arXiv:2305.01593v1 [cs.DS] 2 May 2023

通过近凸最小加卷积加速 0-1 背包问题

Karl Bringmann* Alejandro Cassis†

2023年5月3日

摘要

我们重新审视经典的 0-1 背包问题：给定 n 个物品及其重量和收益，以及重量预算 W，目标是找出一个物品子集，使其总重量至多为 W，同时总收益最大。我们研究以任意物品的最大收益 $p_{\max}$和最大重量 $w_{\max}$为参数的伪多项式时间算法。我们的主要结果是针对 0-1 背包问题的算法，其运行时间分别为 $\widetilde{O}(n\, w_{\max}\, p_{\max}^{2/3})$ 和 $\widetilde{O}(n\, p_{\max}\, w_{\max}^{2/3})$，优于 Pisinger 在《算法杂志》[99]中提出的运行时间为 $O(n\, p_{\max}\, w_{\max})$ 的算法。在 $p_{\max} \approx w_{\max} \approx n$ （且 $W \approx$最优值 $\approx n^2$）的情况下，我们的算法首次突破了三次方时间复杂度下限 n^3。

为得到我们的结果，我们提出了一种高效算法，用于计算近凸函数的最小加卷积。更准确地说，如果存在一个凸函数 f，使得对于每个 i，$f(i) \le f(i) \le f(i) + \Delta$，那么我们称函数 $f\colon [n] \mapsto \mathbf{Z}$ 为 Δ-近$\breve{}$ $\breve{}$ $\breve{}$凸函数，其参数为 $\Delta \ge 1$。我们设计了一种算法，可在 $O(n\Delta)$ 时间内计算两个 Δ-近凸 $\breve{}$函数的最小加卷积。对于我们所知的所有应用，这一工具都可以取代 Bateni、Hajiaghayi、Seddighin 和 Stein 的预测技术 [STOC '18]，而且我们认为它适用范围更广。

*萨尔兰大学和马克斯·普朗克信息学研究所，萨尔兰信息学园区。本研究是 TIPEA 项目的一部分，该项目获得了欧洲研究理事会（ERC）欧洲联盟“地平线 2020”研究与创新计划的资助（资助协议编号：850979）。电子邮件：bringmann@cs.uni-saarland.de。

†萨尔兰大学和马克斯·普朗克信息学研究所，萨尔兰信息学园区。本研究是 TIPEA 项目的一部分，该项目获得了欧洲研究理事会（ERC）通过欧盟“地平线 2020”研究与创新计划提供的资助（资助协议编号：850979）。电子邮箱：acassis@cs.uni-saarland.de。

1 引言

在 0-1 背包问题中，给定一组 n 物品 $\mathcal{I} = \{(p_1, w_1) \ldots, (p_n, w_n)\}$，其中物品 i 的收益为 $p_i \in \mathbb{N}$，重量为 $w_i \in \mathbb{N}$，且重量预算为 $W \in \mathbb{N}$。目标是在约束条件 $\sum_{i=1}^{n} w_i x_i \leq W$ 和 $x \in \{0,1\}^n$ 下计算最优值 $\mathrm{OPT} := \max \sum_{i=1}^{n} p_i x_i$。这个计算机科学和运筹学领域的经典基础问题已被研究数十年（例如，参见介绍该主题及相关问题的著作 [KPP04]）。背包问题是弱 NP 困难问题，而 Bellman [Bel57]提出的教科书级动态规划算法可在 $O(n \cdot \min\{W, \mathrm{OPT}\})$ 时间内求解该问题。

近期研究探讨了背包问题及相关问题的细粒度复杂性，目标是针对不同参数给出尽可能优的伪多项式时间算法，参见表 1 以及 [Bri17、JW19、BW21、ABB+21、CH22、DMZ23、Kle22、EW18、JR19]。本文研究 0-1 背包问题关于两个自然参数的复杂性：物品中的最大重量，记为 $w_{\max}$，以及最大收益，记为 $p_{\max}$。注意，不失一般性，可以假设 $w_{\max} \leq W$ 且 $p_{\max} \leq$最优值。因此，在某些实例上，对这些参数具有较小多项式依赖的算法可以比标准动态规划算法更快。

此前已有多项研究考察了这种参数化，参见 表 1。为比较这些运行时间，注意到任何可行解至多包含全部物品，因此不失一般性，可以假设 $W \leq n \cdot w_{\max}$ 且 最优值 $\leq n \cdot p_{\max}$。注意，当$p_{\max} \approx w_{\max} \approx n$ (且 $W \approx$ 最优值 $\approx n^2$) 时，所有已知算法都需要 $\Omega(n^3)$的时间。特别是在这种情形下，Pisinger 于 '99 [Pis99] 提出的 运行时间为 $O(n\, w_{\max}\, p_{\max})$ 的算法仍是目前已知的最佳算法。本文 突破了这一三次方瓶颈：

定理 1. *存在一种用于 0-1 背包问题的随机化算法，其运行时间为* [1]

$$\widetilde{O}((p_{\max} W)^{2/3} (n w_{\max})^{1/3} + n w_{\max})$$

并且以高概率成功。利用上界 $W \leq n w_{\max}$，该运行时间至多为$\widetilde{O}(n\, w_{\max}\, p_{\max}^{2/3})$。

对称地，我们得到以下结论：

定理 2. *存在一种用于 0-1 背包问题的随机化算法，其运行时间为*

$$\widetilde{O}((w_{\max} \mathrm{OPT})^{2/3} (n p_{\max})^{1/3} + n p_{\max})$$

并且以高概率成功。利用上界 最优值 $\leq n p_{\max}$，该运行时间至多为$\widetilde{O}(n\, p_{\max}\, w_{\max}^{2/3})$。

最小加卷积 给定函数 f、g：$[n] \mapsto \mathbb{Z}$，其最小加卷积是函数 h：$[2n] \mapsto \mathbb{Z}$，定义为 $h(x) = \min_{x'} f(x') + g(x - x')$，其中 $x \in [2n]$。朴素算法可在 $O(n^2)$ 时间内计算该卷积，而目前已知的最佳算法运行时间为 $n^2/2^{\Omega(\sqrt{\log n})}$ [BCD+14、Wil18、CW21]。由于缺乏更快的算法，人们提出了最小加卷积假设，认为该问题不存在真正次二次时间算法 [CMWW19、KPS17]。尽管有这一假设，仍有一些具有结构的最小加卷积实例可以更快求解 [AKM+87、BHSS18、BHWZ94、CL15、CDXZ22]。这些改进是获得表 1 所列背包问题算法的关键（唯一的例外是 Bellman 和 Pisinger 的算法 [Bel57、Pis99]）：

[1] 我们使用 $\widetilde{O}(\cdot)$ 来忽略输入规模和最大输入数值中的多对数因子。

表 1：0-1 背包问题的伪多项式时间算法。

参考文献	运行时间
Bellman [Bel57]	$O(n \cdot \min\{W,$最优值$\})$
Pisinger [Pis99]	$O(n \cdot p_{\max} \cdot w_{\max})$
Kellerer 和 Pferschy [KP04],也 [BHSS18, AT19]	$\widetilde{O}(n + w_{\max} \cdot W)$
Bateni、Hajiaghayi、Seddighin 和 Stein [BHSS18]	$\widetilde{O}(n + p_{\max} \cdot W)$
Axiotis 和 Tzamos [AT19]	$\widetilde{O}(n \cdot$最小值$\{ w_{\max}^2, p_{\max}^2 \})$
Bateni、Hajiaghayi、Seddighin 和 Stein [BHSS18]	$\widetilde{O}((n+W) \cdot$最小值$\{ w_{\max}, p_{\max} \})$
Polak、Rohwedder 和 Węgrzycki [PRW21]	$O(n +$最小值$\{ w_{\max}^3, p_{\max}^3 \})$
Bringmann 和 Cassis [BC22]	$\widetilde{O}(n +(W +$最优值$)^{1.5})$
定理 1	$\widetilde{O}(n \cdot w_{\max} \cdot p_{\max}^{2/3})$
定理 2	$\widetilde{O}(n \cdot p_{\max} \cdot w_{\max}^{2/3})$

- 当其中一个函数为凸函数时，可以在以下时间内计算它们的最小加卷积$O(n)$ 利用 SMAWK 算法 [AKM+87]。Kellerer 和 Pferschy 间接将其用于背包问题[2] [KP04]，；Axiotis 和 Tzamos [AT19] 以及 Polak 等人 [PRW21]则明确地将其用于背包问题。

- 当函数单调且取值有界时，Chi 等人提出的算法可以在 $\widetilde{O}(n^{1.5})$ 时间内计算其最小加卷积。[CDXZ22]Bringmann 和 Cassis 已将其用于背包问题 [BC22]。

- Bateni 等人 [BHSS18]引入了预测技术，以证明可以高效计算由背包问题中的某些实例产生的最小加卷积。更准确地说，设 h 为两个给定函数 f、g的最小加卷积：$[n] \mapsto \mathbf{Z}$。他们证明，如果给定 n 个区间 $[x_i \,..\, y_i]$，满足以下条件：(i) 对每个 $i \in [n]$ 和 $j \in [x_i \,..\, y_i]$，$|h(i+j) - f(i)+g(j)| \leq \Delta$；(ii) 对于每个输出 $h(k)$，至少存在一个 i，使得 $f(i) + gk - i = h(k)$，并且 $k - i \in [x_i \,..\, y_i]$；以及 (iii) 对所有区间~和 $x_i \leq x_j$，$0 \leq x_i$、$y_i < n$，对所有 $i < j$，$y_i \leq y_j$；那么 h 可以在时间 $O(n \cdot \Delta)$内计算出来。他们证明，这一方法适用于背包问题的背景。

我们的定理 1和2 属于同一类改进，因为我们为一类新的结构化最小加卷积实例设计了高效算法，即近凸˘函数：我们称 $f\colon [n] \mapsto \mathbf{Z}$ 是 Δ-近凸的，如果存在一个凸函数 $f\colon [n] \mapsto \mathbf{Q}$ 使得 $\breve{f}(i) \leq f(i) \leq \breve{f}(i) + \Delta$ 对于所有 $i \in [n]$均成立。我们的定理如下：

定理 3(近凸最小加卷积)。设输入为 $f\colon [n] \mapsto [-U \,..\, U]$ 和 $g\colon [m] \mapsto [-U \,..\, U]$，其中 n、m、$U \in \mathbf{N}$。设 $\Delta \geq 1$，使得 f 和 g 均为 Δ-近凸的~凸函数。则 f 与 g 的最小加卷积可在 $O((n+m) \cdot \Delta)$ 时间内计算完成。

我们将定理 3 视为 Bateni 等人提出的预测技术的替代方案。[BHSS18]事实上，预测技术的所有应用都利用近凸性来确保满足其前提条件，因此据我们所知，所有这类应用都可以由我们的定理 3取代。由于预测技术既难以表述又难以应用，我们认为我们的定理 3 将预测技术替换为一种易于应用且表述简洁的工具。此外，定理 3为具有结构的最小加卷积实例提供了一种新工具。本文利用该工具推进了 0-1 背包问题的研究，我们相信它也具有更广泛的适用性。

[2]Kellerer 而 Pferschy 没有使用 SMAWK，而是给出了一种用于计算最小加法卷积的不同算法娱乐网址美国卷积这些实例的运行时间为 $O(n\log\ n)$。

我们的技术 我们证明定理 3 的方法如下。令 f、g: $[n] \mapsto \mathbf{Z}$ 为输入函数，并令 h 为它们的最小加卷积，这是我们希望计算的对象。首先我们观察到，可以获得凸近似 f、g，以见证 Δ-近凸性，以及 $f^{\smile}$ 和 g，并高效地计算它们的最小加卷积 h。通过利用 h 以及 $f^{\smile}$ 和 $g^{\smile}$ 的凸性，我们识别出一个结构化集合 $R \subseteq [n]^2$，其具有如下性质：任意 $(i,j) \in [n]^2 \setminus R$ 都满足 $f(i) + g(j) > h(j)$。随后，我们给出一个简单的递归算法，用一族 $\mathcal{C}$ 互不相交的二进盒 $I \times J$ 覆盖 R，其中 $(I,J) \in \mathcal{C}$ 满足 I、$J \subseteq [n]$ 和 $I \times J \subseteq R$。因此，我们可以通过计算和集 $A := \{(i, f(i)) \mid i \in I\} + \{(j, g(j)) \mid j \in J\}$，并对每个 $(I,J) \in \mathcal{C}$ 取 $h(k) = \min\{y \mid (k,y) \in A\}$，来推断 h。为高效完成这一过程，我们注意到，在 I 和 J 内，函数 $f[I]$ 和 $g[J]$ 与具有相同斜率的线性函数相近，其加性误差至多为 $\pm O(\Delta)$（这由它们的 Δ-近凸性推出）。这意味着它们的和集很小；更准确地说，其大小为 $O((|I| + |J|)\Delta)$。最后，我们利用已知工具，这些工具可以在与和集大小成正比的时间内计算和集。通过小和集的覆盖来高效计算最小加卷积的思想，受到 Chan 和 Lewenstein 的 [CL15] 算法的启发，该算法针对有界单调序列（其中他们完全没有使用凸性）。我们的算法与 Bateni 等人的预测技术 [BHSS18]存在一些相似之处。特别是，用函数近似线性的二进盒进行覆盖，类似于他们利用算法所需的区间 $[x_i \,..\, y_i]$ 的方式。

