SMOOTHFAC

1. Problem

Let N be a natural number and let t be a fixed threshold. The problem is to find a subfactorization of $N!$ with admissible factors from the set $J = \{t, t+1, \ldots, N\}$ such that

$$\prod_{j \in J} j^{x_j} \Big| N!$$

where the x_j are non-negative integers counting how often j is included in the product (the notation $a|b$ means a divides b).

The objective is to maximize the total number of factors used, *i.e.* the sum of the x_j.

A valid subfactorization of $N!$ satisfies

$$\sum_{j \in J} v_p(j) \cdot x_j \le v_p(N!) \qquad \forall p \in \Pi_N \tag{1.1}$$

where Π_N is the set of primes less than or equal to N and $v_p(j)$ counts the number of times the prime p divides the number j.

The objective together with the constraints for each prime p define an integer linear program. When relaxing the x_j to be non-negative reals, the problem becomes a linear program. From any feasible value x of the linear program, one can recover a subfactorization of $N!$ as

$$\prod_{j \in J} j^{\lfloor x_j \rfloor}.$$

The dual problem the the above linear program is to find weights w_p that minimize

$$\sum_{p \in \Pi_N} v_p(N!) \cdot w_p$$

subject to the constraints

$$\sum_{p \in \Pi_N} v_p(j) \cdot w_p \ge 1 \qquad \forall j \in J \tag{1.2}$$

2. Algorithm

For further discussion, it is useful to switch to matrix notation. Let F be the matrix with elements $(F)_{ij} = \nu_i(j)$ for $i \in \Pi_N$ and $j \in J$, let c be the vector with elements $(c)_i = \nu_i(N!)$ for $i \in \Pi_N$ and let e be the vector of all ones (with conforming dimension). Then, the problem can be written as

$$\begin{aligned} \max_x \quad & e^\mathsf{T} x \\ \text{s.t.} \quad & Fx \leq c \\ & x \geq 0. \end{aligned} \tag{2.1}$$

The algorithm has three phases. It partitions the factors $J = J_S \uplus J_R$ where J_S is the set of $\lceil\sqrt{N}\rceil$-smooth numbers (i.e. all prime divisors of $j \in J_S$ are smaller than or equal to $\lceil\sqrt{N}\rceil$), and J_R contains all factors with a prime divisor larger than $\lceil\sqrt{N}\rceil$. The first phase deals with the non-smooth factors J_R heuristically; the second phase handles the smooth factors J_S using linear programming; the third phase handles the residual prime divisors not used by the subfactorization resulting from the first two phases.

Phase 1. Partition the prime divisors in small primes $\Pi_S = \Pi_{\lceil\sqrt{N}\rceil}$ and large primes $\Pi_L = \Pi_N \setminus \Pi_S$. This allows to rewrite the inequality $Fx \leq c$ in block matrix form as

$$\begin{bmatrix} F_{S,S} & F_{S,R} \\ 0 & F_{L,R} \end{bmatrix} \cdot \begin{bmatrix} x_S \\ x_R \end{bmatrix} \leq \begin{bmatrix} c_S \\ c_L \end{bmatrix}.$$

The first step of the algorithm is to fix values for x_R greedily as follows. Partition the non-smooth factors J_R according to their largest prime divisor

$$J_R = \biguplus_{p \in \Pi_L} J_p \qquad \text{with} \qquad J_p = \{ j \mid \nu_p(j) = 1 \}$$

Choose for each J_p its *smallest* element and assign it the full weight c_p of the right hand side

$$x_j = \begin{cases} \nu_p(N!) & \text{if } j = p \cdot \lceil t/p \rceil \\ 0 & \text{otherwise,} \end{cases} \qquad j \in J_p,\ p \in \Pi_L.$$

This heuristic can be justified *post hoc* by observing that the optimal dual multipliers $w^\star$ of the original linear program (2.1) scale as

$$w_p^\star \approx \log p / \log t \qquad p \in \Pi_S.$$

Fixing $w_p = \log p / \log t$, $p \in \Pi_S$ makes the choice $j = p \cdot \lceil t/p \rceil$ optimal among all $j \in J_p = \{ p \cdot (\lceil t/p \rceil + k) \mid k = 0, 1, \dots \}$.

Phase 2. Fixing x_R as above allows to *deflate* the problem and work with a reduced linear program

$$\begin{aligned} \max_{x_S} \quad & e^\mathsf{T} x_S \\ \text{s.t.} \quad & F_{S,S} x_S \le d_S \\ & x_S \ge 0. \end{aligned} \tag{2.2}$$

where the deflated right hand side $d_S = c_S - F_{S,R} \cdot x_R \ge 0$ [PROOF NEEDED]. This linear program has only $\pi(\lceil\sqrt{N}\rceil)$ constraints which is a significant reduction from the original $\pi(N)$ constraints.

The reduced linear program can be efficiently solved using a sifting strategy. Start solving the problem with a subset of columns W (the *working set*) containing the $2\sqrt{N}$ smallest elements of J_S. After each solve, scan for columns not included in the working set having positive reduced cost

$$e - w_W^\star \cdot F_{S,S\setminus W}$$

where $w_W^\star$ are the optimal dual variables from the problem with working set W. Scanning is done from smallest to largest element in $J_S \setminus W$. Whenever a batch of 200 improving columns is found, reoptimize the problem with the columns added to the working set. When no more improving columns are found, the solution is optimal.

Phase 3. Let $x_S^\star$ be an optimal solution of the reduced linear program (2.2). The final step of the algorithm is to greedily use the residual prime divisors $d_S - F_{S,S} \cdot \lfloor x_S^\star \rfloor$ to find additional factors $j \ge t$. To this end, pick the largest available prime divisor p and scan if any of the factors $p \cdot (\lceil t/p \rceil + k)$, for $k = 0, 1, \ldots$ divides the product of the remaining prime divisors. If yes, add the factor, remove the divisors and reiterate; otherwise stop.

3. Implementation

The implementation attempts to be (somewhat) space efficient. The bulk of memory is used for one list `f[]` of length N. The elements stored in `f[]` are in the interval $[-\sqrt{N}, \sqrt{N}]$ which allows to use 32-bit signed integers for the desired range $N \le 10^{11}$. All other lists and dictionaries have size $O(\sqrt{N})$, except the explicit representation of the linear program which is (roughly) $O(\sqrt{N} \cdot \log N)$ (assuming the size of the working set is a bounded multiple of $\sqrt{N}$).

The list `f[]` is used to represent the matrices $F_{S,S}$, $F_{S,R}$ and to store the divisor counts $v_p(N!)$ of the large primes $p \in \Pi_L$. The encoding is as follows:

$$\texttt{f[j]} = \begin{cases} \text{smallest } p|j & \text{if } j \in J_S \\ -v_j(N!) & \text{if } j \in \Pi_L \\ -\lceil t/p \rceil & \text{if } j = p \cdot \lceil t/p \rceil,\, p \in \Pi_L,\, p < t \\ 0 & \text{otherwise} \end{cases}$$

The sign of `f[]` encodes whether j is smooth or non-smooth. To determine whether $j \in \Pi_L$ for $j \geq t$, a trial division `j/-f[j]` is required.

The columns of $F_{S,S}$ can be recovered by repeated division `j ← j / f[j]`.

The same is possible for the columns of $F_{S,R}$ for which the corresponding variable in x_R is non-zero. For a non-smooth composite j, `f[j]` contains the ($\lceil \sqrt{N} \rceil$-smooth) value $-\lceil t/p \rceil$. Thus, the repeated division strategy can be started at `-f[j]`.