

Fair Sums on Loaded Dice: Determination Methods and Combination Generation

C. Jiang, M. Radel, S. Seshadri, D. Sy, and A. Verma
Supervised by Professor W. Gasarch

September 2026

Contents

1	Introduction	2
2	Setup and the Gasarch–Kruskal Theorem	2
2.1	Notation	2
2.2	The Gasarch–Kruskal Theorem	3
3	Determinations	5
3.1	Cyclotomic Polynomials	5
3.2	Proof	6
4	Algorithms	7
4.1	Algorithm A: backtracking over semifair supports	8
4.2	Algorithm B: distributing the cyclotomic factors	10
4.3	Comparison	12
4.4	Generating profiles, and empirical verification	12
5	Construction of Dice Combinations	14
5.1	The Mod Cases	14
5.2	Range Addition and Infinite Families	15
5.3	Generalizing to More Dice	15
5.4	The Merge Theorem	16

1 Introduction

A standard die gives each of its faces the same probability of being rolled. However, when two standard dice are rolled and their values are added, the resulting sum is not uniform. For example, when two six-sided dice are rolled, sums near the middle occur more often than sums near the endpoints. This leads to a natural question: if we are allowed to load the dice by assigning probabilities to their faces, can we make their sum uniform?

We study this problem primarily for pairs of dice with different numbers of faces. Given two dice with d_1 and d_2 faces, we ask whether there is a way to assign probabilities to their faces so that every possible sum occurs with the same probability. In order to do so, some sides will have to have probability 0. Although assigning probability 0 to a face is not physically realizable, the Gasarch-Kruskal theorem implies that a fair loading would otherwise be impossible. We label the faces of an n -sided die by $0, 1, \dots, n-1$. Since two dice with d_1 and d_2 faces can produce $d_1 + d_2 - 1$ different sums, a fair loading must make each of these sums occur with probability $1/(d_1 + d_2 - 1)$.

A central starting point for this problem is the Gasarch-Kruskal theorem, which gives conditions under which dice can be loaded to produce a uniform sum. We use polynomial representations of the dice loadings to study these conditions. In particular, the product of the probability polynomials of the dice encodes the probabilities of all possible sums. This allows the problem to be connected to the factorization of certain cyclotomic polynomials.

We then develop several algorithms for determining whether a given collection of dice can be fairly loaded. The first algorithm searches directly through possible supports of the dice, while another uses the cyclotomic factorization of the target polynomial. We compare these approaches computationally and use them to generate data on which dice combinations can be fairly loaded and what their possible loadings look like.

The computational results also reveal several patterns in the two-die case. One particularly simple family is $(n, n+1)$, which can always be fairly loaded by making the smaller die uniform and using only the two endpoint faces of the larger die. This extends to the more general family $(n, n(x-1)+1)$. We investigate how these cases relate to other fair pairs and whether all fair two-die combinations can be generated from the basic cases through a small number of construction operations.

Finally, we extend these ideas beyond pairs of dice. We give a construction that produces fair loadings for arbitrarily many dice by placing the active faces of each new die at regular intervals determined by the range already produced. We also show how an existing fair collection of dice can be treated as a single uniform block when adding another die, and establish a Merge Theorem that allows several dice in a fair collection to be combined into one. Together, these results show that the problem of fair sums has a rich structure connecting computation, polynomial factorization, and constructive methods for building new fair dice combinations.

2 Setup and the Gasarch–Kruskal Theorem

2.1 Notation

A *die of order* n , for an integer $n \geq 2$, has its faces labeled $0, 1, \dots, n-1$. A *loading* of such a die is a probability vector $\mathbf{p} = (p_0, \dots, p_{n-1})$, with $p_j \geq 0$ for every j and $\sum_{j=0}^{n-1} p_j = 1$; we write X for the corresponding random variable, so that $X = j$ with probability p_j . It is convenient

to record a loading as a *probability polynomial*

$$P(x) = \sum_{j=0}^{n-1} p_j x^j,$$

so that $P(1) = 1$. This bookkeeping device is useful for the following reason: if X and Y are independent dice with probability polynomials P and Q , then $X + Y$ has probability polynomial PQ , since the coefficient of x^k in PQ is $\sum_{i+j=k} p_i q_j = \Pr[X + Y = k]$. Adding dice therefore corresponds to multiplying polynomials.

A *sack* of m dice is a choice of orders $n_1, \dots, n_m \geq 2$ together with an independent loading of each die; we call $\mathbf{n} = (n_1, \dots, n_m)$ the *profile* of the sack. Writing X_i for the outcome of die i and $S = X_1 + \dots + X_m$ for the total, S takes its values in $\{0, 1, \dots, t-1\}$, where

$$t := 1 + \sum_{i=1}^m (n_i - 1)$$

is the *number of totals* of the profile. We also record the *weight* of the profile,

$$|\mathbf{n}| := \sum_{i=1}^m n_i = t + m - 1,$$

and write $\Psi_t(x) := 1 + x + \dots + x^{t-1}$ for the probability polynomial of the uniform distribution on $\{0, \dots, t-1\}$.

Definition 2.1 (Fair sack). A sack is *fair* if S is uniformly distributed on $\{0, 1, \dots, t-1\}$; equivalently, writing P_i for the probability polynomial of die i ,

$$P_1(x)P_2(x) \cdots P_m(x) = \frac{1}{t} \Psi_t(x).$$

The *support* of a loading is the set $A = \{j : p_j > 0\}$ of faces with positive probability. A die is *semifair* (or *nice*) if it is uniform on its support, i.e. $p_j = 1/|A|$ for every $j \in A$, and if A is *symmetric*: $j \in A$ if and only if $n-1-j \in A$. Alternatively, semifair means that for an n -sided die, there exists some $p > 0$ such that for each face i , either $p_i = p$ or $p_i = 0$. For a semifair loading with support A we write

$$G(x) := \sum_{j \in A} x^j \in \{0, 1\}[x]$$

for its *support polynomial*, so that $P(x) = G(x)/G(1)$ and $G(1) = |A|$.

2.2 The Gasarch–Kruskal Theorem

The following fact requires no hypothesis beyond fairness itself, and is proved directly from Definition 2.1.

Lemma 2.2 (Endpoint Lemma). *In a fair sack, $p_{i,0} > 0$ and $p_{i,n_i-1} > 0$ for every die i .*

Proof. The minimal total $S = 0$ occurs only when every die shows its lowest face, so by independence

$$0 < \Pr[S = 0] = \Pr[X_1 = 0] \cdots \Pr[X_m = 0] = p_{1,0} \cdots p_{m,0}.$$

A product of nonnegative reals is positive only if every factor is positive (a single zero factor would collapse the product), so $p_{i,0} > 0$ for every i . The identical argument applied to the maximal total $S = t - 1$, which occurs only when every die shows its highest face, gives $p_{i,n_i-1} > 0$ for every i . $\square$

A considerably deeper fact — that fairness forces *every* positive-probability face to carry equal probability, and not merely the two extreme faces isolated by Lemma 2.2 — is due to Gasarch and Kruskal. We take it here as a given, foundational input, exactly as it is used throughout the rest of this paper.