为证明定理 1 和定理 2，我们采用划分与卷积范式；这一范式已用于近期许多子集和与背包问题算法中，参见例如 [Bri17、BHSS18、BC22、JR19、Cha18、KX19、DJM23、BN21]。具体而言，我们将物品随机划分为 q 组。在每组中，我们使用标准动态规划算法，计算仅使用该组物品时，对于每个重量 i，在重量不超过 i 的条件下可达到的最大收益 $f(i)$。然后通过最小加卷积组合所有组对应的函数 f。关键的观察是，由于随机划分，我们只需计算一个较小重量区间内的值 $f(i)$。

进一步的相关工作 Cygan 等人 [CMWW19] 和 Künnemann 等人 [KPS17] 证明了，在最小加卷积假设下，对于满足以下条件的实例，不存在真正的次二次时间背包问题算法：$w_{\max}$，$W = \Theta(n)$ 且 $p_{\max}$，最优值 $= \Omega(n^2)$；对满足以下条件的实例，也有对称的结论：$p_{\max}$，最优值 $= \Theta(n)$ 且 $w_{\max}$，$W = \Omega(n^2)$。这意味着，在这些情形下，Bellman 的动态规划算法在条件意义下是最优的。

以 $p_{\max}$ 和 $w_{\max}$ 为参数的伪多项式时间算法，也被用于研究密切相关的无界背包问题。这里的设定与 0-1 背包问题相同，但解现在可以包含每种物品的任意多个副本。Chan 和 He [CH22] 为这个问题给出了一个运行时间为 $O(n \cdot \min\{p_{\max}, w_{\max}\})$的算法；根据最小加卷积假设，这是最优的。Bringmann 和 Cassis [BC22] 给出了一个运行时间为 $\widetilde{O}(n + (p_{\max} + w_{\max})^{1.5})$ 的算法，当 $w_{\max} \approx p_{\max} \approx n$时，该算法更快。

概述本文结构安排如下。在第 2 节中，我们介绍一些形式化预备知识并确立相关记号。在第 3 节中，我们给出背包问题的算法，并在假设定理 1和2成立的前提下证明定理 3。在第 4 节中，我们将给出最小加卷积的算法，并证明定理 3。

2 预备知识

我们记 $\mathbf{N} = \{0, 1, 2, \ldots\}$。对于 $t \in \mathbf{N}$，我们定义 $[t] := \{0, 1, \ldots, t\}$。令 $A \in \mathbf{Z}^{n+1}$ 为一个整数序列，即对于 $i \in [n]$，有 $A[i] \in \mathbf{Z}$。有时我们也将这样的序列称为一个函数A: $[n] \mapsto \mathbf{Z}$。据此，我们用记号 $-A$ 表示 A 的逐项取反。给定 $a, b \in \mathbf{R}$ 且满足 $a \le b$，我们定义 $[a \,..\, b] := \{\max(0, \lfloor a \rfloor), \max(0, \lfloor a \rfloor) + 1, \ldots, \lceil b \rceil - 1, \lceil b \rceil\}$。

在 $[a..b]$ 的定义中使用非标准舍入和截断至 0，有助于在 a 和 b 可能不是非负整数时，对子序列 $A[a..b]$ 进行索引。

两个序列 $A[0..n] \in \mathbf{Z}^{n+1}$，$B[0..m] \in \mathbf{Z}^{m+1}$ 的最大加卷积记为 MaxConv(A, B)，它是一个长度为 $n+m+1$ 的序列，其中对于每个 $k \in [n+m]$，都有 MaxConv(A, B)$[k] := \max_{i+j=k} A[i] + B[j]$。最小加卷积 MinConv($A$, B) 的定义类似，只需将 max 替换为 min。注意，对序列中的元素取负后，这两种运算是等价的。

事实 4 . 对于任意 $A \in \mathbf{Z}^{n+1}$、$B \in \mathbf{Z}^{m+1}$，都有 $\textsc{MaxConv}(A, B) = -\textsc{MinConv}(-A, -B)$.

我们将使用以下简便记法：给定序列 $A[0..n]$，$B[0..n]$、区间I，$J \subseteq [n]$ 和 $K \subseteq [2n]$，对于每个 $k \in K$，记 $C[K] := \text{MaxConv}(A[I], B[J])$ 表示计算$C[k] := \max\{A[i] + B[j] : i \in I, j \in J, i + j = k\}$。

我们称函数 $f : [n] \mapsto \mathbf{Q}$为凸的，如果对每个 $i \in [1..n-1]$，$f(i) - f(i-1) \le f(i+1) - f(i)$ 都成立。若 $f$is凹的，则称其为凹的。

定义 5 （近似凸与近似凹函数）。对于 $\Delta \ge 0$，我们称函数 $f : [n] \mapsto \mathbf{Z}$ 是 Δ-近似凸的，如果存在一个凸函数 $f : [n] \mapsto \mathbf{Q}$，使得 $f(i) \le f(i) \le f(i) + \Delta$。若 f *is* Δ-近似凹，则 $-f$ *is* Δ-近似凸。

~如果输入由 N 个 $[-U..U]$ 中的数字组成，我们将 $O(T) = \bigcup_{c \ge 0} O(T \log^c(NU))$ 记作。

3 更快的 0-1 背包算法

本节证明定理 1。设 $(\mathcal{I}, W)$ 为一个 0-1 背包实例。下文中，我们用 n 表示物品的数量：$= |\mathcal{I}|$。我们将物品集合 $\mathcal{I}$ 与 $\{1, \ldots, n\}$ 等同。我们用指示向量 $x \in \{0,1\}^n$ 表示该背包实例的一个解（即 $\mathcal{I}$ 的一个子集）。对于物品集合 $\mathcal{J} \subseteq \mathcal{I}$ 的一个子集，我们令 $w_{\mathcal{J}}(x) := \sum_{i \in \mathcal{J}} w_i x_i$ 且 $p_{\mathcal{J}}(x) := \sum_{i \in \mathcal{J}} p_i x_i$。我们定义利润序列 $\mathcal{P}_{\mathcal{I}}[\cdot]$，其中对每个 $j \in \mathbf{N}$，均有

$$\mathcal{P}_{\mathcal{I}}[j] = \max\{\, p_{\mathcal{I}}(x) \mid x \in \{0,1\}^n, w_{\mathcal{I}}(x) \le j \,\}.$$

注意，$\mathcal{P}_{\mathcal{I}}$ 是单调不减的，且最优值 OPT $= \mathcal{P}_{\mathcal{I}}[W]$。计算 $\mathcal{P}_{\mathcal{I}}[0..j]$ 的教科书式方法是使用动态规划：

事实 6. 对于任意 $j \in \mathbf{N}$，序列 $\mathcal{P}_{\mathcal{I}}[0..j]$ 可在 $O(nj)$时间内计算得到。

在介绍算法之前，我们先对给定的背包实例 $(\mathcal{I}, W)$ 作两个简单的观察。首先，忽略重量大于容量 W 的物品后，不失一般性，可以假设 $w_{\max} \le W$。现在，每个单独的物品都是一个可行解，因此有 $p_{\max} \le$ 最优值。其次，注意到如果 $W \ge n \cdot w_{\max}$，那么该实例就是平凡的，因为我们可以装下所有物品。因此，不失一般性，可以假设 $W \le n \cdot w_{\max}$。此外，由于任意可行解至多包含全部 n 物品，因此可得最优值 $\le n \cdot p_{\max}$。

算法

现在我们描述该算法。设置参数 $q := \min\{(n/p_{\max})^{2/3}(W/w_{\max})^{1/3}, W/w_{\max}\}$向下取整到最近的 2的幂，$\Delta := w_{\max} W/q$ 以及 $\eta := 11 \log n$。对于每个 $\ell \in [\log q]$，我们定义区间 J^ℓ：
$= [\frac{W}{q} 2^\ell - \sqrt{\Delta 2^\ell \eta} .. \frac{W}{q} 2^\ell + \sqrt{\Delta 2^\ell \eta}]$。

我们先将物品 $\mathcal{I}$ 均匀随机地分成 q 组 $\mathcal{I}_1^0$，……，$\mathcal{I}_q^0$。我们的思路是为每个 $\mathcal{I}_j^0$ 计算一个关联的数组 C_j^0，然后以树状方式将它们合并。影响运行时间的一个关键因素是，我们只计算每个数组 C_j^ℓ 的 $|J^\ell|$ 个条目。具体步骤如下：

基础情形对于每个 $\mathcal{I}_j^0$，我们使用事实 6来计算 $\mathcal{P}_{\mathcal{I}_j^0}[0\,..\,\frac{W}{q}+\sqrt{\Delta\eta}]$并定义子数组 $C_j^0[J^0] := \mathcal{P}_{\mathcal{I}_j^0}[J^0]$.

组合 遍历 层级 $\ell = 1, \ldots q$。对于 $j \in [1\,..\,q/2^\ell]$我们设置 $\mathcal{I}_j^\ell := \mathcal{I}_{2j-1}^{\ell-1} \cup \mathcal{I}_{2j}^{\ell-1}$。然后，通过选取 $C_j^\ell[J^\ell]$中的相关条目，计算子数组；该数组是$C_{2j-1}^{\ell-1}[J^{\ell-1}]$与 $C_{2j}^{\ell-1}[J^{\ell-1}]$的最大加卷积。

返回答案 （注意，当 $\ell = \log(q)$时，有 $\mathcal{I}_1^{\log q} = \mathcal{I}$。）我们返回值 $C_1^{\log q}[W]$。伪代码见 算法 1 。

算法 1 背包算法。给定一组物品 $\mathcal{I}$ 和重量预算 W，该算法计算可达到的最大收益。

$q \leftarrow \min\{\,(n/p_{\max})^{2/3}(W/w_{\max})^{1/3}, W/w_{\max}\,\}$ rounded down to the closest power of 2
$\Delta \leftarrow w_{\max} W/q$
$\eta \leftarrow 11 \log n$
$\mathcal{I}_1^0, \ldots, \mathcal{I}_q^0 \leftarrow$ random partitioning of $\mathcal{I}$ into q groups
for $i = 1 \ldots q$ **do**
    Compute $\mathcal{P}_{\mathcal{I}_j^0}[0\,..\,\frac{W}{q}+\sqrt{\Delta\eta}]$ using standard dynamic programming (Fact 6)
    $J^0 \leftarrow [\,\frac{W}{q} - \sqrt{\Delta\eta}\,..\,\frac{W}{q} + \sqrt{\Delta\eta}\,]$
    $C_j^0[J^0] \leftarrow \mathcal{P}_{\mathcal{I}_j^0}[J^0]$
for $\ell = 1 \ldots \log(q)$ **do**
    $J^\ell \leftarrow [\,\frac{W}{q}2^\ell - \sqrt{\Delta 2^\ell \eta}\,..\,\frac{W}{q}2^\ell + \sqrt{\Delta 2^\ell \eta}\,]$
    for $j = 1, \ldots, q/2^\ell$ **do**
        $\mathcal{I}_j^\ell \leftarrow \mathcal{I}_{2j-1}^{\ell-1} \cup \mathcal{I}_{2j}^{\ell-1}$
        Compute $C_j^\ell[J^\ell] \leftarrow \textsc{MaxConv}(C_{2j-1}^{\ell-1}[J^{\ell-1}], C_{2j}^{\ell-1}[J^{\ell-1}])$ using Theorem 3
return $C_1^{\log q}[W]$

