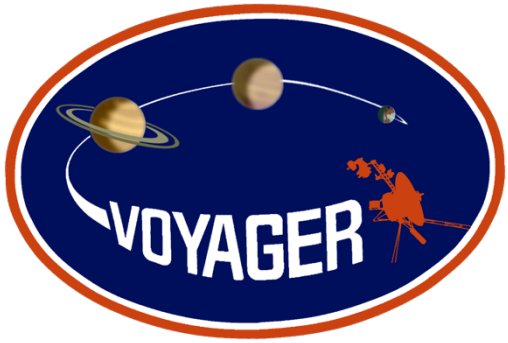
Einführung in die Reed-Solomon-Codierung



Inhaltsverzeichnis

1 Einleitung

Stell dir vor, du könntest eine Sprache sprechen, die fast jede Technologie versteht – von deinem Smartphone bis zu den Raumsonden, die durchs All fliegen. Genau das macht die Reed-Solomon-Codierung möglich. Sie ist wie ein Superheld in der Welt der Daten, der dafür sorgt, dass Nachrichten immer klar und deutlich ankommen, selbst wenn sie durch Stürme oder gegen Interferenzen ankämpfen müssen.

Die Reed-Solomon-Kodierung wurde erstmals in der Voyager-Mission eingesetzt, weil sie eine effiziente und robuste Methode zur Fehlerkorrektur bietet, die für die Datenübertragung über extrem lange Distanzen im Weltraum unerlässlich ist. Die Voyager-Raumsonden wurden mit der Aufgabe gestartet, Daten und Bilder von den äusseren Planeten unseres Sonnensystems zurück zur Erde zu senden. Aufgrund der enormen Entfernungen sind die Signale, die die Raumsonden zurück zur Erde senden, sehr schwach und anfällig für Störungen und Rauschen.

Wenn du dich mit der Reed-Solomon-Codierung beschäftigst, lernst du nicht nur, wie man coole Sachen wie CDs, DVDs und Blu-rays zum Laufen bringt, sondern auch, wie du deine eigenen Ideen vor Datenverlust schützen kannst. Es ist ein bisschen wie Zaubertricks für Daten – du sorgst dafür, dass sie sicher von einem Ort zum anderen kommen, ohne dass etwas verloren geht.

Und das Beste? Es ist total spannend zu sehen, wie deine Fähigkeiten direkt die Welt beeinflussen können. Jeder QR-Code, den du scannst, und jede Netflix-Serie, die du ohne Probleme streamst, nutzt diese Technik. Indem du lernst, wie du mit Reed-Solomon-Codes umgehst, wirst du zu dem Magier hinter der Technik, die jeder täglich nutzt.

Also, pack deine Neugier aus und tauche ein in die Welt der Fehlerkorrektur. Es ist nicht nur super nützlich, sondern auch ein Riesenspass, die Geheimnisse zu lüften, die hinter den Kulissen unserer vernetzten Welt arbeiten. Wer weiss, vielleicht entwickelst du die nächste grosse Technologie, die auf Reed-Solomon-Codierung basiert!

Lass uns gemeinsam die Datenwelt ein wenig besser und sicherer machen. Es ist deine Chance, die Superheldin der Technologie von morgen zu werden!

2 Einordnung der mathematischen Hintergründe

Die folgende Auflistung bietet einen ersten Überblick über die mathematischen Grundlagen, die für das Verständnis und die Anwendung von Reed-Solomon-Codes notwendig sind. Es ist wichtig zu betonen, dass dies lediglich eine kurze Einführung in die Theorie hinter diesen mächtigen Werkzeugen der Fehlerkorrektur darstellt. Der Ansatz ist dabei sehr praxisorientiert, um nicht nur die abstrakten Konzepte, sondern auch deren konkrete Anwendung in realen Szenarien zu beleuchten. Wir werden sehen, wie diese mathematischen Prinzipien in die Entwicklung von Technologien einfliessen, die unseren Alltag durchdringen – von der sicheren Übertragung von Nachrichten bis hin zum fehlerfreien Lesen von Speichermedien. Letztlich ist das Ziel, ein solides Verständnis zu entwickeln, das es ermöglicht, die Reed-Solomon-Codierung effektiv für praktische Herausforderungen einzusetzen