Theorem 2.3 (Gasarch–Kruskal). *Every die of a fair sack is semifair.*

Theorem 2.3, together with Lemma 2.2, reduces the search for fair sacks of a given profile to a search over symmetric supports containing both endpoints, and pins down exactly the scalar relating each support polynomial to Ψ_t .

Proposition 2.4 (Exact multiplicity). *Let $\mathbf{n} = (n_1, \dots, n_m)$ be the profile of a fair sack, with supports $A_1, \dots, A_m$ and support (probability) polynomials $G_1, \dots, G_m$. Then*

$$G_1(x)G_2(x) \cdots G_m(x) = \Psi_t(x)$$

exactly, and not merely up to a scalar multiple.

Proof. Since $P_i = G_i/G_i(1)$ for every i by Theorem 2.3, multiplying the identity of Definition 2.1 through by $\prod_i G_i(1)$ gives

$$G_1(x) \cdots G_m(x) = \frac{G_1(1) \cdots G_m(1)}{t} \Psi_t(x).$$

By Lemma 2.2, $0 \in A_i$ for every i , so $G_i(0) = 1$ for every i ; and the only way to write the exponent 0 as a sum of m nonnegative integers is as $0 + \cdots + 0$, so the constant term of $G_1(x) \cdots G_m(x)$ is $1 \times \cdots \times 1 = 1$. Comparing constant terms on both sides of the displayed identity forces $G_1(1) \cdots G_m(1) = t$, and substituting back removes the scalar entirely. $\square$

Proposition 2.4 justifies working directly with the $\{0, 1\}$ -coefficient polynomials G_i , which is exactly the object of study already used in Section 4.

Definition 2.5. Given a profile $\mathbf{n}$, let $\mathcal{F}(\mathbf{n})$ denote the set of m -tuples $(G_1, \dots, G_m)$ of polynomials with $G_i \in \{0, 1\}[x]$, $\deg G_i = n_i - 1$, $G_i(0) = 1$, and

$$\prod_{i=1}^m G_i(x) = \Psi_t(x).$$

Each such tuple determines a fair sack, assigning die i the uniform distribution on $\text{supp } G_i$.

Example 2.6. Take $\mathbf{n} = (2, 3)$, so $t = 1 + 1 + 2 = 4$ and $\Psi_4(x) = 1 + x + x^2 + x^3$. Taking $A_1 = \{0, 1\}$ and $A_2 = \{0, 2\}$ gives support polynomials $G_1(x) = 1 + x$ and $G_2(x) = 1 + x^2$, and indeed

$$G_1(x)G_2(x) = (1 + x)(1 + x^2) = 1 + x + x^2 + x^3 = \Psi_4(x).$$

Loading the first die uniformly on both faces and the second die uniformly on its two even faces therefore gives a fair sack: the four equally likely outcomes $(0, 0)$, $(0, 2)$, $(1, 0)$, $(1, 2)$ give totals 0, 2, 1, 3, one apiece.

3 Determinations

By Proposition 2.4, deciding whether a profile $\mathbf{n}$ admits a fair sack is the same as deciding whether the polynomial $\Psi_t(x) = 1 + x + \cdots + x^{t-1}$ factors as $G_1 \cdots G_m$ with each $G_i \in \{0, 1\}[x]$ of degree $n_i - 1$. In this section we use the factorization of Ψ_t into cyclotomic polynomials to show that such a factorization is a purely arithmetic object: a way of distributing the divisors of t among the dice. We call the resulting description of $\mathcal{F}(\mathbf{n})$ the *determination* of the profile.

3.1 Cyclotomic Polynomials

For an integer $d \geq 1$, the d -th cyclotomic polynomial is

$$\Phi_d(x) = \prod_{\substack{1 \leq k \leq d \\ \gcd(k, d) = 1}} (x - e^{2\pi i k/d}),$$

the monic polynomial whose roots are exactly the primitive d -th roots of unity. Its degree is $\varphi(d)$, where φ is Euler's totient function. We use the following four classical facts.

Lemma 3.1 (Standard facts). (a) $x^t - 1 = \prod_{d|t} \Phi_d(x)$ for every $t \geq 1$.

(b) Each Φ_d has integer coefficients and is irreducible over $\mathbb{Q}$.

(c) $\Phi_d(0) = 1$ for every $d \geq 2$.

(d) If p is prime, then $\Phi_{p^j}(x) = \sum_{s=0}^{p-1} x^{sp^{j-1}}$ for $j \geq 1$.

Proof. (a) Every t -th root of unity has a unique exact order $d \mid t$, and it is a primitive d -th root of unity. Since $x^t - 1$ has t distinct roots, the product over $d \mid t$ of the polynomials Φ_d is exactly $x^t - 1$. (b) is the classical irreducibility theorem for cyclotomic polynomials. (c) By (a) applied to $x^d - 1$, dividing by $x - 1 = \Phi_1$ gives $\prod_{e|d, e>1} \Phi_e = \Psi_d$, and evaluating at 0 gives $\prod_{e|d, e>1} \Phi_e(0) = 1$. Induction on d , using $\Phi_e(0) = 1$ for $1 < e < d$, gives $\Phi_d(0) = 1$. (The base case $\Phi_2 = 1 + x$ is immediate.) (d) The divisors of p^j that are not divisors of p^{j-1} are just p^j itself, so by (a) $\Phi_{p^j}(x) = \frac{x^{p^j} - 1}{x^{p^{j-1}} - 1} = 1 + x^{p^{j-1}} + \cdots + x^{(p-1)p^{j-1}}$. $\square$

Dividing (a) by $\Phi_1(x) = x - 1$ gives the factorization of Ψ_t that drives everything below:

$$\Psi_t(x) = \prod_{d|t, d>1} \Phi_d(x), \quad \sum_{d|t, d>1} \varphi(d) = t - 1. \quad (1)$$

Write $D(t) = \{d : d \mid t, d > 1\}$, so that $|D(t)| = \tau(t) - 1$. The factors in (1) are pairwise distinct irreducibles, so Ψ_t is squarefree and its factorization over $\mathbb{Q}$ is unique up to scalars.

