Das Subset Sum Problem

Lehren aus dem Generalized Birthday Problem:

- Lösungen mit spezieller Form sind oft leichter zu konstruieren.
- Existieren hinreichend viele Lösungen, dann existieren auch Lösungen spezieller Form.

Definition Subset Sum Problem

Gegeben: $a_1, \ldots, a_n, S \in \mathbb{N}$
Gesucht: $I \subseteq [n], |I| = \frac{n}{2}$ mit $\sum_{i \in I} a_i = S$

- Brute-Force enumeriert alle $I \subseteq [n]$ mit $|I| = \frac{n}{2}$.
- Laufzeit ist $\tilde{\mathcal{O}}(\binom{n}{n/2}) = \tilde{\mathcal{O}}(2^n)$.

Abschätzung für Binomialkoeffizienten

Lemma Stirling-Abschätzung

Für $0 \leq \alpha \leq 1$ gilt $\binom{n}{\alpha n} = \tilde{\Theta}(2^{H(\alpha)n})$,
wobei $H(\alpha) = -\alpha \log(\alpha) - (1-\alpha)\log(1-\alpha)$ die binäre Entropie ist.

Beweis:
Aus der Stirling-Formel $n! \sim \sqrt{2\pi n} \cdot (\frac{n}{e})^n$ folgt

$$\begin{aligned}\binom{n}{\alpha n} &= \frac{n!}{(\alpha n)!((1-\alpha)n)!} = \tilde{\Theta}\left(\frac{(\frac{n}{e})^n}{(\frac{\alpha n}{e})^{\alpha n}(\frac{(1-\alpha)n}{e})^{(1-\alpha)n}}\right) \\ &= \tilde{\Theta}\left(2^{(-\alpha\log\alpha-(1-\alpha)\log(1-\alpha))n}\right) = \tilde{\Theta}(2^{H(\alpha)n})\end{aligned}$$

Korollar

Für $0 \leq \alpha \leq \beta \leq 1$ gilt $\binom{\beta n}{\alpha n} = \binom{\beta n}{\alpha \frac{1}{\beta} \beta n} = \tilde{\Theta}(2^{H(\frac{\alpha}{\beta})\cdot\beta n})$.

MitM für Subset Sum

Idee: Schreibe $\sum_{i \in I} a_i = S$ in der Form

$$\sum_{i \in I_1} a_i = S - \sum_{i \in I_2} a_i \text{ mit } I = I_1 \cup I_2 \text{ und } |I_1| = |I_2| = \frac{n}{4}.$$

Algorithmus Meet-in-the-Middle für Subset Sum

EINGABE: $a_1, \ldots, a_n, S$

1. Permutiere $a_1, \ldots, a_n$.
2. Für alle $I_1 \subseteq [1, \frac{n}{2}]$ mit $|I_1| = \frac{n}{4}$
   1. Erzeuge Liste L mit Einträgen $(I_1, \sum_{i \in I_1} a_i)$.
3. Sortiere L nach der zweiten Komponente.
4. Für alle $I_2 \subseteq [\frac{n}{2} + 1, n]$ mit $|I_2| = \frac{n}{4}$
   1. Falls $S - \sum_{i \in I_2} a_i$ in 2. Komponente von L auftaucht: $I := I_1 \cup I_2$.
5. Falls keine Lösung gefunden, zurück zu Schritt 1.

AUSGABE: I mit $\sum_{i \in I} a_i$ und $|I| = \frac{n}{2}$.

Korrektheit und Komplexität

Korrektheit:

- Benötigen Permutation der a_i in Schritt 1, so dass $|I \cap [1, \frac{n}{2}]| = \frac{n}{4}$.
- Dies geschieht mit Ws

$$\frac{\binom{n/2}{n/4}^2}{\binom{n}{n/2}} = \tilde{\Omega}\left(\frac{2^{\frac{n}{2}} \cdot 2^{\frac{n}{2}}}{2^n}\right) = \tilde{\Omega}(1).$$

- D.h. nach $\mathrm{poly}(n)$ Iterationen erhalten wir eine Lösung.

Komplexität:

- Der Algorithmus benötigt Zeit und Platz $\tilde{\mathcal{O}}(\binom{n/2}{n/4}) = \tilde{\mathcal{O}}(2^{\frac{n}{2}})$.

Repräsentationstrick: Howgrave-Graham, Joux (2010)

Idee:

- Verwende modifiziertes Meet-in-the-Middle mit
$$\sum_{i \in I_1} a_i = S - \sum_{i \in I_2} a_i \text{ mit } I_1, I_2 \subseteq [1, n] \text{ und } |I_1| = |I_2| = \frac{n}{4}.$$
- D.h. I_1, I_2 werden nicht wie zuvor aus disjunkten Mengen gewählt.