- **Modulare Arithmetik**: Diese Art der Arithmetik beschäftigt sich mit Zahlen innerhalb eines festgelegten Bereichs (Modulus). Bei der Berechnung von Reed-Solomon-Codes wird modulare Arithmetik verwendet, um mit Zahlen innerhalb eines endlichen Feldes zu arbeiten, was bedeutet, dass alle Operationen "umhüllen", wenn sie einen bestimmten Wert (den Modulus) erreichen.
- **Endliche Körper (Galois-Felder)**: Ein endlicher Körper ist ein mathematisches System, in dem eine endliche Anzahl von Elementen existiert, auf denen Addition, Subtraktion, Multiplikation und Division (ausser durch null) durchgeführt werden können. Reed-Solomon-Codes verwenden spezielle Arten von endlichen Körpern, sogenannte Galois-Felder, für die Codierung und Decodierung, da sie gut definierte und geschlossene Operationen für die Fehlerkorrektur bieten.
- **Euklidischer Algorithmus**: Ein Verfahren zur Berechnung des grössten gemeinsamen Teilers (ggT) von zwei Zahlen. In Bezug auf Reed-Solomon-Codes wird eine Variante des euklidischen Algorithmus verwendet, um das Fehlerlokalisierungspolynom zu finden, welches essenziell ist, um Fehler in den Daten zu identifizieren und zu korrigieren.
- **Kleiner Satz von Fermat**: Dieses Theorem besagt, dass wenn p eine Primzahl ist, dann wird jede ganze Zahl a, die nicht durch p teilbar ist, die Gleichung $a^{p-1} \equiv 1 \bmod p$ erfüllen. Dieses Theorem hilft beim Verständnis der Struktur von endlichen Feldern, die in Reed-Solomon-Codes verwendet werden.
- **Polynominterpolation**: Die Polynominterpolation ist ein Prozess, bei dem ein Polynom konstruiert wird, das durch eine gegebene Menge von Punkten geht. Für Reed-Solomon-Codes ist dies wichtig, um das ursprüngliche Nachrichtenpolynom aus einer begrenzten Anzahl von Punkten (den codierten Datenpunkten) zu rekonstruieren.
- **Lineare Gleichungssysteme**: Dies sind Sammlungen von linearen Gleichungen, die gemeinsame Lösungen haben. In der Reed-Solomon-Codierung können lineare Gleichungssysteme auftreten, wenn man versucht, Fehler zu korrigieren und die ursprünglichen Daten wiederherzustellen.
- **Diskrete Fouriertransformation**: Dies ist eine Methode zur Umwandlung einer Sequenz von Werten in eine andere, die ihre Frequenzkomponenten zeigt. Bei Reed-Solomon-Codes kann eine spezielle Form, die diskrete Fouriertransformation auf einem endlichen Körper, genutzt werden, um die Codierung und Decodierung von Daten effizient durchzuführen.

Im Folgenden konzentrieren wir uns auf das, was als fundamentale Ebene des Reed-Solomon-Codes betrachtet werden kann: die Codierung von Informationen in ein Polynom.

Die zugrundeliegenden mathematischen Prinzipien sind entscheidend für die Durchführung der erforderlichen Operationen auf eine effiziente Weise. Im Kontext der Informatik bedeutet dies, dass sowohl die benötigte Rechenzeit als auch der Speicherbedarf minimal sein sollten.

Bevor wir uns jedoch detaillierter mit den Polynomen und den spezifischen Aspekten des Reed-Solomon-Codes auseinandersetzen, ist es sinnvoll, einige grundlegende Überlegungen zur Datenübertragung anzustellen.

3 Fehlerkorrektur bei Datenübertragung im Allgemeinen

Beim Erarbeiten des Kapitel 4 des Buches "Informatik, Data Science und Sicherheit" hast du die Grundlagen selbstkorrigierender Codes und den Karten-Trick kennengelernt. Nutze dieses Wissen und das Verständnis das du entwickelt hast, um die nachfolgenden Aufgaben zu lösen.

Ein einfacher Ansatz zur Fehlerkorrektur könnte darin bestehen, die Daten zweifach zu übertragen?

Optional: Um deine Überlegungen zu den Aufgaben eins bis vier zu verdeutlichen, erstelle grafische Darstellungen und nimm einen Bildschirmrekord oder eine Videopräsentation auf. Du kannst diese Aufgabe auch als Partnerarbeit gestalten, indem du mit jemandem ein Gespräch führst und dabei deine Gedanken grafisch festhältst.

3.1 Aufgabe 1

Würde das funktionieren? Könnten Fehler korrigiert werden?

Lösung: Die Übertragung der Daten in doppelter Ausführung stellt eine Methode der Fehlererkennung dar. Wenn beide Datensätze verglichen werden und vollständig übereinstimmen, kann man davon ausgehen, dass keine einzelner Fehler aufgetreten ist. Falls es Unterschiede zwischen den beiden Sätzen gibt, weiss man, dass ein Fehler vorliegen muss.

Allerdings ermöglicht diese Methode für sich genommen keine Fehlerkorrektur, sondern lediglich Fehlererkennung. Um tatsächlich Fehler zu korrigieren, müsste man wissen, welcher der beiden Datensätze korrekt ist. Bei nur zwei identischen Übertragungen gibt es keine Möglichkeit zu bestimmen, welcher Satz fehlerhaft ist, wenn sie sich unterscheiden. Zudem wäre es möglich, dass zwei zwar gleiche, aber falsche Datensätze übertragen werden.

3.2 Aufgabe 2

Würde eine dreifache Übertragung der Daten im Vergleich zur doppelten Übertragung eine die Fehlerkorrektur ermöglichen? Falls ja siehst du trotzdem Nachteile?