Example 3.2. For $t = 12$ we have $D(12) = \{2, 3, 4, 6, 12\}$ and

$$\Psi_{12} = \Phi_2 \Phi_3 \Phi_4 \Phi_6 \Phi_{12},$$

with $\Phi_2 = 1 + x$, $\Phi_3 = 1 + x + x^2$, $\Phi_4 = 1 + x^2$, $\Phi_6 = 1 - x + x^2$ and $\Phi_{12} = 1 - x^2 + x^4$, of degrees 1, 2, 2, 2, 4, which sum to $11 = t - 1$.

3.2 Proof

The main result of this section says that a fair sack is nothing more than an ordered partition of $D(t)$ satisfying two local conditions.

Theorem 3.3 (Determination Theorem). *Let $\mathbf{n} = (n_1, \dots, n_m)$ be a profile with $t = 1 + \sum_i (n_i - 1)$. The assignment*

$$(G_1, \dots, G_m) \mapsto (B_1, \dots, B_m), \quad B_i = \{d \in D(t) : \Phi_d \mid G_i\},$$

is a bijection from $\mathcal{F}(\mathbf{n})$ onto the set of ordered partitions $D(t) = B_1 \sqcup \dots \sqcup B_m$ such that for every i

$$\sum_{d \in B_i} \varphi(d) = n_i - 1 \quad \text{and} \quad \prod_{d \in B_i} \Phi_d \in \{0, 1\}[x]. \quad (2)$$

The inverse is $B_i \mapsto G_i = \prod_{d \in B_i} \Phi_d$.

Proof. The map is well defined. Let $(G_1, \dots, G_m) \in \mathcal{F}(\mathbf{n})$. Each $G_i \in \{0, 1\}[x]$ has degree $n_i - 1$, so it is monic, and $\prod_i G_i = \Psi_t$ by Definition 2.5. Hence each G_i divides Ψ_t in $\mathbb{Q}[x]$. By (1) and Lemma 3.1(b), Ψ_t is a product of pairwise distinct irreducibles, so unique factorization in $\mathbb{Q}[x]$ shows that $G_i = c_i \prod_{d \in B_i} \Phi_d$ for a unique subset $B_i \subseteq D(t)$ and a scalar $c_i \in \mathbb{Q}^\times$. Both sides are monic, so $c_i = 1$. Comparing $\prod_i G_i = \Psi_t$ with (1), and using that each Φ_d occurs to the first power on the right, shows that every $d \in D(t)$ lies in exactly one B_i . So the B_i partition $D(t)$. Taking degrees gives $\sum_{d \in B_i} \varphi(d) = n_i - 1$, and $G_i \in \{0, 1\}[x]$ is the second condition of (2).

The map is injective, since $G_i = \prod_{d \in B_i} \Phi_d$ is recovered from B_i .

The map is surjective. Given a partition satisfying (2), put $G_i := \prod_{d \in B_i} \Phi_d$. Then $\prod_i G_i = \prod_{d \in D(t)} \Phi_d = \Psi_t$ by (1); $\deg G_i = \sum_{d \in B_i} \varphi(d) = n_i - 1$; $G_i \in \{0, 1\}[x]$ by hypothesis; and $G_i(0) = \prod_{d \in B_i} \Phi_d(0) = 1$ by Lemma 3.1(c). Thus $(G_1, \dots, G_m) \in \mathcal{F}(\mathbf{n})$, and it maps to $(B_1, \dots, B_m)$. $\square$

Remark 3.4. Theorem 3.3 is the theoretical content behind Proposition 4.5 and Algorithm B. The first condition of (2) is a subset-sum condition on the values $\varphi(d)$. The second is the only place where the arithmetic of t alone does not decide the answer. It is a genuine constraint on the coefficients, since a product of cyclotomic polynomials generally has negative coefficients.

Example 3.5. Take $\mathbf{n} = (3, 10)$, so $t = 12$. The partitions of $D(12)$ with block degree sums 2 and 9 are $\{4\} \mid \{2, 3, 6, 12\}$ and $\{3\} \mid \{2, 4, 6, 12\}$, since $\varphi(4) = \varphi(3) = 2$ and the remaining values $1 + 2 + 2 + 4 = 9$. Both satisfy the 0/1 condition:

$$\Phi_4 = 1 + x^2, \quad \frac{\Psi_{12}}{\Phi_4} = 1 + x + x^4 + x^5 + x^8 + x^9,$$

$$\Phi_3 = 1 + x + x^2, \quad \frac{\Psi_{12}}{\Phi_3} = 1 + x^3 + x^6 + x^9.$$

So $|\mathcal{F}(3, 10)| = 2$. The second solution is the mod case $(3, 3 \cdot 3 + 1)$; the first is the sumset $\{0, 1, 4, 5, 8, 9\}$. In contrast, for $\mathbf{n} = (3, 4)$ with $t = 6$ the partition $\{6\} \mid \{2, 3\}$ is rejected, because $\Phi_6 = 1 - x + x^2 \notin \{0, 1\}[x]$. Only $\{3\} \mid \{2, 6\}$ survives, giving $(1 + x + x^2)(1 + x^3) = \Psi_6$.

The theorem has several immediate consequences.

Corollary 3.6 (Divisor bound). *If $\mathcal{F}(\mathbf{n}) \neq \emptyset$ then $m \leq \tau(t) - 1$. In particular, if t is prime then $\mathcal{F}(\mathbf{n}) = \emptyset$ for every profile with $m \geq 2$.*

Proof. By Theorem 3.3, $D(t)$ is partitioned into m blocks, each nonempty because $\deg G_i = n_i - 1 \geq 1$. Hence $m \leq |D(t)| = \tau(t) - 1$. If t is prime then $\tau(t) = 2$, so $m \leq 1$. $\square$

Corollary 3.7 (Parity). *If $\mathcal{F}(\mathbf{n}) \neq \emptyset$, then at most one n_i is even. Moreover, if t is odd then every n_i is odd.*

Proof. For $d > 2$, $\varphi(d)$ is even (nonreal roots of unity come in conjugate pairs), while $\varphi(2) = 1$. Hence $n_i - 1 = \sum_{d \in B_i} \varphi(d)$ is even unless $2 \in B_i$, in which case it is odd. The divisor 2 belongs to $D(t)$ only for even t , and then lies in exactly one block. So at most one $n_i - 1$ is odd, that is, at most one n_i is even, and none is even when t is odd. $\square$

This gives part (ii) of the filter in Lemma 4.2, and Corollary 3.6 strengthens the bound $m \leq \log_2 t$ recorded there whenever $\tau(t) - 1 < \log_2 t$.

When t is a prime power the 0/1 condition disappears, and the determination can be counted exactly.