正确性

我们首先分析该算法的正确性。下面的引理表明，将任意解限制在集合 $\mathcal{I}_j^\ell$ 中后，其重量集中在期望值附近。

引理 7 （集中性）。设 $x \in \{0,1\}^n$ 是给定背包实例的一个解。固定层级 $\ell \in [0\,..\,\log q]$ 和 $j \in [1\,..\,q/2^\ell]$。那么，以至少 $1 - 1/n^4$ 的概率，有：

$$\left| w_{\mathcal{I}_j^\ell}(x) - \frac{w_{\mathcal{I}}(x) \cdot 2^\ell}{q} \right| \leq \sqrt{\Delta 2^\ell} \cdot 10 \log n.$$

证明。回顾一下，物品集合 $\mathcal{I}$ 被随机划分为 $\mathcal{I}_1^0, \ldots, \mathcal{I}_q^0$。因此，注意到 $\mathcal{I}_j^\ell$ 是 $\mathcal{I}$ 的随机子集，其中每件物品以 $p := 2^\ell/q$ 的概率被纳入。对于 $i \in [1\,..\,n]$，令 Z_i 为一个随机变量：它以 p 的概率取值 $w_i \cdot x_i$，以 $1-p$ 的概率取值 0。于是，$w_{\mathcal{I}_j^\ell}(x)$ 的分布为 $Z := \sum_{i=1}^n Z_i$，因此，$\mathbf{E}(Z) = w_{\mathcal{I}}(x)\,p$。

为了证明该结论，我们将使用伯恩斯坦不等式（例如参见 [DP09，定理 1。2]），其表明

$$\begin{aligned} \mathbf{P}(|Z - \mathbf{E}(Z)| \geq t) &\leq 2\exp\left(-\frac{t^2}{2\,\mathbf{Var}(Z) + \frac{2}{3}t \cdot w_{\max}}\right) \\ &\leq 2\exp\left(-\min\left\{\frac{t^2}{4\,\mathbf{Var}(Z)}, \frac{t}{2w_{\max}}\right\}\right). \end{aligned} \tag{1}$$

令 $t := \sqrt{p \cdot w_{\max} W \cdot 10 \log n}$。我们首先给出 $t^2/(\mathbf{4}\text{方差}(Z))$ 的上界。注意，我们可以如下给出方差的上界：

$$\mathbf{Var}(Z) = \sum_{i=1}^{n} p(1-p) w_i^2 x_i^2 \le p \cdot w_{\max} \sum_{i=1}^{n} w_i x_i = p \cdot w_{\max} w_{\mathcal{I}}(x) \le p \cdot w_{\max} W.$$

因此 $t^2/(\mathbf{4Var}(Z)) \ge 10 \log n$。接下来，我们对 $t/(2w_{\max})$ 进行界定。利用 $q \le W/w_{\max}$，可得 $p = \frac{2^\ell}{q} \ge \frac{w_{\max} 2^\ell}{W} \ge \frac{w_{\max}}{W}$。因此，

$$\frac{t}{2w_{\max}} = \frac{\sqrt{p \cdot w_{\max} W} \cdot 10 \log n}{2w_{\max}} \ge 5 \log n.$$

结合上述结果，由（1）可得

$$|w_{\mathcal{I}_j^\ell}(x) - w_{\mathcal{I}}(x) 2^\ell / q| = |Z - \mathbf{E}(Z)| \le t = \sqrt{p w_{\max} W} \cdot 10 \log n = \sqrt{\Delta 2^\ell} \cdot 10 \log n$$

成立的概率至少为 $1 - 2/n^5 \ge 1 - 1/n^4$。 □

利用 引理 7，我们可以证明，在层级 ℓ，只需计算一个长度为$\widetilde{O}(\sqrt{\Delta 2^\ell})$、以 $W 2^\ell / q$为中心的子数组。下面的引理将对此作出精确说明：

引理 8。设 $x \in \{0,1\}^n$ 是给定背包实例的一个解，满足 $w_{\mathcal{I}}(x) \in [W - w_{\max} .. W]$。对于所有层级 $\ell \in [0 .. \log q]$ 以及所有$j \in [1 .. q/2^\ell]$，以下结论成立的概率至少为 $1 - 1/n^2$：

- $w_{\mathcal{I}_j^\ell}(x) \in J^\ell = [\frac{W}{q} 2^\ell - \sqrt{\Delta 2^\ell}\eta .. \frac{W}{q} 2^\ell + \sqrt{\Delta 2^\ell}\eta]$, *and*
- $C_j^\ell[w_{\mathcal{I}_j^\ell}(x)] \ge p_{\mathcal{I}_j^\ell}(x)$.

证明。 根据 引理 7，对于每个 $\ell \in [0 .. \log q]$ 和 $j \in [1 .. q/2^\ell]$，都有

$$|w_{\mathcal{I}_j^\ell}(x) - w_{\mathcal{I}}(x) 2^\ell / q| \le \sqrt{\Delta 2^\ell} \cdot 10 \log n \tag{2}$$

其成立的概率至少为 $1 - 1/n^4$。注意 $q \le W/w_{\max} \le n$。因此，我们可以使用并集界，得出（2）对所有 $\ell \in [0 .. \log q]$ 和 $j \in [1 .. q/2^\ell]$ 成立的概率至少为 $1 - 1/n^2$。从现在起，我们以该事件成立为条件。

我们首先证明陈述中的第一项。固定 $\ell \in [\log q]$ 和 $j \in [1 .. q/2^\ell]$。由（2），有 $|w_{\mathcal{I}_j^\ell}(x) - w_{\mathcal{I}}(x) 2^\ell / q| \le \sqrt{\Delta 2^\ell} \cdot 10 \log n$。根据假设，有 $w_{\mathcal{I}}(x) \in [W - w_{\max} .. W]$。因此，

$$\begin{aligned} |w_{\mathcal{I}_j^\ell}(x) - W 2^\ell / q| &\le |w_{\mathcal{I}_j^\ell}(x) - w_{\mathcal{I}}(x) 2^\ell / q| + \tfrac{2^\ell}{q} |w_{\mathcal{I}}(x) - W| \\ &\le \sqrt{\Delta 2^\ell} 10 \log n + w_{\max} 2^\ell / q \le \sqrt{\Delta 2^\ell} \cdot 11 \log n. \end{aligned}$$

最后一个不等式成立，是因为我们可以利用 $2^\ell \le q$ 和 $w_{\max} \le W$ 得到 $w_{\max} 2^\ell / q \le \sqrt{w_{\max} W} \cdot \sqrt{2^\ell / q} = \sqrt{\Delta 2^\ell}$。由于 $\eta = 11 \log n$，这意味着 $w_{\mathcal{I}_j^\ell}(x) \in J^\ell = [\frac{W}{q} 2^\ell - \sqrt{\Delta 2^\ell}\eta .. \frac{W}{q} 2^\ell + \sqrt{\Delta 2^\ell}\eta]$。至此，第一个物品的证明完成。

接下来，我们通过归纳法证明该引理的第二项。考虑基础情形 $\ell = 0$。根据第一项，对于任意 $j \in [1 .. q]$，我们有 $w_{\mathcal{I}_j^0}(x) \in J^0$。特别地，有$C_j^0[w_{\mathcal{I}_j^0}(x)] = \mathcal{P}_{\mathcal{I}_j^0}[w_{\mathcal{I}_j^0}(x)]$（见 第 8 行）。然后，由于 $\mathcal{P}_{\mathcal{I}_j^0}[i]$ 是 $\mathcal{I}_j^0$中重量至多为 i的物品子集的最大收益，因此有 $\mathcal{P}_{\mathcal{I}_j^0}[w_{\mathcal{I}_j^0}(x)] \ge p_{\mathcal{I}_j^0}(x)$，从而完成了基础情形的证明。

现在我们进行归纳步骤：固定 $\ell \geq 1$，并假设 $C_j^{\ell-1}[w_{\mathcal{I}_j^{\ell-1}}(x)] \geq p_{\mathcal{I}_j^{\ell-1}}(x)$对所有 $j \in [1\,..\,q/2^{\ell-1}]$均成立。根据该引理的第一项，对于每个 $j \in [1\,..\,q/2^{\ell}]$，我们有 $w_{\mathcal{I}_j^{\ell}}(x) \in J^{\ell}$。因此，根据 第 13 行的计算，有

$$\begin{aligned} C_j^{\ell}[w_{\mathcal{I}_j^{\ell}}(x)] &= \max\{\, C_{2j-1}^{\ell-1}[i] + C_{2j}^{\ell}[i'] \colon i, i' \in J^{\ell-1},\, i + i' = w_{\mathcal{I}_j^{\ell}}(x) \,\} \\ &\geq C_{2j-1}^{\ell-1}[w_{\mathcal{I}_{2j-1}^{\ell-1}}(x)] + C_{2j}^{\ell}[w_{\mathcal{I}_{2j}^{\ell-1}}(x)] \\ &\geq p_{\mathcal{I}_{2j-1}^{\ell-1}}(x) + p_{\mathcal{I}_{2j}^{\ell-1}}(x) = p_{\mathcal{I}_j^{\ell}}(x). \end{aligned}$$

在第二步中，我们利用了前面所示的 $w_{\mathcal{I}_{2j-1}^{\ell-1}}(x)\, w_{\mathcal{I}_{2j}^{\ell-1}}(x) \in J^{\ell-1}$。第三步由归纳假设得出。最后一个等式成立，因为 $\mathcal{I}_j^{\ell} = \mathcal{I}_{2j-1}^{\ell-1} \cup \mathcal{I}_{2j}^{\ell-1}$。 □

引理 9 (算法 1的正确性)。 设 $x^* \in \{0,1\}^n$ 是给定背包实例的一个最优解。那么，对于每个 $i \in [w_{\mathcal{I}}(x^*)\,..\,W]$，均有 $C_1^{\log q}[i] = \mathcal{P}_{\mathcal{I}}[i]$，且其概率至少为 $1 - 1/n^2$。

证明。 我们可以在线性时间内检验 $O(n)$ 最优解是否由所有物品组成；若是，则该实例显然成立。因此，不失一般性，假设 x^* 不包含所有物品。特别地， x^*至少遗漏一个物品，因此其重量满足 $w_{\mathcal{I}}(x^*) \in [W - w_{\max}\,..\,W]$。根据 引理 8，有 $C_1^{\log q}[w_{\mathcal{I}}(x^*)] \geq p_{\mathcal{I}}(x^*) = \mathcal{P}_{\mathcal{I}}[w_{\mathcal{I}}(x^*)]$，其概率至少为 $1 - 1/n^2$。从现在起，我们以该事件为条件。下面将用到以下辅助断言：

▷命题 10。序列 $C_1^{\log q}[J_{\log q}]$是单调非递减的，并且对所有 $i \in J^{\log q}$都满足 $C_1^{\log q}[i] \leq \mathcal{P}_{\mathcal{I}}[i]$。

证明。 首先，我们通过归纳法证明单调性。注意，在基础情形 $\ell = 0$中，序列 $C_j^0[J^0] = \mathcal{P}_{\mathcal{I}_j^0}[J^0]$根据 $\mathcal{P}_{\mathcal{I}_j^0}$的定义是单调非递减的。对于层级 $\ell > 0$，序列 C_j^{ℓ}通过对层级 $\ell - 1$的序列进行最大加卷积得到。注意到两个单调非递减序列的最大加卷积仍然是单调非递减的，即可得出结论。

命题的第二部分成立，因为（通过归纳法）每个对应于 $i \in J^{\log q}$ 的条目 $C_1^{\log q}[i]$，都表示由 $\mathcal{I}$ 个物品组成、总重量不超过 i 的某个子集的收益。◁

由于 x^*是一个最优解，因此有 $\mathcal{P}_{\mathcal{I}}[i] = p_{\mathcal{I}}(x^*)$对所有 $i \in [w_{\mathcal{I}}(x^*)\,..\,W]$均成立。因此，断言 10 推出 $C_1^{\log q}[i] = \mathcal{P}_{\mathcal{I}}[i]$对所有 $i \in [w_{\mathcal{I}}(x^*)\,..\,W]$均成立，证明完毕。 □

运行时间

现在我们分析 算法 1的运行时间。关键的加速来自 第 13 行中的计算，在此我们使用 定理 3 来执行最大加卷积。由于 定理 3是以近似凹函数的最小加卷积来表述的，因此我们将使用以下推论：

推论 11。 给定输入 $f\colon [n] \mapsto [-U\,..\,U]$和 $g\colon [m] \mapsto [-U\,..\,U]$，其中$U \in \mathbf{N}$。令 $\Delta \geq 1$ 满足 f 和 g 均为 Δ-近似凹。则，$\textsc{MaxConv}(f, g)$ 可在时间 $O((n+m)\Delta)$内计算