Eine dreifache Übertragung der Daten verbessert tatsächlich die Möglichkeit zur Fehlerkorrektur im Vergleich zur doppelten Übertragung. Bei dreifacher Übertragung kann ein einfaches Mehrheitsprinzip angewandt werden: Empfängt man drei identische Datensätze, kann man mit hoher Wahrscheinlichkeit davon ausgehen, dass keine Fehler vorliegen. Treten jedoch Unterschiede auf, so lässt sich der korrekte Datenwert anhand der Mehrheit bestimmen – das heisst, der Wert, der mindestens zweimal auftritt, wird als korrekt angenommen. Dies alles aber nur, wenn nur genau ein Fehler auftritt.

Dies ermöglicht eine grundlegende Form der Fehlerkorrektur, da man sich für den Datenwert entscheidet, der am häufigsten empfangen wurde. Zudem ist sie sehr ineffizient, da sie die Datenmenge verdreifacht, was den Übertragungsaufwand signifikant erhöht.

Man muss 9 Zahlen versenden, um 3 Zahlen gegen 1 Fehler abzusichern.

Zudem: Durch die Verdreifachung der Nachrichtenlänge steigt auch die Wahrscheinlichkeit für das Auftreten mehrerer Fehler. Daher ergibt es keinen signifikanten Vorteil, wenn man durch die Verdreifachung der Nachrichtenlänge lediglich einen Fehler korrigieren kann.

Bei persistenten Störquellen oder defekter Hardware besteht das Risiko, dass Fehler reproduzierbar sind und somit bei jeder Übertragung identisch auftreten. Wenn die Fehler bei einer mehrfachen Datenübertragung identisch sind, also bei jeder Übertragung genau dieselben Daten falsch übermittelt werden, kann das Mehrheitsprinzip versagen. Bei einer dreifachen Übertragung würde dies bedeuten, dass der fehlerhafte Wert in der Mehrheit ist und daher fälschlicherweise als korrekt angenommen wird.

3.3 Aufgabe 3

Würde eine vierfache Übertragung der Daten die Situation im Hinblick auf das Vorkommen mehrerer Fehler verbessern?

Selbst eine vierfache Datenübertragung würde das Problem nicht lösen, da bei zwei Fehlern innerhalb von vier Zahlen nicht unterschieden werden kann, ob ursprünglich eine 2 oder eine 5 gesendet wurde. In einem solchen Szenario offenbart sich die Grenze der Effizienz bei der einfachen Mehrfachübertragung. Daher ist diese Methode nicht die bevorzugte Wahl für Fehlerkorrektur.

Bessere Alternativen bietet die Reed-Solomon-Codierung, deren Konzepte im Folgenden vorgestellt werden.

4 Reed-Solomon-Codierung

Die zentrale Idee besteht darin, Informationen als Zahlenfolgen zu übermitteln – nehmen wir das Beispiel 3, 2, 8.

Diese Zahlen werden als Koeffizienten in ein Polynom eingebettet, was zu

$$3x^2+2x+8$$

führt.

Anschliessen werden sieben verschiedene x Werte in dieses Polynom eingesetzt. Der Empfänger kennt die verwendeten x-Werte und kann so die Polynomwerte ausrechnen. Auch der Grad des Polynoms ist bekannt.

Also zum Beispiel wurde abgemacht, dass die ersten natürlichen Zahlen eingesetzt werden. Also stellt sich die Frage: welche y-Werte gehören zu den x-Werten 1 bis und mit 7.

Die Anzahl der benötigten Werte bestimmt sich anhand des Grades der Polynome bzw. der Anzahl Zahlen, die übertragen werden sollen. Dazu nach der nächsten Aufgabe mehr.

4.1 Aufgabe 4

Stelle ein Polynom in Python grafisch dar und markiere spezifische Punkte dieses Polynoms auf dem Graphen.

1. Definiere das Polynom: Verwende das Polynom $3x^2+2x+8$. Schreibe Code, um die Koeffizienten dieses Polynoms in Python zu definieren.
2. Erzeuge x-Werte: Erzeuge eine Reihe von x-Werten, die von 1 bis 7 reichen. Verwende dafür die Funktion np.linspace aus der NumPy-Bibliothek.
3. Berechne y-Werte: Berechne die y-Werte des Polynoms für jeden x-Wert. Nutze die Funktion np.polyval, um die y-Werte basierend auf den Koeffizienten des Polynoms zu finden.
4. Zeichne den Graphen: Verwende Matplotlib, um einen Graphen des Polynoms zu zeichnen. Stelle sicher, dass der Graph Titel und Achsenbeschriftungen hat.
5. Markiere die Punkte: Markiere jeden Punkt (x/y) auf dem Graphen mit einer auffälligen Farbe.
6. Gib die Punkte aus: Gib alle Punkte (x/y) in der Konsole aus. Du kannst dafür z.B. zip() nutzen.

Tipps:

- Vergiss nicht, die Bibliotheken NumPy und Matplotlib zu importieren.
- Nutze eine Schleife, um die Punkte auszugeben.