Theorem 3.8 (Prime-power case). *Let $t = p^e$ with p prime and $e \geq 1$. Every partition $\{1, \dots, e\} = S_1 \sqcup \dots \sqcup S_m$ into nonempty blocks defines a fair sack with*

$$G_i = \prod_{j \in S_i} \Phi_{p^j}, \quad n_i = 1 + \sum_{j \in S_i} \varphi(p^j),$$

and every fair sack with $t = p^e$ arises this way from a unique partition. Consequently the number of fair sacks with exactly m dice, up to reordering, is the Stirling number $S(e, m)$, and the total over all m is the Bell number $B(e)$.

Proof. Here $D(p^e) = \{p, p^2, \dots, p^e\}$, so blocks $B_i \subseteq D(t)$ correspond to subsets $S_i \subseteq \{1, \dots, e\}$ via $d = p^j$. By Theorem 3.3 it suffices to show that the 0/1 condition holds automatically. By Lemma 3.1(d),

$$\prod_{j \in S} \Phi_{p^j}(x) = \sum_{(s_j)_{j \in S} \in \{0, \dots, p-1\}^S} x^{\sum_{j \in S} s_j p^{j-1}}.$$

The exponents are integers whose base- p digits are supported on the positions $j - 1$ for $j \in S$. By uniqueness of base- p representation, distinct digit tuples give distinct exponents, so every coefficient is 0 or 1. The correspondence with partitions is then a bijection by Theorem 3.3, and counting unordered partitions into m blocks gives $S(e, m)$ by definition. $\square$

Example 3.9. For $t = 8 = 2^3$, the partition $\{1\} \mid \{2, 3\}$ gives $G_1 = \Phi_2 = 1 + x$ and $G_2 = \Phi_4 \Phi_8 = (1 + x^2)(1 + x^4) = 1 + x^2 + x^4 + x^6$, so $\mathbf{n} = (2, 7)$. There are $S(3, 1) + S(3, 2) + S(3, 3) = 1 + 3 + 1 = 5 = B(3)$ fair sacks with $t = 8$ in total.

4 Algorithms

We describe two algorithms which, given a size-profile $\mathbf{n}$, return the complete list $\mathcal{F}(\mathbf{n})$ of fair sacks with that profile, together with a third, elementary routine that generates every profile up to a prescribed range and thereby turns the solvers into an exhaustive experiment.

The two solvers differ in exactly one respect: how much of the theory they are told. Algorithm A knows only that the dice of a fair sack are semifair, and searches the resulting local configuration space by backtracking. Algorithm B knows in addition that the product of the die

polynomials is Ψ_t , hence that each die polynomial is a product of cyclotomic factors of Ψ_t , and searches the (much smaller) space of distributions of those factors among the dice.

Throughout, a *size-profile* (or simply *profile*) is a finite sequence $\mathbf{n} = (n_1, \dots, n_m)$ of integers $n_i \geq 2$, the orders of the m dice. we write

$$t = 1 + \sum_{i=1}^m (n_i - 1), \quad |n| = \sum_{i=1}^m n_i = t + m - 1,$$

for the number of totals and the *weight* of the profile. We write $\Psi_t(x) = 1 + x + \dots + x^{t-1}$, Φ_d for the d -th cyclotomic polynomial, φ and τ for the Euler and divisor functions, $\Omega(t)$ for the number of prime factors of t counted with multiplicity, and $H(t)$ for the number of ordered factorizations of t into factors exceeding 1 (so $H(1) = 1$ and $H(t) = \sum_{d|t, d < t} H(d)$).

Two normalizations reduce the problem to a purely combinatorial one. First, a die of order 1 contributes the constant 1 to the product of die polynomials and may be deleted from, or adjoined to, any sack without affecting fairness; every implementation therefore strips such dice on entry and reinstates them on exit, and we assume $n_i \geq 2$ from now on. Second, by the Gasarch–Kruskal theorem every die of a fair sack is semi-fair, so its probability vector is the uniform vector on its support $A_i \subseteq \{0, 1, \dots, n_i - 1\}$; rescaling by $|A_i|$ replaces the die polynomial by the monic 0/1 *support polynomial* $G_i(x) = \sum_{j \in A_i} x^j$. The problem to be solved is therefore the following.

Definition 4.1. Given a profile $\mathbf{n}$, let $\mathcal{F}(\mathbf{n})$ denote the set of m -tuples $(G_1, \dots, G_m)$ of polynomials with $G_i \in \{0, 1\}[x]$, $\deg G_i = n_i - 1$, $G_i(0) = 1$, and

$$\prod_{i=1}^m G_i(x) = \Psi_t(x). \quad (3)$$

The two algorithms below all compute $\mathcal{F}(\mathbf{n})$; the corresponding sack assigns to die i the uniform distribution on $\text{supp } G_i$.

Both begin with the same constant-time filter, which we record for reference.

Lemma 4.2. *Let $\mathbf{n}$ be a profile with $\mathcal{F}(\mathbf{n}) \neq \emptyset$. Then*

- (i) *the n_i are pairwise distinct;*
- (ii) *at most one n_i is even;*
- (iii) *$n_m \geq \frac{t}{2} + 1$;*
- (iv) *$\prod_i |\text{supp } G_i| = t$ for every $(G_1, \dots, G_m) \in \mathcal{F}(\mathbf{n})$; in particular $2^m \leq t$, i.e. $m \leq \log_2 t$.*

4.1 Algorithm A: backtracking over semifair supports

Overview

A semifair die of order n is determined by a palindromic 0/1 vector whose first and last entries are 1; equivalently, by an arbitrary subset S of the $\lceil n/2 \rceil - 1$ symmetry classes $\{j, n - 1 - j\}$ with $1 \leq j \leq \lceil n/2 \rceil - 1$, the class $\{0, n - 1\}$ being forced. There are thus exactly $2^{\lceil n/2 \rceil - 1}$ semifair dice of order n (Algorithm 1), and Algorithm A simply searches the product of these sets for tuples

satisfying (3), assigning the dice in increasing order of n_i and maintaining the partial product $P_i = G_1 \cdots G_i$.

The search is made viable by one prune, which exploits the fact that the target Ψ_t is the *smallest* polynomial that a partial product can grow into.

Lemma 4.3 (Monotonicity). *Let $P \in \mathbb{Z}_{\geq 0}[x]$ and let $G \in \mathbb{Z}_{\geq 0}[x]$ with $G(0) = 1$. Then $PG - P \in \mathbb{Z}_{\geq 0}[x]$; that is, the coefficients of the product dominate those of P coefficientwise. Consequently, if some coefficient of a partial product P_i exceeds 1, no completion of the branch satisfies (3).*

Proof. $PG - P = P(G - 1)$ and $G - 1 \in \mathbb{Z}_{\geq 0}[x]$ because $G(0) = 1$ and G has nonnegative coefficients. All coefficients of Ψ_t equal 1, so a coefficient that has already exceeded 1 can never be reduced. $\square$

Correctness of Algorithm 2 is then immediate: the recursion enumerates every tuple of semifair supports of the prescribed orders except those killed by Lemma 4.3, and tests (3) at each leaf; by the Gasarch–Kruskal theorem every element of $\mathcal{F}(\mathbf{n})$ is such a tuple, so none is missed.