证明。 注意到 $-f$ 和 $-g$ 是 Δ-近凸的（定义 5），该结果由 定理 3和 事实 4得出。 □

以下引理表明，两个近凹序列的最大加卷积仍然是近凹的。

引理 12. 设 $f\colon [n] \mapsto \mathbf{Z}$ 是 Δ_f-近似凹的，且 $g\colon [m] \mapsto \mathbf{Z}$ 是 Δ_g-近似凹的。那么，$h := \textsc{MaxConv}(f, g)$ 是 Δ_h-近似凹的，且 $\Delta_h \le$ 最大值$\{\Delta_f, \Delta_g\}$。

证明。设 $f, \breve{g}$ 是逐点最小凹函数，满足 $f \ge f$，$\breve{g} \ge g$，并令 $h := \text{MaxConv}(f, \breve{g})$。我们将证明 $h \ge h \ge h - \max\{\Delta_f, \Delta_g\}$，由此可推出结论。

为证明 $h \ge h$，固定 $k \in [n+m]$，并令 i^* 为 $h(k)$ 的见证，即 $h(k) = f(i^*)+g(k-i^*)$。于是，$h(k) \ge f(i^*) + \breve{g}(k - i^*) \ge f(i^*) + g(k - i^*) = h(k)$。因此 $h \ge h$。

为证明对于 $\Delta := \max\{\Delta_f, \Delta_g\}$，固定 $k \in [n + m]$，并令 i^* 为 $h(k)$ 的见证，即 $h(k) = f(i^*) + \breve{g}(k - i^*)$。注意，$f$ 是由 f 上的个点构成的分段线性插值。特别地，存在 $i_L \le i^* \le i_R$，使得 $f(i_L) = f(i_L)$ $f(i_R) = f(i_R)$，且 $f(i)$ 对 $i \in [i_L \,..\, i_R]$ 是线性的。类似地，对于 $j^* := k - i^*$，存在 $j_L \le j^* \le j_R$，使得 $\breve{g}(j_L) = g(j_L)$ $\breve{g}(j_R) = g(j_R)$，且 $\breve{g}(j)$ 对 $j \in [j_L \,..\, j_R]$ 是线性的。我们选取满足此性质的最大 i_L, j_L 和最小 i_R, j_R。

令 $\hat{\imath}_L := \max\{i_L, k - j_R\}$，$\hat{\imath}_R := \min\{i_R, k - j_L\}$。注意到，函数 $\check{s}(i) := f(i) + \breve{g}(k - i)$ 对于 $i \in [\hat{\imath}_L \,..\, \hat{\imath}_R]$ 是线性的，并且 $\hat{\imath}_L \le i^* \le \hat{\imath}_R$。此外，根据 $\breve{h}$ 的定义，我们有 $\check{s}(i) = f(i) + \breve{g}(k - i) \le h(k)$，对于 $i \in [\hat{\imath}_L \,..\, \hat{\imath}_R]$。由于 i^* 是 $h(k)$ 的见证，我们有 $\check{s}(i^*) = \breve{h}(k)$。结合以上结论，对于所有 $i \in [\hat{\imath}_L \,..\, \hat{\imath}_R]$，我们得到 $\check{s}(i) = \breve{h}(k)$。特别地，$\breve{f}(\hat{\imath}_L) + \breve{g}(k - \hat{\imath}_L) = \breve{h}(k)$，因此

$$\begin{aligned} h(k) &\ge f(\hat{\imath}_L) + g(k - \hat{\imath}_L) \\ &= \breve{f}(\hat{\imath}_L) + \breve{g}(k - \hat{\imath}_L) + (f(\hat{\imath}_L) - \breve{f}(\hat{\imath}_L)) + (g(k - \hat{\imath}_L) - \breve{g}(k - \hat{\imath}_L)) \\ &= \breve{h}(k) + (f(\hat{\imath}_L) - \breve{f}(\hat{\imath}_L)) + (g(k - \hat{\imath}_L) - \breve{g}(k - \hat{\imath}_L)). \end{aligned} \tag{3}$$

最后，由于 $\hat{\imath}_L \in \{i_L, k - j_R\}$ 和 $f(i_L) = f(i_L)$ 以及 $g(j_R) = \breve{g}(j_R)$，式 (3) 中最后两个求和项之一必定为 0。利用 f 和 g 的近凹性，我们可以将另一个求和项界定为 $f(\hat{\imath}_L) - f(\hat{\imath}_L) \ge -\Delta_f$ 或 $g(k - \hat{\imath}_L) - \breve{g}(k - \hat{\imath}_L) \ge -\Delta_g$。由此可得 $h(k) \ge h(k) - \Delta_f$ 或 $h(k) \ge \breve{h}(k) - \Delta_g$。无论哪种情况，我们都可以得出结论：对于每个 $k \in [n + m]$，$h(k) \ge \breve{h}(k) - \max\{\Delta_f, \Delta_g\}$ 成立。 □

下一个引理表明，我们合并的序列在第 13 行是近凹的 e.

引理 13 (近似凹性)。 对于每个层级 $\ell \in [1 \,..\, q]$ 和每个 $j \in [1 \,..\, q/2^\ell]$，都有 $C_j^\ell[J^\ell]$ 是 $p_{\max}$-近似凹的。

证明。 我们通过归纳法证明该陈述。考虑基础情形 $\ell = 0$。对于每个 $j \in [1 \,..\, q]$，都有 $C_j^0[J^0] = \mathcal{P}_{\mathcal{I}_j^0}[J^0]$。接下来，我们论证 $\mathcal{P}_{\mathcal{I}_j^0}$ 是 $p_{\max}$-近似凹的。考虑背包问题的分数贪心解：按利润重量比对物品 $(p_1, w_1) \ldots, (p_m, w_m)$ 进行非递减排序，使得 $p_1/w_1 \ge p_2/w_2 \ge \cdots \ge p_m/w_m$。令 $M := \sum_{i=1}^m w_i$。然后，通过设置断点构造序列 $\mathcal{P}[0 \,..\, M]$

$$\tilde{\mathcal{P}}[0] = 0,\ \tilde{\mathcal{P}}[w_1] = p_1,\ \tilde{\mathcal{P}}[w_1 + w_2] = p_1 + p_2,\ \ldots,\ \tilde{\mathcal{P}}[w_1 + \cdots + w_m] = p_1 + \cdots + p_m,$$

以及每一对相邻断点之间的线性插值。通过这种方式，$\mathcal{P}[i]$ 对应于容量为 i 的背包问题的<escape>分数型 </escape>版本的最优解，即在这种设定下，物品可以按分数形式装入解中。

▷ 命题 14。序列 $\mathcal{P}$ 是凹的，并且对于每个 $i \in [M]$，都有 $\mathcal{P}[i] \ge \mathcal{P}_{\mathcal{I}_j^0}[i] \ge \mathcal{P}[i] - p_{\max}$。

证明。对于每个 $i \in [1 \,..\, M - 1]$，有 $\mathcal{P}[i] - \mathcal{P}[i-1] \ge \mathcal{P}[i+1] - \mathcal{P}[i]$，因为按收益重量比排序后，断点之间线性部分的斜率非递减。这意味着 $\mathcal{P}$ 是凹的。

对于每个 $i \in [M]$，都有 $\mathcal{P}[i] \geq \mathcal{P}_{\mathcal{I}_j^0}[i]$，因为 $\mathcal{P}[i]$ 是分数背包问题的最优解。此外，注意到达到 $\mathcal{P}[i]$ 的解至多包含一个按分数分配的物品。移除该物品后，我们得到一个容量为 i 的 0-1 背包问题的可行（整数）解，且收益最多减少 $p_{\max}$。这意味着 $\mathcal{P}_{\mathcal{I}_j^0}[i] \geq \mathcal{P}[i] - p_{\max}$。◁

根据 命题 14，我们得出 $\mathcal{P}_{\mathcal{I}_j^0}[0\,..\,M]$ 是 $p_{\max}$-近似凹的（参见 定义 5），因此 $C_j^0[J^0]$ 也具有这一性质，从而完成基础情形的证明。

对于归纳步骤，考虑层级 $\ell > 0$。固定一个 $j \in [1\,..\,q/2^\ell]$。根据归纳假设，$C_{2j-1}^{\ell-1}[J^{\ell-1}]$ 和 $C_{2j}^{\ell-1}[J^{\ell-1}]$ 都是 $p_{\max}$-近似凹的。因此，根据 引理 12 可得 $C_j^\ell[J^\ell] = \textsc{MaxConv}(C_{2j-1}^{\ell-1}[J^{\ell-1}],\ C_{2j}^{\ell-1}[J^{\ell-1}])$ 是 $p_{\max}$-近似凹的，从而完成证明。□

引理 15。固定一个层级 $\ell \in [1\,..\,q]$ 和一次迭代 $j \in [1\,..\,q/2^\ell]$。计算 C_j^ℓ 在第 13 行 所需的时间为 $\widetilde{O}(p_{\max}\sqrt{\Delta 2^\ell})$

证明。根据 引理 13，序列 $C_{2j-1}^{\ell-1}[J^{\ell-1}]$、$C_{2j}^{\ell-1}[J^{\ell-1}]$ 是 $p_{\max}$-近似凹的。因此，根据推论 11，它们的最大加卷积可在 $\widetilde{O}(p_{\max}|J^\ell|) = \widetilde{O}(p_{\max}\sqrt{\Delta 2^\ell})$ 的时间内计算出来，其中用到了 $\eta = O(1)$。□

引理 16（算法 1 的**运行时间**）。算法 1 的运行时间为

$$\widetilde{O}((p_{\max}W)^{2/3}(nw_{\max})^{1/3} + nw_{\max}).$$

证明。回想一下，$q = \min\{(n/p_{\max})^{2/3}(W/w_{\max})^{1/3}, W/w_{\max}\}$（相差至多一个 2 的因子）。由于 $W \leq nw_{\max}$，我们有 $q \leq n$。此外，由于不失一般性地假设 $w_{\max} \leq n$，注意到 $q < 1$ 当且仅当 $q = (n/p_{\max})^{2/3}(W/w_{\max})^{1/3} < 1$。这意味着 $p_{\max} > n\sqrt{W/w_{\max}}$。但在这种情况下，所声称的运行时间为 $\Omega(nW)$，因此标准的 $O(nW)$ 动态规划算法（事实 6）已经达到了我们的时间界。因此，我们可以不失一般性地假设 $1 \leq q \leq n$，即 q 是将物品集合 $\mathcal{I}$ 划分为若干组时，组数的一个有效选择。

我们先界定基础情形的运行时间，即计算数组 C_j^0 对于 $j \in [1\,..\,q]$ 在第 5 行中。根据事实 6 以及定义 $\Delta = w_{\max}W/q$，这需要的时间为

$$O\left(\sum_{j=1}^{q} |\mathcal{I}_j^0|(\tfrac{W}{q} + \sqrt{\Delta\eta})\right) = O\left(n(\tfrac{W}{q} + \sqrt{\Delta\eta})\right) = \widetilde{O}\left(n\tfrac{W}{q} + n\sqrt{\tfrac{w_{\max}W}{q}}\right). \tag{4}$$

现在，我们来界定在第 9 行至第 14 行中执行的组合步骤的耗时。在层级 $\ell \in [1\,..\,q]$ 和第 $j \in [1\,..\,q/2^\ell]$ 次迭代中，执行第 13 行所需时间为 $\widetilde{O}(p_{\max}\sqrt{\Delta 2^\ell})$，根据引理 15 可得。因此，总耗时可界定为

$$\sum_{\ell=1}^{\log q}\sum_{j=1}^{q/2^\ell} \widetilde{O}(p_{\max}\sqrt{\Delta 2^\ell}) = \sum_{\ell=1}^{\log q} \frac{q}{2^\ell}\widetilde{O}\left(p_{\max}\sqrt{\tfrac{w_{\max}W}{q}2^\ell}\right) = \sum_{\ell=1}^{\log q} \widetilde{O}\left(p_{\max}\sqrt{\tfrac{qw_{\max}W}{2^\ell}}\right),$$

由于这是一个等比级数，因此其上界为首项 $O(p_{\max}\sqrt{qw_{\max}W})$。将其与 (4) 结合，可得总耗时为

$$\widetilde{O}\left(p_{\max}\sqrt{qw_{\max}W} + n\tfrac{W}{q} + n\sqrt{\tfrac{w_{\max}W}{q}}\right).$$

回顾 $q = \Theta(\min\{(n/p_{\max})^{2/3}(W/w_{\max})^{1/3},\ W/w_{\max}\})$，我们得到总时间

$$\widetilde{O}((p_{\max}W)^{2/3}(nw_{\max})^{1/3} + nw_{\max} + (p_{\max}W)^{1/3}(nw_{\max})^{2/3}).$$