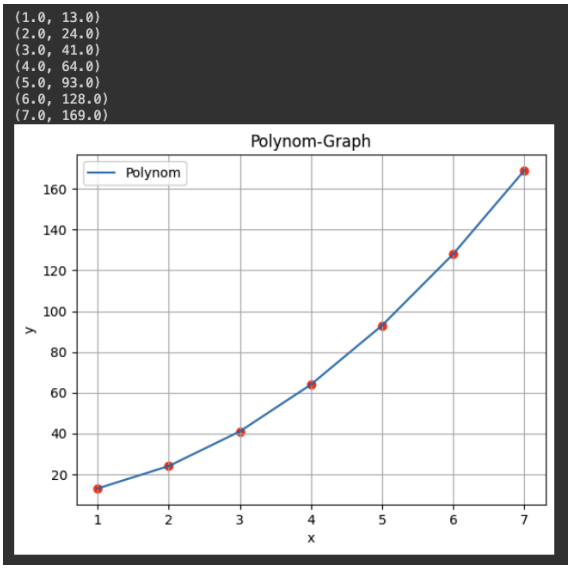
Zusätzliche Herausforderung (optional):

Füge eine Legende zum Graphen hinzu, die erklärt, was die Linie und die markierten Punkte darstellen.

Lösung:

```python
import numpy as np
import matplotlib.pyplot as plt

# Definiere die Koeffizienten des Polynoms
# Zum Beispiel für das Polynom 3x^2 + 2x + 8
koeffizienten = [3, 2, 8]

# Erzeuge die x-Werte
x = np.linspace(1, 7, 7)

# Berechnen die y-Werte des Polynoms
y = np.polyval(koeffizienten, x)

# Ausgabe der Punkte
for xi, yi in zip(x, y):
    print(f'({xi}, {yi})')

# Zeichnen des Graphen
plt.plot(x, y, label='Polynom')
plt.scatter(x, y, color='red')  # Punkte auf dem Graphen markieren

# Titel und Labels hinzufügen
plt.title('Polynom-Graph')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.legend()
plt.show()
```

Wenn der Code richtig umgesetzt wurde, wirst du folgendes Ergebnis sehen:



Dem Empfänger werden also die Zahlen 13, 24, 41, 64, 93, 128, und 169 gesendet. Der Empfänger erhält sieben rote Punkte, und seine Aufgabe besteht darin, aus diesen Punkten das ursprüngliche Polynom zu rekonstruieren. Wenn keine Fehler aufgetreten sind, ist dieser Prozess relativ einfach. Der Empfänger weiss, dass das Polynom vom zweiten Grad ist und daher für die Polynominterpolation lediglich drei Punkte benötigt werden. Somit kann er, vorausgesetzt es liegen keine Fehler vor, einfach drei der sieben Punkte auswählen, um das Polynom zweiten Grades zu rekonstruieren.

Um ein Polynom aus drei gegebenen Punkten zu berechnen, können wir ein lineares Gleichungssystem verwenden. Wir wählen ein Polynom mit einem Grad von höchstens 2 (da wir drei Punkte haben und ein Polynom von Grad n durch n+1 Punkte bestimmt).

$$P_1(x_1, y_1), P_2(x_2, y_2) \text{ und } P_3(x_3, y_3)$$

Das allgemeine quadratische Polynom lautet:

$$P(x) = ax^2 + bx + c$$

Um das Polynom zu bestimmen, setzen wir die Koordinaten der Punkte in das Polynom ein:

$$P_1(x_1) = ax_1^2 + bx_1 + c = y_1$$

$$P_2(x_2) = ax_2^2 + bx_2 + c = y_2$$

$$P_3(x_3) = ax_3^2 + bx_3 + c = y_3$$

Dies führt zu einem linearen Gleichungssystem mit drei Gleichungen und den Unbekannten *a, b und c.*

Das Gleichungssystem lautet:

$$\begin{aligned} ax_1^2 + bx_1 + c &= y_1 \\ ax_2^2 + bx_2 + c &= y_2 \\ ax_3^2 + bx_3 + c &= y_3 \end{aligned}$$

Durch Lösen dieses Gleichungssystems erhalten wir die Koeffizienten des Polynoms P(x), und somit können wir das Polynom vollständig bestimmen.

Angenommen, wir haben die Punkte:

$$P_1(1,2), (\ P_2(2,3) \text{ und } (\ P_3(3,5)$$

Das Gleichungssystem lautet:

$$\begin{aligned} I&: a * 1^2 + b * 1 + c = 2 \\ II&: a * 2^2 + b * 2 + c = 3 \\ III&: a * 3^2 + b * 3 + c = 5 \end{aligned}$$

Nun nehmen wir schrittweise eine Variable nach der anderen und berechnen dadurch die Werte für a, b und c. Dabei können verschiedene Wege gegangen werden. Wir beginnen damit Gleichung I nach c aufzulösen und in Gleichung II und III einzusetzen.

$$\begin{aligned} I&: c = 2 - a - b \\ II&: 4a + 2b + 2 - a - b \\ III&: 9a + 3b + 2 - a - b \end{aligned}$$

Anschliessend lösen wir Gleichung II nach der Variablen b auf und setzen dieses Ergebnis in Gleichung III ein.