Proposition 4.4. *Assume $\mathbf{n}$ passes the filter of Lemma 4.2, and let F_i denote the number of feasible prefixes of length i , that is, the number of tuples $(G_1, \dots, G_i)$ of semifair supports of orders $n_1, \dots, n_i$ whose product has all coefficients at most 1 (so $F_0 = 1$). Then:*

- (a) *the unpruned search tree has $\prod_{i=1}^m 2^{\lceil n_i/2 \rceil - 1} = 2^{\lfloor (t-1)/2 \rfloor}$ leaves;*
- (b) *the pruned search visits $\sum_{i=1}^m F_{i-1} 2^{\lceil n_i/2 \rceil - 1}$ nodes and performs $O(t n_i)$ coefficient operations at each node of depth i ;*
- (c) *if $\mathcal{F}(\mathbf{n}) \neq \emptyset$, the search visits at least $2^{\lceil n_m/2 \rceil - 1} \geq 2^{t/4 - 1/2}$ nodes.*

Proof. (a) By Lemma 4.2(ii) at most one $n_i - 1$ is odd, so $\sum_i (\lceil n_i/2 \rceil - 1) = \sum_i \lfloor (n_i - 1)/2 \rfloor = \lfloor \frac{1}{2} \sum_i (n_i - 1) \rfloor = \lfloor (t - 1)/2 \rfloor$. (b) Each node of depth i arises from a feasible prefix of length $i - 1$ and a choice of G_i ; the work at the node is one convolution of a polynomial of degree $< t$ by a polynomial with at most n_i terms, followed by a scan of the result. (c) If $\mathcal{F}(\mathbf{n}) \neq \emptyset$, fix a solution; each of its prefixes is feasible, so $F_i \geq 1$ for all i , and in particular the whole option set of the last (largest) die is enumerated. By Lemma 4.2(iii), $n_m \geq t/2 + 1$, whence $\lceil n_m/2 \rceil - 1 \geq t/4 - 1/2$. $\square$

Algorithm 1 SEMIFAIRSUPPORTS(n): the $2^{\lceil n/2 \rceil - 1}$ semifair support polynomials of order n .

Input: an integer $n \geq 2$

Output: all monic palindromic $G \in \{0, 1\}[x]$ of degree $n - 1$ with $G(0) = 1$

- 1: $h \leftarrow \lceil n/2 \rceil - 1$
 - 2: **for all** $S \subseteq \{1, 2, \dots, h\}$ **do**
 - 3: **emit** $G_S(x) = 1 + x^{n-1} + \sum_{j \in S} (x^j + x^{n-1-j})$
 - 4: **end for**
-

Algorithm 2 Algorithm A: backtracking over semifair supports.

Input: a profile $\mathbf{n} = (n_1 < n_2 < \dots < n_m)$

Output: $\mathcal{F}(\mathbf{n})$

```

1: procedure FAIRSACKSA( $\mathbf{n}$ )
2:    $t \leftarrow 1 + \sum_i (n_i - 1)$ ;  $\mathcal{L} \leftarrow \emptyset$ 
3:   if  $\mathbf{n}$  fails Lemma 4.2(i)–(iii) then
4:     return  $\emptyset$ 
5:   end if
6:   for  $i \leftarrow 1$  to  $m$  do
7:      $\mathcal{N}_i \leftarrow \text{SEMIFAIRSUPPORTS}(n_i)$ 
8:   end for
9:    $\text{EXTEND}(1, 1, ( ))$ 
10:  return  $\mathcal{L}$ 
11: end procedure

12: procedure EXTEND( $i, P, (G_1, \dots, G_{i-1})$ )
13:   if  $i = m + 1$  then
14:     if  $P = \Psi_t$  then
15:        $\mathcal{L} \leftarrow \mathcal{L} \cup \{(G_1, \dots, G_m)\}$ 
16:     end if
17:     return
18:   end if
19:   for all  $G \in \mathcal{N}_i$  do
20:      $Q \leftarrow P \cdot G$ 
21:     if every coefficient of  $Q$  is  $\leq 1$  then ▷ Lemma 4.3
22:        $\text{EXTEND}(i + 1, Q, (G_1, \dots, G_{i-1}, G))$ 
23:     end if
24:   end for
25: end procedure

```

4.2 Algorithm B: distributing the cyclotomic factors

Overview

The polynomial Ψ_t has simple roots, namely the t -th roots of unity other than 1, and factors over $\mathbb{Z}$ as

$$\Psi_t(x) = \prod_{d|t, d>1} \Phi_d(x), \quad \sum_{d|t, d>1} \varphi(d) = t - 1. \quad (4)$$

Write $D(t) = \{d : d | t, d > 1\}$ and $r = |D(t)| = \tau(t) - 1$. Because $\mathbb{Z}[x]$ is a unique factorization domain and the factorization (4) is squarefree, any solution of (3) is obtained by distributing the r irreducible factors among the m dice. This is the content of the next proposition, which converts the problem into a constrained set-partition problem and eliminates the exponential-in- t enumeration of Algorithm A entirely.

Proposition 4.5. *The assignment $(G_1, \dots, G_m) \mapsto (B_1, \dots, B_m)$, where $B_i = \{d \in D(t) : \Phi_d | G_i\}$, is a bijection from $\mathcal{F}(\mathbf{n})$ onto the set of ordered partitions $D(t) = B_1 \sqcup \dots \sqcup B_m$ such that,*

for every i ,

$$\sum_{d \in B_i} \varphi(d) = n_i - 1 \quad \text{and} \quad \prod_{d \in B_i} \Phi_d \in \{0, 1\}[x]. \quad (5)$$

Proof. Let $(G_1, \dots, G_m) \in \mathcal{F}(\mathbf{n})$. The G_i have positive integer leading coefficients whose product is 1, so each G_i is monic and divides Ψ_t in $\mathbb{Z}[x]$; since Ψ_t is squarefree with the irreducible factors listed in (4), unique factorization yields $G_i = \prod_{d \in B_i} \Phi_d$ for a unique $B_i \subseteq D(t)$, and comparing (3) with (4) shows that the B_i partition $D(t)$. Conditions (5) hold because $\deg G_i = n_i - 1$ and $G_i \in \{0, 1\}[x]$. Conversely, given such a partition, put $G_i = \prod_{d \in B_i} \Phi_d$. Then $\prod_i G_i = \Psi_t$ by (4), $\deg G_i = n_i - 1$, and $G_i(0) = \prod_{d \in B_i} \Phi_d(0) = 1$ since $\Phi_d(0) = 1$ for $d > 1$; with the second condition of (5) this is exactly the requirement of Definition 4.1. The two assignments are mutually inverse. $\square$

