计算物理作业 1

杨远青 22300190015

2024 年 9 月 14 日

1 题目 1：五次幂丢番图方程

1.1 题目描述

Find all integer solutions to the **Diophantine equation** $a^5 + b^5 + c^5 + d^5 = e^5$ within the range $[0, 200]$.

1.2 程序描述

1967 年，Lander 和 Parkin 使用 CDC 6600 计算机首次找到方程

$$a^5 + b^5 + c^5 + d^5 = e^5$$

的一个解，即

$$27^5 + 84^5 + 110^5 + 133^5 = 144^5$$

这个解也推翻了欧拉的幂和猜想（即至少需要 k 个 k 次幂才能表示另一 k 次幂的和，且 $k \geq 2$）。相关参考资料可参见 Diophantine Equation—5th Powers。2004 年，J. Frye 使用分布式并行计算找到了第二个解：

$$55^5 + 3183^5 + 28969^5 + 85282^5 = 85359^5$$

题目要求我们找到所有满足 $0 \leq a \leq b \leq c \leq d < e \leq 200$ 的整数解

为了解决这个问题，我们首先采用暴力搜索的方法，遍历所有可能的 a, b, c, d, e 组合，并逐一检查是否满足方程。在 `brute_force.f90` 中实现了这一方法，其伪代码见算法 1。该程序的运行时间达到了 8.953 秒，速度较为缓慢。

在网络上寻找更优解法时，我偶然发现了一个专门讨论整数幂方程的论坛：Euler Free。该论坛提到可以通过数论优化的方法。注意到方程中的 x^5 是齐次的，首先注意到到费马小定理：

$$a^{p-1} \equiv 1 \pmod{p}, \quad \gcd(a, p) = 1.$$

基于此，在搜索时可以将步长扩大到 5。然而，进一步查阅 Lander 和 Parkin 的论文后，未能找到有效的进一步启发。在 Rosetta Code 网站上，我发现了 C 语言中使用的Mod30 Trick，这让我意识到可以进一步扩展费马小定理的应用范围。即我们可以证明：

$$a^5 \equiv a \pmod{30}$$

这意味着在搜索时可以将步长扩展到 30。（昨天使用这个问题测试过了 ChatGPT o1 pre，它枚举证明了所有情况。）具体而言，对于 $n = 2, 3, 5$，我们可以分别讨论奇偶性和模数的同余关系：- 对于 $n = 2$，由于奇偶性，显然有 $a^5 \equiv a \pmod 2$。

- 对于 $n = 3$：- 若 $\gcd(a, 3) = 1$，则 $a^2 \equiv 1 \pmod 3$，因此

$$a^5 = a \cdot (a^2)^2 \equiv a \cdot 1^2 \equiv a \pmod 3$$

- 若 $3 \mid a$，则 $a \equiv 0 \pmod 3$，因此 $a^5 \equiv 0 \pmod 3$。- 对于 $n = 5$：- 若 $\gcd(a, 5) = 1$，则 $a^4 \equiv 1 \pmod 5$，因此

$$a^5 = a \cdot a^4 \equiv a \cdot 1 \equiv a \pmod 5$$

- 若 $5 \mid a$，则 $a \equiv 0 \pmod 5$，因此 $a^5 \equiv 0 \pmod 5$。

根据剩余定理，我们立马可以得出：$a^5 \equiv a \pmod{30}$

利用这个技巧，我们得到了算法 2，经测试，其 Fortran 实现的运行时间缩短至 0.797 秒。此外，通过反向查找，即从 e 开始，结合 mod30 技巧，我们进一步优化了算法，得到了算法 3，运行时间进一步缩短至 0.438 秒。

尽管运行时间有所改善，但 mod30 技巧只是常数级别的优化，不能显著加快找到第二个非平凡解的速度。为了进一步降低算法的复杂度，我们尝试将暴力搜索的时间复杂度从 $\mathcal{O}(N^5)$ 降低至 $\mathcal{O}(N^3)$ 。最自然的想法是使用哈希表和分治法，即借助 Hash Table 存储单个元素的幂和双元素幂和的键对，将线性搜索的 $\mathcal{O}(N)$ 成本转嫁到建表操作上，再分治匹配。

遗憾的是，Fortran 对哈希表的支持较为有限，通常需要依赖外部库，而我对其实现原理尚不完全理解。在 C++ 中借助 llu(unsigned long long int) 构造大质数，似乎可以减少碰撞并建立一个简易的 Hash Table, 并借助拓展的牛顿法快速求五次根, 据此对 a, b, c, d 的搜索剪枝, 可惜这两个技巧我并没成功在 Fortran 中复现。

不过我决定使用 Python 进行关于哈希表加速的对比研究，因为其内置了优化后的字典类型，并通过 gpt 建议的多线程进一步优化，最终将运行时间从 3.929s 加速到了 0.056s。

Codes 文件夹中有算法 1-3 的 Fortran 实现，其目录内使用 `gfortran -o brute_force -brute_force.f90` `gfortran -o mod30 -mod30_trick.f90` `gfortran -o mod30v2 -mod30_trick_reverse.f90` 等命令可以编译。编译选项 $-o$ 后面的程序名可自选，再执行即可。

以及算法 4/5 的 Python 实现，其目录内使用 `python -u brute_force.py` 、 `python -u hash_quick.py` 编译后运行。

1.3 伪代码

Algorithm 1: Brute-force solution to the Diophantine equation

Input: N: Integer (the upper bound, $N = 200$)