$$\begin{aligned} II&: b = 3 - 3a - 2 \\ III&: 8a + 2(3 - 3a - 2) + 2 \end{aligned}$$

In Gleichung III haben wir nun nur noch eine Variable, die wir durch Lösen der Gleichung bestimmen können.

$$a = \frac{1}{2}$$

Jetzt können wir a einsetzen, und unser Gleichungssystem sieht folgendermassen aus:

$$\begin{aligned} I&: \frac{1}{2} + b + c = 2 \\ II&: 2 + 2b + c = 3 \\ III&: \frac{9}{2} + 3b + c = 5 \end{aligned}$$

Jetzt lösen wir Gleichung I nach b auf und setzen sie in Gleichung II ein.

$$\begin{aligned} I&: b = \frac{3}{2} - c \\ II&: 2 + 2\left(\frac{3}{2} - c\right) + c = 3 \end{aligned}$$

In Gleichung II haben wir nun nur noch eine Variable, die wir durch Lösen der Gleichung bestimmen können.

$$2 = c$$

Da a und c jetzt bekannt sind, setzen wir sie in Gleichung III ein und bestimmen so den Wert von b.

$$\frac{9}{2} + 3b + 2 = 5$$

$$b = -\frac{1}{2}$$

Der Vollständigkeit halber setzen wir a b und c in das allgemeine Polynom ein und haben so das Polynom. Im Fall der Reed-Solomon-Codierung kann überprüft werden, ob die anderen Punkte auf dem Polynom liegen, um sicherzustellen, dass die berechneten Koeffizienten mit denen übereinstimmen, die der Absender kodiert hatte. Dies geschieht, indem die anderen Punkte in das Polynom eingesetzt werden, und wenn sie auf dem Polynom liegen, bestätigt dies die Richtigkeit der Koeffizienten.

$$\mathrm{P(x)} = 0.5\,\mathrm{x}^2 - 0.5\mathrm{x} + 2$$

4.2 Aufgabe 5

Berechne zuerst das Polynom mit den gegebenen Punkten auf Papier und zeige deinen Lösungsweg. Erstelle dann anhand der unten aufgeführten Schritte Python-Code.

Erstelle den Code, um eine Polynom-Interpolation für eine gegebene Menge von Datenpunkten durchzuführen und das Ergebnis grafisch darzustellen. Das interpolierte Polynom soll in seiner algebraischen Form angezeigt werden.

1. **Bereite die Daten vor**: Gegeben sind die x-Werte [1, 2, 3, 4, 5, 6, 7] und die entsprechenden y-Werte [13, 24, 41, 64, 93, 128, 169]. Definiere diese Werte in deinem Python-Programm.
2. **Wähle den Grad des Polynoms**: Für dieses Beispiel sollst du eine Polynom-Interpolation mit einem Polynom 2. Grades durchführen.
3. **Berechne die Koeffizienten des Polynoms**: Verwende np.polyfit, um die Koeffizienten des Polynoms zu berechnen, das am besten zu deinen Daten passt.
4. **Erstelle das Polynom**: Nachdem du die Koeffizienten berechnet hast, erstelle das Polynom mit np.poly1d und gib es aus.
5. **Zeichne das Polynom und die Datenpunkte**: Erzeuge eine Reihe von x-Werten, die für die Zeichnung des Polynoms verwendet werden. Zeichne dann das Polynom und markiere die ursprünglichen Datenpunkte auf demselben Graphen mit mathplotlib.
6. **Gestalte den Graphen**: Füge dem Graphen einen Titel sowie Beschriftungen für die x- und y-Achse hinzu und aktiviere das Gitternetz für bessere Lesbarkeit.

Lösung:

$$\text{I: } a+b+c = 13$$
$$\equiv |$$
$$\text{II: } 4a+2b+c = 24$$
$$\text{III } 9a+3b+c = 41$$

$$\text{I: } c = 13-a-b$$
$$\text{II: } 4a+2b+13-a-b = 24$$
$$\text{III: } 9a+3b+13-a-b = 41$$

$$\text{II: } b = 11-3a$$
$$\text{III: } 8a+2(11-3a)+13 = 41$$

$$a = 3$$

$$\text{I: } 3+b+c = 13$$
$$\text{II: } 12+2b+c = 24$$
$$\text{III } 27+3b+c = 41$$