Nachteile:

- Größe von L für $(i_1, \sum_{i \in I_1} a_i)$ ist $\binom{n}{n/4}$ statt $\binom{n/2}{n/4}$.
- Falls $I_1 \cap I_2 \neq \emptyset$ ist $\sum_{i \in I_1 \cup I_2} a_i$ keine Lösung.

Vorteil:

- Anzahl *Repräsentationen* einer Lösung $I = I_1 \cup I_2$ ist $R := \binom{n/2}{n/4}$.
- **Bsp**: $I = \{1, 2, 5, 6\} \subseteq [1, 8]$ kann z.B. als $I_1 = \{1, 2\}$ und $I_2 = \{5, 6\}$ oder als $I_1 = \{1, 5\}$ und $I_2 = \{2, 6\}$ dargestellt werden.

Ziel: Konstruiere $\frac{1}{R}$-Bruchteil von L mit *einer* Repräsentation.

- D.h. wir konstruieren eine Liste L' der Größe
$$\frac{\binom{n}{n/4}}{\binom{n/2}{n/4}} = \mathcal{O}(2^{(H(\frac{1}{4}) - \frac{1}{2})n}) = \mathcal{O}(2^{0.311n})$$
- Kann in Gesamtlaufzeit $\tilde{\mathcal{O}}(2^{0.337n})$ realisiert werden.

Lineare Codes

Definition $[n, k]$-Code

Ein linearer $[n, k]$-Code C ist ein k-dimensionaler Unterraum $C \subseteq \mathbb{F}_2^n$.

Anmerkungen:

- Jeder $[n, k]$-Code C besitzt eine Generatormatrix $G \in \mathbb{F}_2^{k \times n}$ mit
$$C = \{\mathbf{x}G \mid \mathbf{x} \in \mathbb{F}_2^k\}.$$
- Alternativ: Definiere C mittels Parity-Check Matrix $P \in \mathbb{F}_2^{(n-k) \times n}$
$$C = \{\mathbf{c} \in \mathbb{F}_2^n \mid P \cdot \mathbf{c}^t = \mathbf{0}\}.$$

Definition Syndrom

Sei $P \in \mathbb{F}_2^{(n-k) \times n}$ eine Parity-Check Matrix von C und $\mathbf{x} \in \mathbb{F}_2^n$. Dann heißt $s(\mathbf{x}) := P \cdot \mathbf{x}^t$ das Syndrom von $\mathbf{x}$.

Distanz

Korollar

Sei $\mathbf{x} = \mathbf{c} + \mathbf{e} \in \mathbb{F}_2^n$ mit $\mathbf{c} \in C$. Dann gilt $s(\mathbf{x}) = s(\mathbf{e})$.

- D.h. das Syndrom hängt nur von $\mathbf{e}$ ab, nicht vom Codewort $\mathbf{c}$.

Definition Distanz

Sei C ein $[n, k]$-Code. Wir definieren die *Distanz* von C als

$$d = \min_{\mathbf{c},\mathbf{c}' \in C, \mathbf{c} \neq \mathbf{c}'} \{\mathrm{wt}(\mathbf{c} + \mathbf{c}')\}.$$

Wir bezeichnen C auch als $[n, k, d]$-Code.

- Eindeutige Dekodierung von $\mathbf{x} = \mathbf{c} + \mathbf{e}$ möglich, sofern

$$\mathrm{wt}(\mathbf{e}) \leq \lfloor \tfrac{d-1}{2} \rfloor.$$

Syndrom-Dekodierung

Problem Syndrom-Dekodierung

Gegeben: $P \in \mathbb{F}_2^{(n-k)\times n}$, ω, $\mathbf{x} = \mathbf{c} + \mathbf{e} \in \mathbb{F}_2^n$ mit $\mathbf{c} \in C$ und $\mathrm{wt}(\mathbf{e}) = \omega$

Gesucht: $\mathbf{e} \in \mathbb{F}_2^n$

Anmerkungen:

- Syndrom-Dekodierung erlaubt die Dekodierung von $\mathbf{x}$ als
$$\mathbf{c} = \mathbf{x} + \mathbf{e}.$$
- Brute-Force enumeriert alle $\mathbf{e} \in \mathbb{F}_2^n$ mit $\mathrm{wt}(\mathbf{e}) = \omega$ in Zeit $\tilde{\mathcal{O}}(\binom{n}{\omega})$.
- Idee: Verkleinere Suchraum durch lineare Algebra.