Output: *solutions*: List of tuples (a, b, c, d, e) where $0 \leq a \leq b \leq c \leq d < e \leq N$

```
for a ← 0 to N do
    for b ← a to N do
        for c ← b to N do
            for d ← c to N do
                for e ← d + 1 to N do
                    if a^5 + b^5 + c^5 + d^5 = e^5 then
                        solutions.append((a, b, c, d, e)) ;    // Store the solution tuple
                    end
                end
            end
        end
    end
end
return solutions
```

Algorithm 2: Mod30 trick for solving the Diophantine equation

Input: N: Integer (the upper bound, $N = 200$)
Output: *solutions*: List of tuples (a, b, c, d, e) where $1 \leq a \leq b \leq c \leq d < e \leq N$

for $a \leftarrow 1$ **to** N **do**
  for $b \leftarrow a$ **to** N **do**
    for $c \leftarrow b$ **to** N **do**
      for $d \leftarrow c$ **to** N **do**
        $r\_left \leftarrow \mathrm{mod}(a + b + c + d, 30)$; `// Compute remainder for` e
        $e\_start \leftarrow d + ((r\_left - d) \bmod 30)$; `// Ensure` $e \geq d$ `and` $a + b + c + d \equiv e \pmod{30}$
        for $e \leftarrow e\_start$ **to** N **do**
          if $(e - e\_start) \bmod 30 = 0$ **then**
            `// Increase the step size to 30`
            if $a^5 + b^5 + c^5 + d^5 = e^5$ **then**
              solutions.append((a, b, c, d, e))
            end
          end
        end
      end
    end
  end
end
return *solutions*

Algorithm 3: Reverse Mod30 trick for solving the Diophantine equation

Input: N: Integer (the upper bound, $N = 200$)
Output: *solutions*: List of tuples (a, b, c, d, e) where $1 \leq a \leq b \leq c \leq d < e \leq N$

for $e \leftarrow N$ **to** 1 **do**
  for $d \leftarrow e$ **to** 1 **do**
    for $c \leftarrow d$ **to** 1 **do**
      for $b \leftarrow c$ **to** 1 **do**
        $a\_min \leftarrow (e - d - c - b) \bmod 30$; `// Compute minimal` a `using modular arithmetic`
        if $a\_min \leq 0$ **then**
          $a\_min \leftarrow a\_min + 30$; `// Adjust` a_min `to fit within the modulus`
        end
        for $a \leftarrow a\_min$ **to** b **do**
          if $(a - a\_min) \bmod 30 = 0$ **then**
            if $a^5 + b^5 + c^5 + d^5 = e^5$ **then**
              solutions.append((a, b, c, d, e))
            end
          end
        end
      end
    end
  end
end
return *solutions*

Algorithm 4: Brute-force solution using a single hash

Input: *limit*: Upper bound

Output: *solution*: Tuple (a, b, c, d, e) or `None`

$pow\_5 \leftarrow [n^5$ for n in $[0, limit]]$; // List of n^5 values
$pow5\_to\_n \leftarrow \{n^5 : n$ for n in $[0, limit]\}$; // Dictionary mapping n^5 to n
for $a \leftarrow 1$ **to** *limit* **do**
  for $b \leftarrow a$ **to** *limit* **do**
    for $c \leftarrow b$ **to** *limit* **do**
      for $d \leftarrow c$ **to** *limit* **do**
        $pow\_5\_sum \leftarrow pow\_5[a] + pow\_5[b] + pow\_5[c] + pow\_5[d]$ **if** pow_5_sum *in* $pow5\_to\_n$ **then**
          $e \leftarrow pow5\_to\_n[pow\_5\_sum]$ **return** (a, b, c, d, e)
        end
      end
    end
  end
end
return *None*

Algorithm 5: Parallel brute-force solution using double hash

Input: *limit*: Upper bound

Output: *solution*: Tuple (a, b, c, d, e) or `None`

$power\_5 \leftarrow \{i^5 : i$ in $[1, limit]\}$; // Dictionary of fifth powers
$sum2 \leftarrow \{\}$; // Dictionary to store sums of two fifth powers
for $i \leftarrow 1$ **to** *limit* // Parallelized computation **do**
  $a5 \leftarrow i^5$ **for** $j \leftarrow i$ **to** *limit* **do**
    $sum2[a5 + j^5] \leftarrow (i, j)$
  end
end
$sk \leftarrow$ sorted keys of $sum2$ **foreach** p *in sorted(power_5.keys())* **do**
  foreach s *in* sk **do**
    if $p \leq s$ **then**
      break ; // Exit the loop early if no further solutions are possible
    end
    if $p - s$ *in* $sum2$ **then**
      return $(power\_5[p], sum2[s], sum2[p - s])$
    end
  end
end
return *None*

1.4 输入输出示例

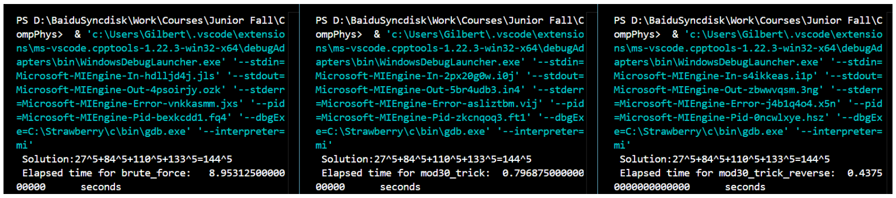

图 1: 程序运行结果

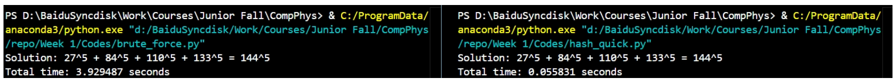

图 2: 程序运行结果