最后注意到，根据算术-几何平均不等式，我们有

$$\begin{aligned}(p_{\max}W)^{1/3}(nw_{\max})^{2/3} &= \sqrt{(p_{\max}W)^{2/3}(nw_{\max})^{1/3}nw_{\max}} \\ &\le O((p_{\max}W)^{2/3}(nw_{\max})^{1/3} + nw_{\max}).\end{aligned}$$

因此，总运行时间为 $\widetilde{O}((p_{\max}W)^{2/3}(nw_{\max})^{1/3} + nw_{\max})$，如所述。 □

证明*定理 1*。运行算法 1。根据 引理 9，我们得到 $\mathcal{I}_1^{\log q}[W] =$最优值，其概率至少为 $1-1/n^2$，这证明了正确性。运行时间由 引理 16直接得到。注意，通过重复算法 $c/2$ 次，我们可以获得成功概率 $1-1/n^c$，其中任意常数 $c \ge 2$ 均适用。 □

定理 2 的证明概要 我们的论述集中于证明定理 1。定理 2 中所述对称变体的证明非常相似，因此我们仅概述所需的修改。本质上，我们需要在各处将利润与重量互换，这相应意味着将最大加卷积替换为最小加卷积。具体而言：我们不再处理利润序列 $\mathcal{P}_{\mathcal{I}}$，而是处理重量序列 $\mathcal{W}_{\mathcal{I}}$，其中条目 $\mathcal{W}_{\mathcal{I}}[j]$存储利润至少为 j的解的最小重量。我们不知道最优值，但 $\tilde{}$ $\tilde{}$ $\tilde{}$可以在线性时间内计算出一个满足 $V - p_{\max} \le$ 最优值 $\le V$ 的近似值 V（参见例如 [KPP04，《定理 2.5》4]）。在算法中，我们将所有 $w_{\max}$ 替换为 $p_{\max}$，并将所有 W 替换为 $\tilde{V}$。经过这些修改，函数 C_j^{ℓ} 现在是 $w_{\max}$-近似凸的（而不是 $p_{\max}$-近似凹的），因此我们直接使用定理 3，而不是推论 11。通过这种方式，我们得到数组 $C_1^{\log q}[\tilde{V} - p_{\max} \,..\, \tilde{V}] = \mathcal{W}_{\mathcal{I}}[\tilde{V} - p_{\max} \,..\, \tilde{V}]$。然后，我们可以将最优值推断为满足 $\mathcal{W}_{\mathcal{I}}[i] \le W$的最大 $i \in [V - p_{\max} \,..\, V]$ $\tilde{}$ $\tilde{}$。

重构最优解到目前为止，我们只关注返回给定背包实例的最优收益。要重构一个满足 $x \in \{0,1\}^n$ 使得$p_{\mathcal{I}}(x) =$ 最优值的解，我们按如下方式进行。运行算法 1 后，我们得到每个 $\ell \in [\log q]$和 $j \in [1 \,..\, q/2^{\ell}]$对应的序列$C_1^{\ell}[J^{\ell}]$。对于输出项 $C_1^{\log q}[W]$，我们找到一个见证 $i \in J^{\log q - 1}$，即满足 $C_1^{\log q-1}[i] + C_2^{\log q-1}[W-i] = C_1^{\log q}[W]$的索引 i。只需尝试所有可能性，就能在 $|J^{\log q-1}| = \widetilde{O}(\sqrt{\Delta q}/2)$时间内完成。然后，我们继续递归地为 i 和 $W-i$寻找见证。最终，我们会在每个数组 C_j^0中找到一个对应于 $j \in [1 \,..\, q]$的项。请注意，这一步所需时间与所有序列 $\sum_{\ell=0}^{\log q} q/2^{\ell} \cdot O(|J^{\ell}|) = \widetilde{O}(q\sqrt{\Delta}) = \widetilde{O}(\sqrt{q w_{\max} W}) \le \widetilde{O}(nw_{\max})$的总长度成正比，其中最后一步用到了 $q \le W/w_{\max}$ 和 $W \le nw_{\max}$。最后，注意到每个数组 C_j^0 都是通过事实 6 中的标准动态规划算法计算得到的，该算法允许我们在计算固定项 $C_j^0[i]$ 所需的相同时间内取回对应的解。因此，我们可以在不增加总体运行时间的情况下取回最优解。

4 近凸序列的最小加卷积

在本节中，我们证明定理 3。

定理 3（近凸最小加卷积）。给定输入 $f\colon [n] \mapsto [-U \,..\, U]$和 $g\colon [m] \mapsto [-U \,..\, U]$，其中 n、m、$U \in \mathbf{N}$。令 $\Delta \ge 1$ 满足 f 和 g 均为 Δ-近 $\tilde{}$ 凸。那么，f 和 g 的最小加卷积可在 $O((n+m)\cdot\Delta)$ 时间内计算。

4.1 预备知识

在本节中，固定函数 $f\colon [n] \mapsto [-U \,..\, U]$，$g\colon [m] \mapsto [-U \,..\, U]$。回顾一下，$\breve{}$我们称 $f\colon [n] \mapsto \mathbf{Z}$ 是 Δ_f-近凸的，如果存在一个凸函数 $f\colon [n] \mapsto \mathbf{Q}$，使得

$\breve{f}(i) \le f(i) \le f(i) + \Delta_f$对所有 $i \in [n]$均成立（见定义 5）。首先注意，点 $\{(i, f(i)) \mid i \in [n]\}$ 的下凸包给出了满足 $f \le f$ 的逐点最大凸函数 f。由于这些点已按 x 坐标排序，因此可通过 Graham 扫描 [Gra72],在 $O(n)$ 时间内计算。然后可以推出 $\Delta_f = \max\{1, \max_{i\in[n]} f(i) - \breve{f}(i)\}$。因此，从现在起，我们假设已知 $f, \Delta_f, \breve{g}, \Delta_g$。令 $\Delta := \max\{\Delta_f, \Delta_g\}$。设 $h :=$ 最小卷积$(f, \breve{g})$，且 $h :=$ 最小卷积(f, g)。目标是计算 h。

我们先引入一些记号。我们称 $(i, j) \in [n] \times [m]$ 为一个点。我们将点 (i, j) 视为位于一个网格的第 i 行、第 j 列，其中 (0,0) 位于左下角，(n, m) 位于右上角。点 (i, j) 位于对角线 $i + j$ 上。对于任意 $\delta \ge 0$，若点 (i, j) 满足 $\breve{f}(i) + \breve{g}(j) \le \breve{h}(i + j) + \delta$，则该点是 δ-相关 的。我们用 R_δ 表示所有δ-相关点的集合。

0-相关点之所以重要，是因为有如下观察：若 i 是 $h(k)$ 的一个见证，则 $f(i) + \breve{g}(k - i) = h(k)$。因此注意到，当且仅当 $(i, k - i)$ 是一个 0-相关点时，i 才是 $h(k)$ 的见证。

2Δ相关点的重要性由以下引理体现：

引理 17. 若 $(i, j) \notin R_{2\Delta}$ 则 $f(i) + g(j) > h(i + j)$。

Th也就是说，在计算 h 时，可以忽略那些与 2Δ 无关的点 .

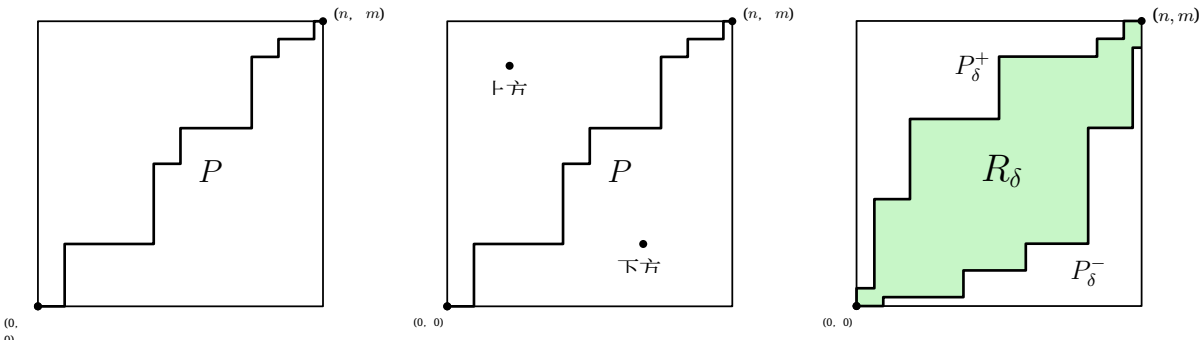
证明。 由于 (i, j) 与 2Δ 无关，因此有 $f(i) + g(j) \ge f(i) + \breve{g}(j) > h(i + j) + 2\Delta$。设 $k := i + j$，并令 i^* 为 $h(k)$ 的一个见证，即 $f(i^*) + \breve{g}(k - i^*) = h(k)$。于是，

$$h(k) \le f(i^*) + g(k - i^*) \le \breve{f}(i^*) + \Delta + \breve{g}(k - i^*) + \Delta = \breve{h}(k) + 2\Delta < f(i) + g(j). \qquad \square$$

我们称点集 P为一条单调路径，若对每个 $k \in [n + m]$ P恰好包含一个点 (i_k, j_k) 位于对角线 k上，且满足 $(i_{k+1}, j_{k+1}) \in \{(i_k + 1, j_k),$ 以及 $(i_k, j_k + 1)\}$ 对每个$k \in [n + m - 1]$均成立；示意图见 图 1a 。对于任意 $\delta > 0$，我们令

$$P_\delta^+ := \{(i, k - i) \mid k \in [n + m], i \in [n] \text{ is maximal s.t. } (i, k - i) \text{ is } \delta\text{-relevant}\},$$
$$P_\delta^- := \{(i, k - i) \mid k \in [n + m], i \in [n] \text{ is minimal s.t. } (i, k - i) \text{ is } \delta\text{-relevant}\}.$$

接下来的两个引理表明 P_δ^+，P_δ^- 是单调路径，并且 P_δ^+，P_δ^- 构成了 R_δ的边界，示意见 图 1c 。这确立了 R_δ 的结构，后续我们将利用这一点。



(a) 一条单调路径 P　　(b) P上方和下方的点　　(c) R_δ 介于 P_δ^+ 和 P_δ^- 之间

图 1：第 4 节 中所用概念的可视化。

引理 18 (单调路径)。 对于任意 $\delta \ge 0$, P_δ^-, P_δ^+，它们都是单调路径。

˘证明。由于每个 $k \in [n+m]$, $h(k)$ 都有一个见证，因此存在一个 0-相关点 $(i, k-i)$。由于每个 0-相关点也都是 δ-相关的，因此可知 P_δ^- 对每个 $k \in [n+m]$ 都恰好包含一个满足 $i_k + j_k = k$ 的点 (i_k, j_k)。

接下来，我们证明 $i_{k-1} \le i_k \le i_{k-1} + 1$ 对所有 $k \in [1..n+m]$ 都成立。由于$i_k + j_k = k$, 因此也可推出 $k - 1 - j_{k-1} \le k - j_k \le k - 1 - j_{k-1} + 1$，从而得到$j_{k-1} \le j_k \le j_{k-1} + 1$。由于 $i_{k+1} + j_{k+1} = k + 1 = i_k + j_k + 1$，可知 $(i_k,\ j_k) \in \{(i_{k-1} + 1,\ j_{k-1}),\ (i_{k-1},\ j_{k-1} + 1)\}$。因此，只需证明 $i_{k-1} \le i_k \le i_{k-1} + 1$。我们分两种情形讨论。

情形 1： $i_k \le i_{k-1}$。我们证明在此情形下 $i_k \ge i_{k-1}$ （因此 $i_k = i_{k-1}$）。令 i^*_{k-1} 为˘的一个见证 h（$k-1$）。注意，根据 P_δ^- 的定义， $i^*_{k-1} \ge i_{k-1}$

$$\breve{f}(i_k) + \breve{g}(k - i_k) \le \breve{h}(k) + \delta \le \breve{f}(i^*_{k-1}) + \breve{g}(k - i^*_{k-1}) + \delta,$$

其中，第一个不等式根据 P_δ^- 的定义成立，第二个不等式则根据˘的定义成立。重新整理可得

$$\breve{g}(k - i_k) - \breve{g}(k - i^*_{k-1}) \le \breve{f}(i^*_{k-1}) - \breve{f}(i_k) + \delta. \tag{5}$$