$$b = 10-c$$
$$12+2(10-c)+c = 24$$
$$c = 8$$

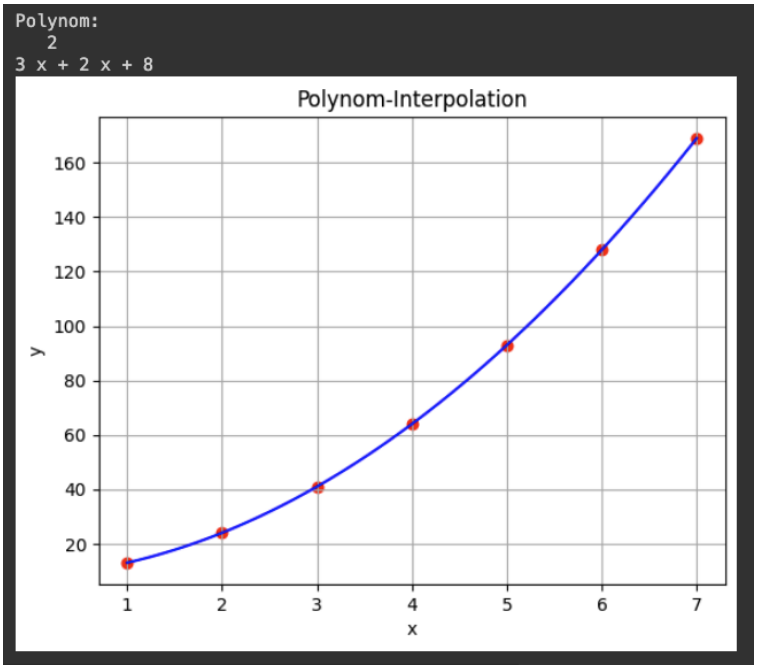
$$27+3b+8 = 41$$
$$b = 2$$

$$P(x) = 3x^2+2x+8$$

Lösung Python:

```python
import numpy as np
import matplotlib.pyplot as plt

# Gegebene Datenpunkte
x_punkte = np.array([1, 2, 3, 4, 5, 6, 7])
y_punkte = np.array([13, 24, 41, 64, 93, 128, 169])

# Polynom-Grad für die Interpolation
grad = 2

# Berechnen der Polynomkoeffizienten für die Interpolation
koeffizienten = np.polyfit(x_punkte, y_punkte, grad)

# Erstellen eines Polynoms mit diesen Koeffizienten
polynom = np.poly1d(koeffizienten)

# Ausgeben des Polynoms
print("Polynom:")
print(polynom)

# Erzeugen von x-Werten für die Zeichnung des Polynoms
x_linie = np.linspace(min(x_punkte), max(x_punkte), 100)

# Berechnen von y-Werten basierend auf dem Polynom
y_linie = polynom(x_linie)

# Zeichnen der ursprünglichen Datenpunkte
plt.scatter(x_punkte, y_punkte, color='red')

# Zeichnen des Polynoms
plt.plot(x_linie, y_linie, color='blue')

# Titel und Labels hinzufügen
plt.title('Polynom-Interpolation')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.show()
```

Wenn der Code richtig umgesetzt wurde, wirst du folgendes Ergebnis sehen:



Nach der erfolgreichen Rekonstruktion des Polynoms sind auch die Koeffizienten bekannt, womit klar ist, welche Nachricht übermittelt wurde. Nun stellen wir uns die Frage, was geschieht, wenn Fehler auftreten.

Angenommen, es treten zwei Fehler auf. In diesem Fall ist es möglich, dass die beiden fehlerhaften Punkte zusammen mit einigen der korrekten Punkte ein alternatives Polynom bilden.

Um das etwas zu veranschaulichen, finde ein Polynom zweiten Grades, dass das gegebene Polynom $3x^2+2x+8$ an den Stellen x=2 und x=3 schneidet.

1. Berechne die y-Werte von p(x) an den Stellen x=2 und x=3
2. Setze ein allgemeines Polynom zweiten Grades an. ($q(x) = ax^2 +bx+c$)
3. Verwende die unter 1. gefundenen Punkte, um ein Gleichungssystem mit zwei Gleichungen aufzustellen.
4. Für eine eindeutige Lösung brauchen wir eine 3. Bedingung darum setzen wir a = 1 das heisst, wir legen den führenden Koeffizienten des Polynoms fest. Wir wissen also jetzt schon das Polynom wird mit x^2 beginnen.
5. Jetzt kann das Gleichungssystem gelöst werden.

4.3 Aufgabe 6

Setze dieses Vorgehen in Python Code um. Benutze dazu np.linalg.solve().
Lass zusätzlich zum neuen Polynom die Koeffizienten b und c ungerundet ausgeben, um ein Gefühl dafür zu bekommen, wie stark gerundet wurde.

Lösung:

```python
import numpy as np

# Definiere das gegebene Polynom
def p(x):
    return 3*x**2 + 2*x + 8

# Berechne die y-Werte an den Stellen x=2 und x=3
y2 = p(2)
y3 = p(3)

# Setze a = 1

# Erstelle das Gleichungssystem
# 2b + c = y2 - 4
# 3b + c = y3 - 9

A = np.array([[2, 1], [3, 1]])
B = np.array([y2 - 4, y3 - 9])

# Löse das Gleichungssystem
b, c = np.linalg.solve(A, B)

# Konvertiere die Lösungen in Integer
b_int = int(round(b))
c_int = int(round(c))

# Ausgabe des neuen Polynoms
print(f"Das neue Polynom ist q(x) = x^2 + {b_int}x + {c_int}")
print(f"b ungerundet {b}")
print(f"c ungerundet {c}")
```

Wenn der Code richtig umgesetzt wurde, wirst du folgende Ausgabe sehen:

```
Das neue Polynom ist q(x) = x^2 + 12x + -4
b ungerundet 11.999999999999998
c ungerundet -3.999999999999996
```