Algorithm 3 performs the search suggested by Proposition 4.5: it assigns blocks to the dice one at a time, in increasing order of the target degree $n_i - 1$, so that the most constrained die is placed first. Two prunes are used. The first is the 0/1 test of (5), applied to each candidate block as soon as it is formed; by Lemma 4.3 a block whose product has a coefficient exceeding 1 can never occur in a solution. The second is a subset-sum reachability test: before descending, one checks that each still-unplaced target degree $n_j - 1$ is realisable as a sum of φ -values of still-unassigned divisors. Both the set of reachable sums and its update are conveniently held in a t -bit mask, computed by $\text{reach} \mid= \text{reach} \ll \varphi(d)$ over the unassigned d .

Proposition 4.6. *Let $r = \tau(t) - 1$. Algorithm 3 visits at most $(m + 1)^r$ nodes; at each node it enumerates the admissible blocks in $O(r \cdot (1 + \beta))$ time, where β is the number of blocks found, and spends $O(t|B|)$ integer operations on forming and testing each candidate block B . Precomputation of $\{\Phi_d\}_{d \in D(t)}$ costs $O(\tau(t)t^2)$ operations. The total is*

$$O\left((m + 1)^{\tau(t)} \cdot t^2\right) = \exp(t^{o(1)}),$$

since $\tau(t) = t^{O(1/\log \log t)}$ and $m \leq \log_2 t$.

Proof. A node is a partial assignment of the divisors, i.e. a map $D(t) \rightarrow \{1, \dots, m\} \cup \{*\}$, whence the bound $(m + 1)^r$. The block enumeration is the standard subset-sum backtracking with suffix-sum bounds, whose running time is proportional to the number of nodes of its own tree, each of which either leads to a solution or is cut by the bound; forming $\prod_{d \in B} \Phi_d$ takes $|B|$ convolutions of polynomials of degree $< t$. For the cyclotomic polynomials one uses $\Phi_n = (x^n - 1) / \prod_{d|n, d < n} \Phi_d$, a division of polynomials of degree $\leq t$. The stated asymptotics follow from bounds on τ . $\square$

The exponent has been reduced from $\Theta(t)$ in Algorithm A to $\tau(t) = t^{o(1)}$, which is a genuine gain — but it is still an exponent, and it is large precisely for the arithmetically interesting t . The following example is representative.

Remark 4.7 (Memoisation). The recursion $\Phi_n = (x^n - 1) / \prod_{d|n, d < n} \Phi_d$ must be memoised. If $C(n)$ denotes the number of calls made by the unmemoised recursion, then $C(1) = 1$ and $C(n) = 1 + \sum_{d|n, d < n} C(d)$, from which one gets $C(n) = 2H(n)$ for $n > 1$ by induction. Thus the naive precomputation costs $\Theta(H(t))$ polynomial divisions.

Algorithm 3 Algorithm B: distributing the cyclotomic factors of Ψ_t .

Input: a profile $\mathbf{n}$, indexed so that $n_1 - 1 \leq \dots \leq n_m - 1$

Output: $\mathcal{F}(\mathbf{n})$

```

1: procedure FAIRSACKSB( $\mathbf{n}$ )
2:    $t \leftarrow 1 + \sum_i (n_i - 1)$ ;  $\mathcal{L} \leftarrow \emptyset$ 
3:   if  $\mathbf{n}$  fails Lemma 4.2(i)–(iii) then
4:     return  $\emptyset$ 
5:   end if
6:    $D \leftarrow \{d : d \mid t, d > 1\}$ ; compute  $\varphi(d)$  and  $\Phi_d$  for  $d \in D$ 
7:   ASSIGN(1,  $D$ )
8:   return  $\mathcal{L}$ 
9: end procedure

10: procedure ASSIGN( $i, R$ )  $\triangleright R = \text{divisors not yet assigned}$ 
11:   if  $i = m + 1$  then
12:      $\mathcal{L} \leftarrow \mathcal{L} \cup \{(G_1, \dots, G_m)\}$ ; return
13:   end if
14:    $\Sigma \leftarrow \{\sum_{d \in B'} \varphi(d) : B' \subseteq R\}$   $\triangleright \text{bitmask of reachable subset sums}$ 
15:   for  $j \leftarrow i$  to  $m$  do
16:     if  $n_j - 1 \notin \Sigma$  then
17:       return
18:     end if
19:   end for
20:   for all  $B \subseteq R$  with  $\sum_{d \in B} \varphi(d) = n_i - 1$  do
21:      $G \leftarrow \prod_{d \in B} \Phi_d$ 
22:     if  $G \in \{0, 1\}[x]$  then
23:        $G_i \leftarrow G$ ; ASSIGN( $i + 1, R \setminus B$ )
24:     end if
25:   end for
26: end procedure

```

4.3 Comparison

The improvement collapses a whole dimension of the search, representing a significant speed-up.

	search space	size	leaf test
A	tuples of semifair supports	$2^{\lfloor (t-1)/2 \rfloor}$	$\prod_i G_i = \Psi_t$
B	ordered partitions of $\{\Phi_d\}_{d \mid t, d > 1}$	$\leq (m+1)^{\tau(t)-1}$	none

4.4 Generating profiles, and empirical verification

A solver settles one profile; to test the theory one wants them all. The following observations make an exhaustive sweep cheap to organise.

Lemma 4.8. *Every fair sack of size $m \geq 1$ whose dice all have order ≥ 2 has a profile that can be written as a strictly increasing sequence $2 \leq n_1 < n_2 < \dots < n_m$; and a fair sack remains*

fair after adjoining or deleting dice of order 1. Consequently the strictly increasing sequences of integers ≥ 2 of weight at most N enumerate, without repetition, all profiles that must be tested in order to determine every fair sack of weight at most N .

Proof. A sack is a set, so its profile is defined up to order; strictness is Lemma 4.2(i); the statement about dice of order 1 was noted in §4. $\square$

Algorithm 4 performs the enumeration by the obvious depth-first recursion: it extends the current tuple by any admissible next order, maintaining strict increase through the lower bound s and the weight bound through the range of the loop. Each tuple is emitted exactly once, and the recursion tree has one node per emitted tuple, so the generator runs in time proportional to the total size of its output. That output has $\sum_{n \leq N} q_2(n)$ elements, where $q_2(n)$ counts the partitions of n into distinct parts, none equal to 1; by the **Hardy–Ramanujan asymptotic** $q(n) \sim \exp(\pi\sqrt{n/3})/(4 \cdot 3^{1/4}n^{3/4})$ **for partitions into distinct parts**, the number of profiles of weight at most N is $\exp(\pi\sqrt{N/3})$ up to a factor polynomial in N . The sweep is therefore subexponential in N , and in the range accessible to Algorithm B its cost, rather than that of the solver, is what limits the experiment.