由于 $i^*_{k-1} \ge i_{k-1} \ge i_k$，我们有 $k - i_k \ge k - i^*_{k-1}$。根据 $\breve{g}$ 的凸性，可得

$$\breve{g}(k - 1 - i_k) - \breve{g}(k - 1 - i^*_{k-1}) \le \breve{g}(k - i_k) - \breve{g}(k - i^*_{k-1}). \tag{6}$$

结合（5）和（6），并重新整理，可得

$$\breve{f}(i_k) + \breve{g}(k - 1 - i_k) \le \breve{f}(i^*_{k-1}) + \breve{g}(k - 1 - i^*_{k-1}) + \delta = \breve{h}(k-1) + \delta,$$

其中，最后一个等式根据 i^*_{k-1} 的定义成立。因此，$(i_k,\ k - 1 - i_k)$ 是 δ-相关的。由于 i_{k-1} 是使 $(i_{k-1},\ k - 1 - i_{k-1})$ 为 δ-相关的最小值，我们得到 $i_{k-1} \le i_k$，如所需。

情形 2： $i_k > i_{k-1}$。我们证明在此情形下 $i_k \le i_{k-1} + 1$ （从而 $i_k = i_{k-1} + 1$）。令 i^*_k˘为 h（k）的一个见证。根据 P_δ^- 的定义 $i^*_k \ge i_k$。此外，

$$\breve{f}(i_{k-1}) + \breve{g}(k - 1 - i_{k-1}) \le \breve{h}(k-1) + \delta \le \breve{f}(i^*_k - 1) + \breve{g}(k - i^*_k) + \delta,$$

其中，第一个不等式由 P_δ^- 的定义得出，第二个不等式则由 h 的定义˘得出。整理后可得

$$\breve{g}(k - 1 - i_{k-1}) - \breve{g}(k - i^*_k) \le \breve{f}(i^*_k - 1) - \breve{f}(i_{k-1}) + \delta. \tag{7}$$

˘由于 $i^*_k - 1 \ge i_k - 1 \ge i_{k-1}$，且根据 f 的凸性，我们有

$$\breve{f}(i^*_k - 1) - \breve{f}(i_{k-1}) \le \breve{f}(i^*_k) - \breve{f}(i_{k-1} + 1). \tag{8}$$

将（7）与（8）合并并整理，得到

$$\breve{f}(i_{k-1} + 1) + \breve{g}(k - 1 - i_{k+1}) \le \breve{f}(i^*_k) - \breve{g}(k - i^*_k) + \delta = \breve{h}(k) + \delta,$$

其中，最后一个等式由 i^*_k 的定义成立。因此，$(i_{k-1}+1,\ k-1-i_{k+1})$ 是 δ-相关的。由于其对角线为 k， $i_{k-1}+1$ 是 i_k 的一个可能取值。根据 i_k 的最小性（由 P_δ^- 的定义可知），我们得到 $i_k \le i_{k-1} + 1$。

在这两种情况下，我们都得到 $i_k \in \{i_{k-1}, i_{k-1} + 1\}$，从而证明了该命题。至此， P_δ^- 的证明完成。 P_δ^+ 的证明是对称的（交换 f 和 g 的角色，基本上会使 P_δ^- 和 P_δ^+ 翻转）。 □

设 (i,j) 为一个点，P 为一条单调路径。设 $(a,b) \in P$ 为与 (i,j) 位于同一对角线上的唯一点，即 $a+b=i+j$。我们称 (i,j) 位于 P 下方，当且仅当 $i<a$；位于 P 上方，当且仅当 $i>a$；位于 P 上，当且仅当 $i=a$。示意图见图 1b。

引理 19。 对于任意 $\delta \geq 0$，R_δ 由所有位于 P_δ^+ 之上或其上的点 (i,j) 以及位于 P_δ^- 之下或其上的点组成。

证明。固定 $k \in [n+m]$，令 $(i^+,\ k-i^+)$ 和 $(i^-,\ k-i^-)$ 分别为 P_δ^+ 和 P_δ^- 中对角线 k 上的点。考虑对角线 k 上的任意 $(i,\ j) \in R_\delta$。由 i^+ 的最大性可得 $i \leq i^+$，类似地，由 i^- 的最小性可得 $i \geq i^-$。因此，R_δ 中没有点位于 P_δ^+ 之上或 P_δ^- 之下。接下来只需证明，对于任意 $i^- \leq i \leq i^+$，我们有 $(i,\ k-i) \in R_\delta$。注意，函数 $r(i) := \check{f}(i) + \check{g}(k-i)$ 是凸函数（因为它是凸函数之和）。由于 $(i^+,\ k-i^+)$ 是 δ-相关的，我们有 $r(i^+) \leq \check{h}(k) + \delta$。类似地，由于 $(i^-,\ k-i^-)$ 是 δ-相关的，我们有 $r(i^-) \leq \check{h}(k) + \delta$。由 r 的凸性可得，对于所有 $i^- \leq i \leq i^+$，$r(i) \leq \check{h}(k) + \delta$。因此，我们得出，对于每个 $i^- \leq i \leq i^+$，都有 $(i,\ k-i) \in R_\delta$。

□

最后，我们需要一些关于和集的背景知识。给定 $A,\ B \subseteq [-U\,..\,U]^2$ 其中 $U \in \mathbf{N}$，我们将 $A+B = \{a+b \mid a \in A, b \in B\}$ 定义为它们的和集，其中加法 $a+b$ 按分量进行。计算 $A+B$ 的朴素方法需要时间 $O(|A| \cdot |B|)$。对于我们的应用，我们希望在接近其大小 $|A+B|$ 的线性时间内计算和集。为此，我们将使用以下工具来计算稀疏非负卷积。给定向量 $P,\ Q \in \mathbf{N}^n$，它们的卷积 $P \star Q \in \mathbf{N}^{2n-1}$ 按坐标定义为 $(P \star Q)[k] = \sum_{i+j=k} P[i] \cdot Q[j]$。

定理 20 (确定性稀疏卷积 [BFN22])。存在一种确定性算法，可在 $A, B \in \mathbf{N}^n$ 时间内计算两个非负向量的卷积，复杂度为 $O(t\,\mathrm{polylog}(n\Delta))$，其中 t 是 $A \star B$ 中的非零项数量，且 Δ 是 A 中的最大项，且 B。

另请参见 [BFN21]，以了解以随机化为代价改进对数因子的工作，以及[CH02、Nak20、GGdC20]此前具有类似保证的随机化算法。

推论 21 （输出敏感的和集计算）。给定 $A\ B \subseteq [-U\,..\,U]^2$ $|A+$，若有 $B| \leq N$，则 $A+B$ 可在时间 $O(N)$ 内计算得到。

证明。令 $A' := \{(x+U) \cdot 5U + (y+U) \mid (x,\ y) \in A\}$，类似地，令 $B' := \{(x+U) \cdot 5U + (y+U) \mid (x,\ y) \in B\}$。注意，这给出了从 $A,\ B \subseteq [-U\,..\,U]^2$ 到 $A',\ B' \subseteq [\Theta(U^2)]$ 的一一嵌入。此外，可以验证，给定 $C' := A' + B'$，我们可以推断出 $C := A+B$（选择 $5U$ 可以防止坐标在求和时相互作用）。

因此，只需计算 $A'+B'$。为此，构造它们的指示向量 $P_{A'},\ P_{B'} \in \mathbf{N}^{\Theta(U^2)}$ 并计算卷积 $P_{C'} = P_{A'} \star P_{B'}$。$P_{C'}$ 中的非零项对应于 $A'+B'$ 的元素。根据 定理 20，该过程的运行时间为 $O(|A'+B'|\,\mathrm{polylog}(N,U)) = O(N)$。

□

4.2 算法

我们已准备好描述该算法。回顾一下，我们可以访问函数 $f,\ \check{f},\ g,\ \check{g}$ 以及数值 $\Delta = \max\{\Delta_f, \Delta_g\}$。

计算 $\check{h}$ = 最小卷积($\check{f}$, $\check{g}$)。 请参阅 算法 2中的伪代码。

引理 22. 算法 2 计算 h = 最小卷积$(f, \check{g})$ 的时间为 $O(n+m)$。

证明。运行时间是显然的。为证明正确性，关注算法 2 中计算的 i_k^* for $k \in [n+m]$。我们声称，路径 P_0^- 等于 $\{(i_k^*, k-i_k^*) \mid k \in [n+m]\}$。也就是说，我们要论证，对于每个 $k \in [n+m]$，i_k^* 是 $h(k)$ 的最小见证。事实上，根据引理 18，P_0^- 是一条单调路径。因此，$i_k^* \in \{i_{k-1}^*, i_{k-1}^*+1\}$。注意，在第 3 行中，我们将 i_k^* 选为 $\check{f}(i) + \check{g}(k-i) + \frac{i}{2n}$ 的最小化元，其中 $i \in \{i_{k-1}^*, i_{k-1}^*+1\}$。因此，该算法正确地计算出 i_k^*（加性项 $i/(2n)$ 确保我们选择最小的 i）。由于 i_k^* 是 $h(k)$ 的最小见证，该算法对所有 $k \in [n+m]$ 正确地计算出 $h(k)$。

□

算法 2 给定凸函数 $f\colon [n] \mapsto \mathbf{Q}$ $\breve{g}\colon [m] \mapsto \mathbf{Q}$，该算法计算 $\breve{h} =$ 最小卷积$(f, \breve{g})$。

$\breve{1}\colon i_0^* \leftarrow 0$，$h(0) \leftarrow f(0) + \breve{g}(0)$ $2\colon$ **对于** $k = 1, \ldots\ldots, n+m$ **执行** $3\colon i_k^* \leftarrow \operatorname{argmin}\{\breve{f}(i) + \breve{g}(k-i) + \overset{i}{2n} \mid i \in \{i_{k-1}^*, i_{k-1}^* + 1\} \cap [n]\}$ $4\colon \breve{h}(k) \leftarrow \breve{f}(i_k^*) + \breve{g}(k - i_k^*)$

计算 $\boldsymbol{h} =$ 最小卷积$(\boldsymbol{f}, \boldsymbol{g})$。回顾一下，$f\colon [n] \mapsto \mathbf{Z}$，且 $g\colon [m] \mapsto \mathbf{Z}$。作为最后一步简化，我们指出，不失一般性，可以假设 $n = m$，且 $n+1$ 是 2 的幂。为此，令 N 为大于 $\max\{n, m\}$ 的最小 2 的幂。我们将函数补齐至长度 N，令 $f(n+j) := 2j \cdot W$（当 $j \in [1\,..\,N-1-n]$ 时），且 $g(m+j) := 2j \cdot W$（当 $j \in [1\,..\,N-1-m]$ 时），其中 W 是一个大于 $\max_{i\in[n]} f(i) + \max_{j\in[m]} g(j)$ 的整数。注意，结果 $h =$ 最小卷积(f, g) 的条目 $h(0), \ldots, h(n+m)$ 保持不变（这是因为 W 取得足够大的值），因此我们可以从补齐后的函数所得结果中读出原始结果。此外，注意到补齐不会改变参数 Δ_f 和 Δ_g。

现在我们可以描述算法了。在运行 算法 2 之后，我们可以假设已经计算出 $\breve{h}$ 以及见证路径 $P_0^- = \{(i_k^*, k - i_k^*) \mid k \in [n+m]\}$。我们将使用以下子程序：

- 相关(i, j)：返回 $f(i) + \breve{g}(j) \le \breve{h}(i+j) + 2\Delta$。
- 下方见证路径(i, j)：返回 $i < i_{i+j}^*$
- 上方见证路径(i, j)：返回 $i > i_{i+j}^*$

现在我们可以计算 $h =$ 最小卷积(f, g)，方法是调用 RecMinConv$([0\,..\,n], [0\,..\,m])$。伪代码见 算法 3 。

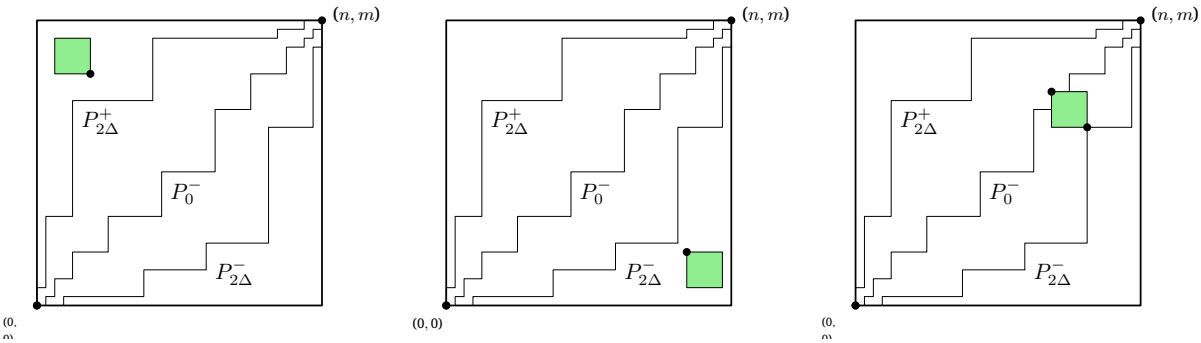
算法 3 给定区间 $I = [i_A\,..\,i_B]$，$J = [j_A\,..\,j_B]$，该算法计算贡献-$f[I]$ 和 $g[J]$ 对 MinConv(f, g) 的贡献。