4.4 Aufgabe 7

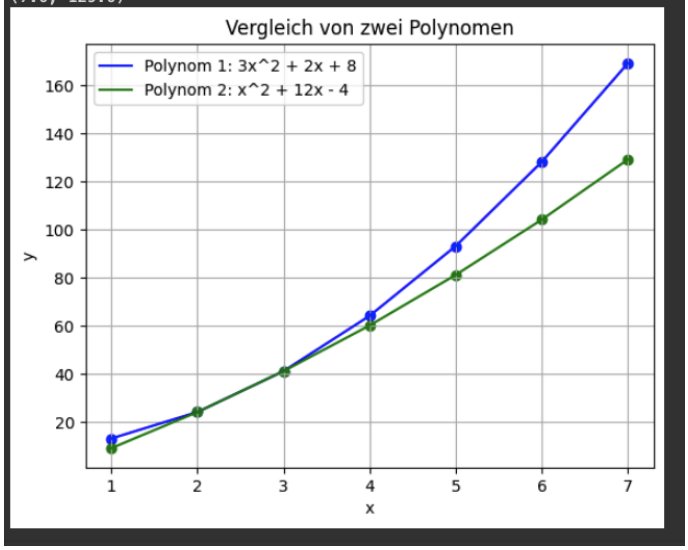
Passe den Code aus Aufgabe 4 so an, dass beide Polynome und die Punkte ausgegeben werden.

Lösung:

```
import numpy as np
import matplotlib.pyplot as plt

# Definiere die Koeffizienten des ersten Polynoms (zum Beispiel
3x^2 + 2x + 8)
koeffizienten1 = [3, 2, 8]

# Definiere die Koeffizienten des zweiten Polynoms (zum Beispiel
x^2 + 12x - 4)
koeffizienten2 = [1, 12, -4]

# Erzeuge die x-Werte
x = np.linspace(1, 7, 7)

# Berechne die y-Werte des ersten Polynoms
y1 = np.polyval(koeffizienten1, x)

# Berechne die y-Werte des zweiten Polynoms
y2 = np.polyval(koeffizienten2, x)

# Ausgabe der Punkte für das erste Polynom
print("Punkte des ersten Polynoms (3x^2 + 2x + 8):")
for xi, yi in zip(x, y1):
    print(f'({xi}, {yi})')

# Ausgabe der Punkte für das zweite Polynom
print("\nPunkte des zweiten Polynoms (x^2 + 12x - 4):")
for xi, yi in zip(x, y2):
    print(f'({xi}, {yi})')

# Zeichne das erste Polynom
plt.plot(x, y1, label='Polynom 1: 3x^2 + 2x + 8', color='blue')
plt.scatter(x, y1, color='blue')

# Zeichne das zweite Polynom
plt.plot(x, y2, label='Polynom 2: x^2 + 12x - 4', color='green')
plt.scatter(x, y2, color='green')

# Titel und Labels hinzufügen
plt.title('Vergleich von zwei Polynomen')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.legend()
plt.show()
```

Wenn der Code richtig umgesetzt wurde, wirst du folgende Ausgabe sehen:

```
Punkte des ersten Polynoms (3x^2 + 2x + 8):
(1.0, 13.0)
(2.0, 24.0)
(3.0, 41.0)
(4.0, 64.0)
(5.0, 93.0)
(6.0, 128.0)
(7.0, 169.0)

Punkte des zweiten Polynoms (x^2 + 12x - 4):
(1.0, 9.0)
(2.0, 24.0)
(3.0, 41.0)
(4.0, 60.0)
(5.0, 81.0)
(6.0, 104.0)
(7.0, 129.0)
```



Zur Erinnerung: Dem Empfänger werden sieben y-Werte für die ersten sieben x-Werte übermittelt. Die korrekten Werte für das erste Polynom wären demnach 13, 24, 41, 64, 93, 128 und 169.

Falls bei der Übermittlung ein Fehler auftritt, könnten unter anderem die Zahlen 13, 24, 41, 60, 81, 128 und 169 anstelle der korrekten Werte empfangen werden.

Die Werte 60 und 81 sind demnach fehlerhaft, könnten jedoch zusammen mit den Werten 24 und 41 ein Polynom formen. Dies wurde in unserem Beispiel absichtlich so konstruiert. Allerdings könnte ein solcher Fall auch durch zufällige Fehler in der Realität auftreten.

Im ungünstigsten Szenario befinden sich, wie die Ausgabe des Codes zeigt, vier der übermittelten Punkte auf einer inkorrekten Kurve (Polynom 2). Wie ersichtlich, befinden sich auf dem Polynom 1 (dem richtigen) immer noch fünf Punkte. Das bedeutet, wir können, anhand der Anzahl der Punkte bestimmen, welches das richtige Polynom ist. Das Polynom auf dem mehr Punkte liegen ist das richtige.