Algorithm 4 Algorithm C: all size-profiles of weight at most N .

Input: an integer $N \geq 2$

Output: every strictly increasing tuple $(n_1 < \dots < n_m)$ with $n_1 \geq 2$, $m \geq 1$ and $\sum_i n_i \leq N$, each exactly once

```

1: procedure PROFILES( $N$ )
2:   REC(2, ( ), 0)
3: end procedure

4: procedure REC( $s, \mathbf{n}, w$ )                                ▷  $s$  = least admissible next order,  $w = |\mathbf{n}|$ 
5:   if  $\mathbf{n} \neq ( )$  then
6:     emit  $\mathbf{n}$ 
7:   end if
8:   for  $d \leftarrow s$  to  $N - w$  do
9:     REC( $d + 1, \mathbf{n} \frown d, w + d$ )
10:  end for
11: end procedure

```

The verification protocol

Composing Algorithm 4 with the two solvers gives a consistency check, which we strengthen to a genuine verification by adding an oracle that is independent of both. For each profile the outputs of A and B are normalised to sets of supports and compared for equality; then every sack produced is checked *ab initio* by convolving the exact rational probability vectors and testing that the resulting distribution of the total is the uniform distribution on $\{0, 1, \dots, t - 1\}$. The oracle shares no code with the solvers and uses none of the structure theory so agreement is evidence for the implementations.

Table 4.4 records the census produced by the sweep. Two features are worth noting. The proportion of fair profiles decays slowly, and fair profiles are not rare. Multiplicities greater than 1 appear at once — the smallest examples being (3, 10) and (4, 9), both with $t = 12$ — and

multiplicity 3 first occurs at weight 37, for the profiles (9, 28) and (10, 27), both with $t = 36$; the smallest profile of size 3 with three fair sacks is (3, 10, 25), again with $t = 36$.

N	profiles	fair profiles	fair sacks	max. multiplicity
10	23	8	8	1
15	75	20	22	2
20	202	46	48	2
26	578	89	103	2
30	1096	109	123	2
40	4643	235	263	3
50	16757	412	496	3
60	53853	571	663	3
70	158326	903	1021	3

5 Construction of Dice Combinations

5.1 The Mod Cases

We've already established an algorithm (see Sections 4.1–4.3) for testing dice combinations to see if a dice combination is loadable, and if so, how many unique sacks. In this section, we take a generative approach.

For any integer $n \geq 2$, a pair of dice where one die has n sides and the other has $n + 1$ sides can always be fairly loaded, where the smaller die (n sides) has probability $p = 1/n$ on every side, and the larger die ($n + 1$ sides) has probability $q = 0.5$ on its two endpoints only, and probability 0 elsewhere. This pattern holds for all n , where the first die has n sides and the second die has $n + 1$ sides. This is referred to throughout the rest of the paper as the $(n, n + 1)$ base case. This also extends to more than two dice (see Section 5.3).

Theorem 5.1 (Mod Theorem). *Generalizing these cases even further, for any pair of dice d_1 and d_2 with $d_1 < d_2$, if $d_2 \bmod d_1 = 1$, the dice can always be fairly loaded, with the smaller die (d_1) having probability $p = 1/d_1$ on every side, and the larger die (d_2) having probability q on every d_1 -th side, starting at the first side and ending at the last, and probability 0 on all other sides. This family will be referred to throughout the rest of the paper as the mod cases, an extension of the $(n, n + 1)$ base case.*

It can alternatively be thought of as $(n, xn + 1)$, where x is any integer greater than or equal to 1. *Mod addition* is the name given to the process of expanding from the $(n, n + 1)$ base case to the $(n, xn + 1)$ mod case.

Proof: Let

$$Q_1(x) = 1 + x + x^2 + \cdots + x^{d_1-1} = \frac{x^{d_1} - 1}{x - 1}$$

and

$$Q_2(x) = 1 + x^{d_1} + x^{2d_1} + \cdots + x^{(b-1)d_1} = \frac{x^{d_1 b} - 1}{x^{d_1} - 1}.$$

Multiplying, the $(x^{d_1} - 1)$ factors cancel:

$$Q_1(x)Q_2(x) = \frac{x^{d_1} - 1}{x - 1} \cdot \frac{x^{d_1 b} - 1}{x^{d_1} - 1} = \frac{x^{d_1 b} - 1}{x - 1} = 1 + x + x^2 + \cdots + x^{d_1 b - 1},$$

and since $d_1b - 1 = d_1 + d_2 - 2 = r - 1$, this will always create fair sums.

5.2 Range Addition and Infinite Families

Theorem 5.2 (Range Addition Theorem). *Based on observations from our computational data, it is possible to generate every dice combination starting from the $(n, n + 1)$ base case using a combination of mod addition (or, alternatively, starting from one of the mod cases) and range addition. We have verified this statement empirically over all tested cases described in Section 5.3. We do not yet have a complete mathematical proof that it holds for all possible dice combinations.*

Range addition is the process of adding to a loadable dice combination in order to keep it loadable. Given two dice d_1 and d_2 , the range is $r = d_1 + d_2 - 1$. You can add the range to either die infinitely many times, and the result will still be a loadable dice combination.

With each iteration of range addition, the die you expand adds r sides to itself, and up to the $(r - 1)$ -th index, all added sides will have probability 0 of being rolled. Each additional side after that (up to the number of sides originally on the die) will repeat the pattern originally on the die.

For example, starting with the $(n, n + 1)$ base case of $(3, 4)$, loadable as $[p, p, p]$ and $[p, 0, 0, p]$, you can make one of three modifications to the dice (there is a fourth modification as well — see Section 5.3):

- **Option 1:** Use mod addition to add 3 sides to the second die any number of times, with every third side having probability p (i.e. $[p, 0, 0, p, 0, 0, p, \dots, 0, 0, p]$), which keeps $d_2 \bmod d_1$ equal to 1.
- **Option 2:** Use range addition on the first die. Since the range is equal to 6, the first die becomes 9-sided, with sides 3–5 having probability 0,¹ and sides 6–8 repeating the initial pattern, giving the dice combination $[p, p, p, 0, 0, 0, p, p, p]$ and $[p, 0, 0, p]$.
- **Option 3:** Use range addition on the second die, which gives the same result as performing mod addition twice.