过程 RecMinConv$(I = [i_A\,..\,i_B], J = [j_A\,..\,j_B])$
  如果 上方见证路径(i_A, j_B) 且 NotRelevant(i_A, j_B) **则** ▷ 情形 1
    返回 $\tilde{h}(k)$ 对所有 $k \in [i_A + j_A\,..\,i_B + j_B]$
  如果 下方见证路径(i_A, j_B) 且不相关(i_B, j_A) **则** ▷ 情形 2
    返回 $\tilde{h}(k) = \infty$ 对所有 $k \in [i_A + j_A\,..\,i_B + j_B]$
  如果 相关(i_A, j_B) 且相关(i_B, j_B) **则** ▷ 情形 3
    计算 $C \leftarrow \{(i, f(i)) \mid i \in I\} + \{(j, g(j)) \mid j \in J\}$ 使用 推论 21
    推断 $\tilde{h}(k) \leftarrow$ 所有 $\{y \mid (k, y) \in C\}$ 的最小值 $k \in [i_A + j_A\,..\,i_B + j_B]$
    返回 $\tilde{h}$
  else ▷ 情形 4
    将 I 拆分为两个长度相等的区间 I_1，I_2，类似地，将 J 拆分为 J_1，J_2
    递归计算 $\tilde{g}_{i,j} \leftarrow$ RecMinConv(I_i, J_j)，其中 $i, j \in \{1, 2\}$
    返回 函数 $\tilde{g}_{i,j}$ 在 $i, j \in \{1, 2\}$ 上的逐点最小值

算法 3 递归地计算 $f[i_A\,..\,i_B]$ 和 $g[j_A\,..\,j_B]$ 对 $h =$ 最小卷积(f, g) 的贡献。接下来我们讨论它的四种情形；情形 1–3 的示意图见图 2。若 (i_A, i_B) 位于见证路径 P_0^- 上方，且与 2Δ 不相关（情形 1），那么正如我们将在下文论证的，$I \times J$ 中没有点会对输出 h 作出贡献，因此此时返回一个哑函数（它处处等于 $+\infty$）。情形 2 对称：(i_B, j_A) 位于 P_0^- 上方，且与 2Δ 不相关；此时我们

同样返回一个哑函数。情形 3 适用于 (i_A, j_B) 和 (i_B, j_A) 都与 2Δ 相关的情况。此时，我们通过计算和集 $C = \{(i, f(i)) \mid i \in I\} + \{(j, g(j)) \mid j \in J\}$，显式计算 $h =$ 最小卷积$(f[i_A..i_B], g[j_A..j_B])$，并将 $h(k)$ 推断为满足 $(k, y) \in C$ 的最小 y；根据和集的定义，这等于满足 $i \in I$，$j \in J$ 和 $i + j = k$ 的最小 $f(i) + g(j)$。注意，通过扫描 C 的所有元素一次，这一步可以在总时间 $O(|C|)$ 内对所有 $k \in [i_A + j_A..i_B + j_B]$ 完成。

最后，如果上述情形均不适用，我们将两个区间 I 和 J 分别划分为相等的两半，并递归处理这两半的全部 4 种组合。我们通过对所有计算得到的函数取逐点最小值来将它们组合起来。这种情形本质上是蛮力法。



（a）情形 1　　（b）情形 2　　（c）情形 3

图 2：情形 1-3 的可视化，来自 算法 3。绿色框表示当前子问题。

正确性

我们首先分析该算法的正确性。

引理 23 (算法 3 的正确性)。 RecMinConv$([0..n], [0..m])$ *(算法 3)* 正确地计算 $h =$ 最小卷积(f, g)。

证明。 设 $k \in [n+m]$，并考虑对角线 k 上的点 (i^*, j^*)，使得 $f(i^*) + g(j^*) = h(k)$，即它是 $h(k)$ 的一个见证。我们要论证，某个递归调用会计算 $f(i^*) + g(j^*)$。在情形 4 中这显然成立，因为 (i^*, j^*) 被一个递归子问题覆盖。情形 3 中也显然成立，因为此时会显式计算 $f(i^*) + g(j^*)$。

为完成正确性证明，我们论证 (i^*, j^*) 不可能属于适用情形 1 或 2 的子问题。回顾一下，若 (i_A, j_B) 位于 P^-_0 上方，且 (i_A, j_B) 与 2Δ 无关，则情形 1 适用于子问题 $I = [i_A..i_B], J = [j_A..j_B]$。由于 (i_A, j_B) 与 2Δ 无关，根据引理 19，(i_A, j_B) 必定位于 $P^+_{2\Delta}$ 上方或 $P^-_{2\Delta}$ 下方。由于 (i_A, j_B) 位于 P^-_0 上方，它只能位于 $P^+_{2\Delta}$ 上方。由于 (i_A, j_B) 是 $I \times J$ 的右下角，因此 $I \times J$ 中的所有点都位于 $P^+_{2\Delta}$ 上方。于是，根据引理 19，$I \times J$ 中的所有点都与 2Δ 无关。反设 $(i^*, j^*) \in I \times J$，则引理 17 推出 $f(i^*) + g(j^*) > h(k)$，这与 (i^*, j^*) 是 $h(k)$ 的见证这一选择矛盾。因此，(i^*, j^*) 不可能属于情形 1 的子问题。情形 2 的论证对称。至此，正确性证明完成。

□

运行时间

接下来，我们分析运行时间。关键洞见是，在相关区域内，两个函数本质上都是线性的，并且具有相同的斜率（见 引理 24）。这意味着情形 3 中计算出的和集很小（见 引理 25），因此可以利用 推论 21高效地计算。在接下来的两个引理中，设 $I = [i_A..i_B] \subseteq [n]$ 和 $J = [j_A..j_B] \subseteq [m]$ 是长度相同的区间 $|I| = |J|$。

引理 24 (相关区域内的近线性)。 如果 $I \times J \subseteq R_{2\Delta}$，那么存在 a、b、$c \in \mathbf{R}$，使得 $|f(i) - (a \cdot i + b)| \leq 2\Delta$ 对所有 $i \in I$ 以及 $|g(j) - (a \cdot j + c)| \leq 2\Delta$ 均成立。

证明。 考虑 $(i_A, f(i_A))$ 与 $(i_B, f(i_B))$ 之间的线性插值：

$$F(x) := \frac{(i_B - x)\breve{f}(i_A) + (x - i_A)\breve{f}(i_B)}{i_B - i_A}.$$

类似地，考虑

$$G(x) := \frac{(j_B - x)\breve{g}(j_A) + (x - j_A)\breve{g}(j_B)}{j_B - j_A}.$$

由 f 和 g 的凸性可知，我们有

$$\breve{f}(i) \leq F(i) \quad \forall i \in I, \qquad \breve{g}(j) \leq G(j) \quad \forall j \in J. \tag{9}$$

考虑对角线 $k := i_A + j_B$，并注意到对于所有 $i \in I$，由于 $|I| = |J|$，我们有 $k - i \in J$。因此，对于每个 $i \in I$，点 $(i, k - i)$ 都是 2Δ-相关的，于是得到

$$\breve{h}(k) \leq \breve{f}(i) + \breve{g}(k - i) \leq \breve{h}(k) + 2\Delta \quad \forall i \in I \tag{10}$$

这意味着

$$\breve{h}(k) \leq F(i) + G(k - i) \leq \breve{h}(k) + 2\Delta \quad \forall i \in I, \tag{11}$$

由于根据 (10)，这些不等式对 $i \in \{i_A, i_B\}$ 成立；根据线性插值，它们在两者之间也成立。

现在，对于任意 $i \in I$，我们有

$$\breve{f}(i) \overset{(10)}{\geq} \breve{h}(k) - \breve{g}(k - i) \overset{(9)}{\geq} \breve{h}(k) - G(k - i) \overset{(11)}{\geq} \breve{h}(k) - (\breve{h}(k) + 2\Delta - F(i)) = F(i) - 2\Delta.$$

因此，$f(i) \in [F(i) - 2\Delta \,..\, F(i)]$，由 $f \leq f \leq f + \Delta_f \leq f + \Delta$ 可得 $|f(i) - F(i)| \leq 2\Delta$。

对于任意 $j \in J$，我们对 $g(j)$ 作如下界定：

$$\breve{g}(j) \overset{(10)}{\geq} \breve{h}(k) - \breve{f}(k - j) \overset{(9)}{\geq} \breve{h}(k) - F(k - j),$$

and

$$\breve{g}(j) \overset{(9)}{\leq} G(j) \overset{(11)}{\leq} \breve{h}(k) - F(k - j) + 2\Delta.$$

因此，$|g(j) - (h(k) - F(k - j) + \Delta)| \leq \Delta$。由 F 的线性性，我们可以写成 $F(k - j) = F(k) - F(j) + F(0)$。由此得到 $|g(j) - (F(j) + \lambda)| \leq \Delta$，对于 $\lambda := h(k) - F(k) - F(0) + \Delta$。由于 $|g(j) - g(j)| \leq \Delta_g \leq \Delta$，我们得到 $|g(j) - (F(j) + \lambda)| \leq 2\Delta$。由于 F 是线性的，令 $F(i) = a \cdot i + b$ 且 $F(j) + \lambda = a \cdot j + c$，证明即告完成。 □

引理 25 (相关区域具有小和集)。 如果 $I \times J \subseteq R_{2\Delta}$ 那么和集 $\{(i, f(i)) \mid i \in I\} + \{(j, f(j)) \mid j \in J\}$ 的大小为 $O(\Delta \cdot (|I| + |J|))$。

证明。 根据 引理 24，对于任意 $(i, j) \in I \times J$ 满足 $i + j = k$ 的情形，我们有

$$f(i) + g(j) = (a \cdot i + b) + (a \cdot j + c) \pm O(\Delta) = a \cdot k + b + c \pm O(\Delta).$$

因此，对于每个 $|I| + |J| - 1$ x 坐标（即 $i + j$ 的取值），和集中有 $O(\Delta)$ 个不同的 y 坐标（即 $f(i) + g(j)$ 的取值）。 □

引理26 （算法 3的运行时间）。 RecMinConv（$[0 \,..\, n]$, $[0 \,..\, m]$） （算法 3） 的运行时间为 $O(n\Delta)$。

证明。 我们首先分析一个递归子问题的运行时间，暂不考虑递归调用的开销。注意，在情形 1 和情形 2 中，只需返回一个虚设值，即无需~遍历 $k \in [\, i_A + j_A \,..\, i_B + j_B \,]$ 来显式返回 $h(k) = \infty$。因此，情形 1 和情形 2 的运行时间为 $O(1)$。我们将这部分时间计入当前子问题的父问题，而当前子问题是一个情形 4 子问题。

考虑情形 4。忽略递归子问题的代价，情形 4 的运行时间为 $O(1)$，这也涵盖了对落入情形 1 和情形 2 的子问题所进行的计费。

考虑情形 3，设 $s := i_B - i_A + 1 = j_B - j_A + 1$ 为当前边长。根据 引理 25，~在 第 7 行 计算出的和集大小为 $O(\Delta s)$。因此，可以在以下时间内计算它： $O(\Delta s)$，利用~推论 21，并且可以在以下时间内由此推导出函数 h：$O(\Delta s)$。

