundant, # <= nmax &];
 st = Union[st,
   Select[DeleteDuplicates[
     Total /@ Subsets[abundant, {2}]], # <= nmax &]];
 ans = Total@Range@nmax - Total@st]

Out[]= {1.40682, 4179871}
```

[*2] mathematica はこういうループ処理が苦手