Note that you can perform any of these operations infinitely on either die, but mod addition must be performed first. Alternatively, you can think of this as starting with any mod case $(n, xn + 1)$ and simply performing range addition (Options 2 or 3) infinitely on either die.

5.3 Generalizing to More Dice

In Section 5.2, we mentioned a fourth operation that can be performed on any set of loadable dice. Given a set of k loadable dice with range r^2 , the probability of rolling any sum from 0 to $r - 1$ is equal, so that set of dice can be treated as a single r -sided die, which is inherently fair. Since we now have a single die with r sides, we can add another die with $r + 1$ sides, with only sides 0 and r having probability $p = 0.5$, and all other sides having probability $p = 0$.

The application of this is that, given a set of k loadable dice with range r , you can always add another die with $r + 1$ sides. The mod theorem also applies here: you can continue to add

¹For the purposes of this paper, sides are indexed starting at 0.

²for $k \geq 2$, $r = d_1 + d_2 + \dots + d_k - k + 1$.

r sides to this new die infinitely, with every r -th side having probability p and all other sides having probability 0.

Additionally, it turns out that range addition works exactly the same way for three or more dice. Pick any die with d_k sides from a set of loadable dice, and you can always add r sides to that die. All added sides up to the $(r - 1)$ -th side will have probability 0, and the remaining d_k sides will repeat the original pattern of the die.

We were able to confirm this empirically: we coded all four operations and tested them on all dice pairs where $d_1 < 1000$ and $d_2 < 1000$, and on all sets of dice where every die has fewer than 100 sides. Comparing this to data generated independently by Claude, we confirmed that the loadable dice combinations produced by our code matched exactly in both cases, confirming that these operations account for all dice combinations within the tested ranges.

5.4 The Merge Theorem

The constructions of the preceding subsections all move in one direction: they start from a fair sack and grow it, one die at a time. The theorem of this subsection moves in the other direction. It shows that any subset of the dice in a fair sack — not necessarily an initial segment, and not necessarily two dice at a time — may be collapsed into a single die without disturbing fairness. Taking that subset to be the entire sack gives exactly the claim used without proof at the start of Section 5.3: that a set of k loadable dice with range r may be treated as a single, inherently fair, r -sided die.

Theorem 5.3 (Merge Theorem). *Let $(G_1, \dots, G_m) \in \mathcal{F}(\mathbf{n})$ have supports $A_1, \dots, A_m$, and let $T \subseteq \{1, \dots, m\}$ be nonempty. Define*

$$A_T := \left\{ \sum_{i \in T} a_i : a_i \in A_i \text{ for each } i \in T \right\}.$$

Replacing the dice indexed by T with a single die loaded uniformly on A_T yields a fair sack of the remaining $m - |T| + 1$ dice.

Proof. Write $G_T(x) := \prod_{i \in T} G_i(x)$ and $R(x) := \prod_{i \notin T} G_i(x)$, so that $G_T(x)R(x) = \Psi_t(x)$ exactly, by Proposition 2.4. By the definition of polynomial multiplication, the coefficient of x^k in $G_T(x)$ is exactly the number of ways to write $k = \sum_{i \in T} a_i$ with $a_i \in A_i$ for each i ; that is, G_T is precisely the (possibly multiply-counted) support polynomial of A_T .

We first show this count is never more than 1, so that $G_T \in \{0, 1\}[x]$ and A_T is a genuine support. Every G_i has nonnegative coefficients, so G_T and R do as well; and $R(0) = 1$, since $0 \in A_i$ for every $i \notin T$ by Lemma 2.2. If some coefficient of G_T were at least 2 at an exponent k , then the coefficient of x^k in the product $G_T(x)R(x)$ would already be at least $2 \cdot R(0) = 2$, using only the contribution from that one term of R and the nonnegativity of the rest; this contradicts $G_T(x)R(x) = \Psi_t(x)$, all of whose coefficients equal 1. Hence every coefficient of G_T is 0 or 1, and no total in A_T is reached in more than one way.

Next we show A_T is symmetric. Let $n_T - 1 := \sum_{i \in T} (n_i - 1)$ denote the top face of the proposed merged die. If $a_i \in A_i$ for each $i \in T$, then $(n_i - 1) - a_i \in A_i$ as well, since each A_i is symmetric by Theorem 2.3; summing these reflections over T shows that

$$(n_T - 1) - \sum_{i \in T} a_i = \sum_{i \in T} ((n_i - 1) - a_i) \in A_T$$

whenever $\sum_{i \in T} a_i \in A_T$. So A_T is symmetric, the merged die is semifair, and

$$G_T(x) \cdot R(x) = \Psi_t(x)$$

is exactly the fairness condition of Definition 4.1 for the reduced, $(m - |T| + 1)$ -die sack. $\square$

Example 5.4. Take the fair sack of profile $(2, 3, 5)$ with supports $A_1 = \{0, 1\}$, $A_2 = \{0, 2\}$, $A_3 = \{0, 4\}$, so that $G_1(x) = 1 + x$, $G_2(x) = 1 + x^2$, $G_3(x) = 1 + x^4$, and indeed $G_1(x)G_2(x)G_3(x) = \Psi_8(x)$. Merging the first two dice, $T = \{1, 2\}$, gives

$$A_T = \{0, 1\} + \{0, 2\} = \{0, 1, 2, 3\},$$

every sum of one face from each of the two dice being attained exactly once. The reduced sack, of profile $(4, 5)$, is fair: $(1 + x + x^2 + x^3)(1 + x^4) = \Psi_8(x)$.

Taking $T = \{1, \dots, m\}$ in Theorem 5.3 gives the following immediate corollary, which supplies the missing proof for the construction described at the start of Section 5.3.

Corollary 5.5. *Every fair sack of profile $\mathbf{n}$ may be treated as a single semifair die of order t , loaded uniformly on $\{0, 1, \dots, t - 1\}$.*

Remark 5.6. Corollary 5.5 accounts rigorously for one of the two constructions of Section 5.3: since the whole sack collapses to a single semifair die of order $t = r$, adjoining a new die to it is no different, as far as Theorem 2.3 and Definition 2.5 are concerned, from adjoining a new die to any other single semifair die. It does not by itself establish that *every* fair sack arises by repeated mod addition and range addition from a $(n, n + 1)$ base case, which Section 5.3 reports as an empirical finding rather than a theorem.

References

- [1] Gasarch, W. I., & Kruskal, C. P. (1999). When can one load a set of dice so that the sum is uniformly distributed? *Mathematics Magazine*, 72(2), 133–138. <https://www.cs.umd.edu/%7Egasarch/papers/dice.pdf>