Information Set Decoding (Prange 1962)

Algorithmus Information Set Decoding

EINGABE: $P \in \mathbb{F}_2^{(n-k)\times n}$, ω, $\mathbf{x} \in \mathbb{F}_2^n$

1. Permutiere Spalten von P, d.h. für eine Permutationsmatrix $U_P \in \mathbb{F}_2^{n\times n}$ berechne $P' := P \cdot U_P$.
2. Erzeuge Einheitsmatrix in rechten Spalten, d.h. für ein invertierbares $U_G \in \mathbb{F}_2^{(n-k)\times(n-k)}$ berechne
$$P_s := U_G \cdot P' \text{ mit } P_s = (H|I_{n-k}) \text{ und } s(\mathbf{x}) := U_G \cdot P\mathbf{x}^t.$$
3. Wähle p geeignet.
4. Für jedes $\mathbf{e}_1 \in \mathbb{F}_2^k$ mit $\mathrm{wt}(\mathbf{e}_1) = p$: Berechne $\mathbf{e}_2^t := H \cdot \mathbf{e}_1^t + s(\mathbf{x})$. Falls $\mathrm{wt}(\mathbf{e}_2) = \omega - p$, setze $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2) \cdot U_p^{-1}$
5. Falls keine Lösung e gefunden wurde, zurück zu Schritt 1.

AUSGABE: $\mathbf{e}$

Korrektheit und Laufzeit von ISD

Korrektheit:

- Schritt 1 permutiert die Koordinaten von $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$.
- Benötigen in Schritt 4, dass $\mathbf{e}_1 \in \mathbb{F}_2^k$ Gewicht $\mathrm{wt}(\mathbf{e}_1) = p$ besitzt.
- Dies geschieht mit Wahrscheinlichkeit
$$p_1 := \frac{\binom{k}{p}\binom{n-k}{\omega-p}}{\binom{n}{\omega}}.$$
- Es gilt $P_s \cdot \mathbf{e}^t = s(\mathbf{x})$ mit $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2)$ und damit
$$H \cdot \mathbf{e}_1^t + I_{n-k} \cdot \mathbf{e}_2^t = U_G \cdot s(\mathbf{x}) \text{ bzw. } \mathbf{e}_2^t = H \cdot \mathbf{e_1} + s(\mathbf{x}).$$

Laufzeit:

- Pro Iteration benötigen wir in Schritt 4 $\tilde{\mathcal{O}}(\binom{k}{p})$, d.h. insgesamt
$$\tilde{\mathcal{O}}\left(\binom{k}{p} \cdot p_1^{-1}\right) = \tilde{\mathcal{O}}\left(\frac{\binom{n}{\omega}}{\binom{n-k}{\omega-p}}\right).$$
- Wird minimiert für $p = 0$, d.h. wir erhalten $\tilde{\mathcal{O}}\left(\frac{\binom{n}{\omega}}{\binom{n-k}{\omega}}\right)$.
- Dies verbessert den Brute-Force Ansatz um den Faktor $\binom{n-k}{\omega}$.
- Laufzeit kann abgeschätzt werden durch $\mathcal{O}(2^{0.058n})$.
(mittels der sogenannten Gilbert-Varshamov Schranke)

Sterns Information Set Decoding (1989)

Idee: Modifiziere Pranges Algorithmus wie folgt.

- Verwende Meet-in-the-Middle statt Brute Force in Schritt 4.
- Permutiere dazu $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) \in \mathbb{F}_2^{\frac{k}{2} \times \frac{k}{2} \times (n-k)}$, so dass
$$\mathrm{wt}(\mathbf{e}_1) = \mathrm{wt}(\mathbf{e}_2) = \frac{p}{2} \text{ und } \mathrm{wt}(\mathbf{e}_3) = \omega - p.$$
- Schreibe $H \in \mathbb{F}_2^{(n-k) \times k}$ als $H = (H_1 | H_2)$ mit $H_1 = H_2 = \mathbb{F}_2^{(n-k) \times \frac{k}{2}}$.
- Matche $H_1 \cdot \mathbf{e}_1^t = H_2 \cdot \mathbf{e}_2^t + s(\mathbf{x})$ auf ℓ Koordinaten exakt.
- Für alle Lösungen $(\mathbf{e}_1, \mathbf{e}_2)$ berechne $\mathbf{e}_3 = H \cdot (\mathbf{e}_1, \mathbf{e}_2)^t + s(\mathbf{x})$.
- Prüfe $\mathrm{wt}(\mathbf{e}_3) \stackrel{?}{=} \omega - p$.
- Optimierung von p, ℓ liefert eine Laufzeitschranke von $\tilde{\mathcal{O}}(2^{0.056n})$.
- Der beste bekannte Algorithmus (BJMM 2012) nutzt zusätzlich den Repräsentationstrick und erreicht $\tilde{\mathcal{O}}(2^{0.049n})$.
- Parameterwahl McEliece: Empfehlung von Codelängen
$$n = \frac{80}{0.049} > 1600.$$