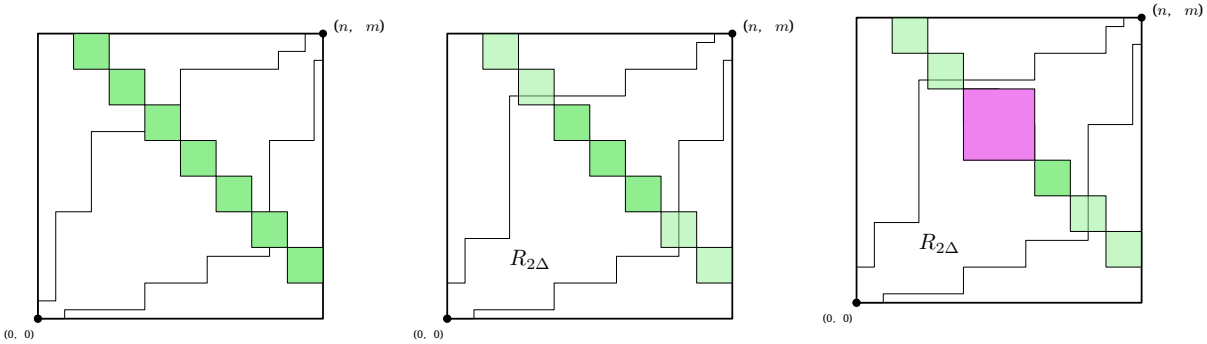
现在我们来界定所有子问题的总运行时间。固定一个边长为 s 的情形，并考虑所有边长为 s 的可能子问题，即所有盒子

$$B^s_{x,y} := [\, x \cdot s \,..\, x \cdot s + s - 1 \,] \times [\, y \cdot s \,..\, y \cdot s + s - 1 \,], \text{ where } x, y \in [\, n/s \,].$$

考虑这些盒子的一条对角线 $D_{s,\ d} := \{\, B^s_{x,x+d} \mid x \in [\, n/s \,] \,\}$，见 图 3a。注意，$D_{s,\ d}$ 中完全位于 P^+_Δ 上方的盒子对应于情形 1 子问题。$D_{s,\ d}$ 中完全位于其下方的盒子对应于情形 2 子问题。位于 $P^+_{2\Delta}$ 下方或其上，且位于 $P^-_{2\Delta}$ 上方或其上的盒子对应于情形 3 子问题。其余盒子与 $P^+_{2\Delta}$ 或 $P^-_{2\Delta}$ 相交，对应于情形 4。

注意，由于 $P^+_{2\Delta}, P^-_{2\Delta}$ 的单调性，$D_{s,d}$ 中至多有两个盒子与 $P^+_{2\Delta}$ 或 $P^-_{2\Delta}$ 相交，因此 $D_{s,d}$ 中至多有两个盒子会作为情形 4 子问题出现。因此，情形 4 在每条对角线上耗时 $O(1)$。我们接下来论证，在 $D_{s,d}$ 的盒子中，至多有两个会作为情形 3 子问题出现。事实上，如果这样的盒子至少有三个，那么中间盒子的父问题也会位于 $P^+_{2\Delta}$ 和 $P^-_{2\Delta}$ 之间，因此这个父问题本身已经是一个情形 3 子问题，参见 图 3b 和 图 3c。这样一来，中间盒子就不会被拆分，也不会~成为递归子问题。因此，对每条对角线 $D_{s,d}$，情形 3 对至多两个盒子分别耗时 $O(\Delta s)$。

接下来需要对所有边长 $1 \le s \le n$ 其中 $s = 2^\ell$ 是 2 的幂（回想一下，在每个递归层级中，我们都会将边长分成两个相等的部分），以及所有 $O(n/s)$ 条对角线d求和，从而得到总时间 $\sum_{\ell=1}^{\log n} O(n/2^\ell) \cdot \widetilde{O}(\Delta 2^\ell) = \widetilde{O}(\Delta n)$。注意，对 ℓ 求和只会额外增加~一个对数因子，而这已被 O-记号所隐藏。 □



(a) 盒子的对角线 $D_{s,\ d}$　(b) $R_{2\Delta}$ 内的三个盒子　(c) 父盒子已包含在 $R_{2\Delta}$ 中

引理 引理26。

参考文献

[ABB+21] Kyriakos Axiotis, Arturs Backurs, Karl Bringmann, Ce Jin, Vasileios Nakos, Christos Tzamos, and Hongxun Wu. 快速而简单的模数子集和。载于 SOSA，第 57–67 页。SIAM，2021。doi:10.1137/1.9781611976496.6. 1[AKM+87] Alok Aggarwal, Maria M. Klawe, Shlomo Moran, Peter W. Shor, and Robert E. Wilber. 矩阵搜索算法的几何应用。Algorithmica，2:195–208，1987。doi:10.1007/BF01840359. 1, 2[AT19] Kyriakos Axiotis and Christos Tzamos. 容量约束动态规划：更快的背包问题与图算法。载于 ICALP，第 132 卷 LIPIcs，第 19:1–19:13 页。Schloss Dagstuhl - Leibniz-Zentrum für I nformatik，2019。doi:10.4230/LIPIcs.ICALP.2019.19. 2[BC22] Karl Bringmann and Alejandro Cassis. 通过有界单调最小加卷积实现更快的背包算法。载于 ICALP，第 229 卷 LIPIcs，第 31:1–31:21 页。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，2022。doi:10.4230/LIPIcs.ICALP.2022.31. 2, 3[BCD+14] David Bremner, Timothy M. Chan, Erik D. Demaine, Jeff Erickson, Ferran Hurtado, John Iacono, Stefan Langerman, Mihai Patrascu, and Perouz Taslakian. 项链、卷积与 X+Y。Algorithmica，69(2):294–314，2014。doi:10.1007/s00453- 012-9734-3. 1[Bel57] Richard Bellman. 动态规划。普林斯顿大学出版社，普林斯顿，新泽西州，美国，1957。doi:10.2307/j.ctv1nxcw0f. 1, 2[BFN21] Karl Bringmann, Nick Fischer, and Vasileios Nakos. 稀疏非负卷积等价于稠密非负卷积。载于 STOC，第 1711–1724 页。ACM，2021。doi:10.1145/3406325.3451090. 13[BFN22] Karl Bringmann, Nick Fischer, and Vasileios Nakos. 稀疏非负卷积的确定性算法与拉斯维加斯算法。载于 SODA，第 3069–3090 页。SIAM，2022。doi:10.1137/1.9781611977073.119. 13[BHSS18] MohammadHossein Bateni, MohammadTaghi Hajiaghayi, Saeed Seddighin, and Cliff Stein. 通过卷积与预测实现快速背包算法。载于 STOC，第 1269–1282 页。ACM，2018。doi:10.1145/3188745.3188876. 1, 2, 3[BHWZ94] Michael R. Bussieck, Hannes Hassler, Gerhard J. Woeginger, and Uwe T. Zimmer- mann. 最大卷积问题的快速算法。Oper. Res. Lett.，15(3):133–141，1994。doi:10.1016/0167-6377(94)90048-5. 1[BN21] Karl Bringmann and Vasileios Nakos. 近似子集和与划分问题的精细化视角。载于 SODA，第 1797–1815 页。SIAM，2021。doi:10.1137/1.9781611976465.108. 3[Bri17] Karl Bringmann. 一种近线性伪多项式时间子集和算法。载于 SODA，第 1073–1084 页。SIAM，2017。doi:10.1137/1.9781611974782.69. 1, 3[BW21] Karl Bringmann and Philip Wellnitz. 稠密子集和的近线性时间算法。载于 SODA，第 1777–1796 页。SIAM，2021。doi:10.1137/1.9781611976465.107. 1[CDXZ22] Shucheng Chi, Ran Duan, Tianle Xie, and Tianyi Zhang. 单调实例的更快最小加乘积算法。载于 STOC，第 1529–1542 页。ACM，2022。doi:10.1145/3519935.3520057. 1, 2

[CH02] Richard Cole 和 Ramesh Hariharan。稀疏匹配和通配符匹配中的候选匹配验证。载于 STOC，第 592–601 页。ACM，2002。doi:10.1145/509907.509992。13[CH22] Timothy M. Chan 和 Qizheng He。关于找零问题及相关问题的更多研究。《计算机与系统科学杂志》，124:159–169，2022。doi:10.1016/j.jcss.2021.09.005。1, 3[Cha18] Timothy M. Chan。0-1 背包问题的近似方案。载于 SOSA，OASIcs 第 61 卷，第 5:1–5:12 页。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，2018。doi:10.4230/OASIcs.SOSA.2018.5。3[CL15] Timothy M. Chan 和 Moshe Lewenstein。通过加性组合学求解聚类整数 3SUM。载于 STOC，第 31–40 页。ACM，2015。doi:10.1145/2746539.2746568。1, 3[CMWW19] Marek Cygan、Marcin Mucha、Karol Wegrzycki 和 Michal Wlodarczyk。关于与 (min, +)-卷积等价的问题。《ACM 算法汇刊》，15(1):14:1–14:25，2019。doi:10.1145/3293465。1, 3[CW21] Timothy M. Chan 和 R. Ryan Williams。确定性全点对最短路径、正交向量等：快速去随机化 Razborov-Smolensky。《ACM 算法汇刊》，17(1):2:1–2:14，2021。doi:10.1145/3402926。1[DJM23] Mingyang Deng、Ce Jin 和 Xiao Mao。通过稠密子集和近似求解背包问题和划分问题。载于 SODA，第 2961–2979 页。SIAM，2023。doi:10.1137/1.9781611977554.ch113。3[DMZ23] Mingyang Deng、Xiao Mao 和 Ziqian Zhong。关于无界子集和相关问题：一种统一的组合方法。载于 SODA，第 2980–2990 页。SIAM，2023。doi:10.1137/1.9781611977554.ch114。1[DP09] Devdatt P. Dubhashi 和 Alessandro Panconesi。《随机算法分析中的集中不等式》。剑桥大学出版社，2009。doi:10.1017/CBO9780511581274。5[EW18] Friedrich Eisenbrand 和 Robert Weismantel。利用 Steinitz 引理得到整数规划的邻近性结果和更快算法。载于 SODA，第 808–816 页。SIAM，2018。doi:10.1137/1.9781611975031.52。1[GGdC20] Pascal Giorgi、Bruno Grenet 和 Armelle Perret du Cray。近乎最优的稀疏多项式乘法。载于 ISSAC，第 202–209 页。ACM，2020。doi:10.1145/3373207.3404026。13[Gra72] Ronald L. Graham。一种确定有限平面点集凸包的高效算法。《信息处理快报》，1(4):132–133，1972。doi:10.1016/0020-0190(72)90045-2。11[JR19] Klaus Jansen 和 Lars Rohwedder。关于整数规划与卷积。载于 ITCS，LIPIcs 第 124 卷，第 43:1–43:17 页。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，2019。doi:10.4230/LIPIcs.ITCS.2019.43。1, 3[JW19] Ce Jin 和 Hongxun Wu。一种简单的近线性伪多项式时间子集和随机算法。载于 SOSA，OASIcs 第 69 卷，第 17:1–17:6 页。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，2019。doi:10.4230/OASIcs.SOSA.2019.17。1

[Kle22] Kim-Manuel Klein。关于无界子集和问题及 Frobenius 问题的细粒度复杂性。载于 SODA，第 3567–3582 页。SIAM，2022。doi:10.1137/1.9781611977073.141。1[KP04] Hans Kellerer 和 Ulrich Pferschy。与背包问题 FPTAS 相关的改进动态规划。《组合优化杂志》，8(1):5–11，2004。doi:10.1023/B:JOCO.0000021934.29833.6b。2[KPP04] Hans Kellerer、Ulrich Pferschy 和 David Pisinger。《背包问题》。Springer，2004。doi:10.1007/978-3-540-24777-7。1, 10[KPS17] Marvin Künnemann、Ramamohan Paturi 和 Stefan Schneider。关于一维动态规划的细粒度复杂性。载于 ICALP，LIPIcs 第 80 卷，第 21:1–21:15 页。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，2017。doi:10.4230/LIPIcs.ICALP.2017.21。1, 3[KX19] Konstantinos Koiliaris 和 Chao Xu。更快的子集和伪多项式时间算法。《ACM 算法汇刊》，15(3):40:1–40:20，2019。doi:10.1145/3329863。3[Nak20] Vasileios Nakos。近乎最优的稀疏多项式乘法。《IEEE 信息论汇刊》，66(11):7231–7236，2020。doi:10.1109/TIT.2020.2989385。13[Pis99] David Pisinger。具有有界重量的背包问题的线性时间算法。《算法杂志》，33(1):1–14，1999。doi:10.1006/jagm.1999.1034。1, 2[PRW21] Adam Polak、Lars Rohwedder 和 Karol Wegrzycki。小物品背包问题与子集和。载于 ICALP，LIPIcs 第 198 卷，第 106:1–106:19 页。Schloss Dagstuhl - Leibniz-Zentrum für Informatik，2021。doi:10.4230/LIPIcs.ICALP.2021.106。2[Wil18] R. Ryan Williams。通过电路复杂性加速全点对最短路径计算。《SIAM 计算杂志》，47(5):1965–1985，2018。doi:10.1137/15M1024524。1