Dabei gehen wir von der Annahme aus, dass höchstens zwei Fehler vorliegen. Wie du bei Aufgabe 6 gesehen hast, benötigt man zur Bestimmung aller drei Koeffizienten eines Polynoms 2. Grades 3 Punkte auf diesem Polynom. Das bedeutet, dass zwei Polynome zweiten Grades maximal zwei Schnittpunkte haben können. Ansonsten sind sie identisch oder haben keine Schnittpunkte oder nur einen Berührungspunkt.

Im Allgemeinen gilt, dass der Grad von Polynomen bestimmt, wie viele Schnittpunkte sie maximal haben können. Die Anzahl Schnittpunkte entspricht dem Grad.

Wenn der Empfänger aus den übermittelten Punkten zwei verschiedene Polynome konstruieren könnte, dann würde das Polynom, auf dem mehr Punkte liegen, als das korrekte identifiziert werden. Zur Erinnerung: die eigentliche zu übermittelnde Information, sind die Koeffizienten des Polynoms zweiten Grades, welche der Empfänger basierend auf den gesendeten Punkten ermittelt.

Demnach wurden sieben Zahlen übermittelt, um drei Zahlen gegen das Risiko von zwei Fehlern abzusichern. Das bedeutet, dass selbst im Falle von zwei Übertragungsfehlern die korrekten Informationen noch rekonstruiert werden können.

Bei der dreifachen Übermittlung in Aufgabe 2 stellte sich heraus, dass neun Zahlen gesendet werden müssen, um drei Zahlen gegen einen Fehler abzusichern. Und trotzdem könnte bei systematischen Fehlern der fehlerhafte Wert irrtümlich als korrekt betrachtet werden.

Setzen wir die Grösse der zu übermittelnden Daten, die wir als "Fracht" bezeichnen, auf f fest. In unserem Beispiel bestand die Fracht aus zu übermittelnden Zahlen.

Anschliessend bestimmen wir, gegen wie viele Fehler wir uns «versichern» möchten, und bezeichnen diese Anzahl als v.

g als den Grad des Polynoms, das für diesen Zweck benötigt wird.

Wir definieren y als die Anzahl der zu sendenden y-Werte und Tragen wir das in eine Tabelle ein mit den werten unseres Beispiels und ergänzen um eine Begründung

Fracht f	Versicherung v	Grad g	Anzahl y-Werte y	Begründung
3	2	2	7	g: für eine Fracht von 3 braucht es ein Polynom mit 3 Koeffizienten also ein Polynom 2. Grades y: damit auf der falschen Kurve weniger Punkte als auf der richtigen sind müssen 7 Werte versendet werden. (bei Polynomen 2. Grades können maximal zwei Punkte des richtigen und des falschen Polynoms identisch sein also bleiben immer fünf Punkte übrig die alleine auf dem richtigen Polynom liegen MÜSSEN, wobei auf dem falschen maximal vier liegen können: zwei identische und zwei fehler)

4.5 Aufgabe 8

Ergänze die Tabelle

Fracht f	Versicherung v	Grad g	Anzahl y-Werte y	Begründung
4	2			g: y:

Lösung:

Fracht f	*Versicherung v*	*Grad g*	*Anzahl y-Werte y*	*Begründung*
4	*2*	*2*	*8*	*g: für eine Fracht von 3 braucht es ein Polynom mit 3 Koeffizienten also ein Polynom 2. Grades* *y: Angenommen, es treten zwei Fehler auf, dann können höchstens drei weitere Punkte auf dem konkurrierenden Polynom liegen. Das ergibt insgesamt fünf Punkte. Um sicherzustellen, dass das korrekte Polynom identifiziert wird, müssen auf diesem mindestens sechs Punkte korrekt liegen. Dazu zählen die acht ursprünglichen Punkte abzüglich der zwei fehlerhaften.*

4.6 Aufgabe 9

Ergänze die Tabelle

Fracht f	Versicherung v	Grad g	Anzahl y-Werte y	Begründung
3	3			g: y:

Lösung:

Fracht f	Versicherung v	Grad g	Anzahl y-Werte y	Begründung
3	3	2	9	*g: für eine Fracht von 3 braucht es ein Polynom mit 3 Koeffizienten also ein Polynom 2. Grades* *y: damit auf der falschen Kurve weniger Punkte als auf der richtigen sind müssen 8 Werte versendet werden. (bei Polynomen 2. Grades können maximal zwei Punkte des richtigen und des falschen Polynoms identisch sein. Werden von den 9 Werten 3 Falsche abgezogen leiben immer 6 Punkte übrig die auf dem richtigen Polynom liegen MÜSSEN.* *Auf dem falschen können maximal 5 liegen die 3 falschen punkte und die zwei die mit dem richtigen polynom identisch sind.*

4.7 Aufgabe 10

Leite aus dem Beispiel, der Aufgabe 7 und der Aufgabe 8 eine allgemeingültige Formel zur Bestimmung von g ab. Leite ebenfalls eine allgemeingültige Formel zur Bestimmung von y ab.

$f - 1 = g$

$f + 2 * v = y$

Es ist nun ersichtlich, in welchen Situationen eine bestimmte Anzahl von Fehlern korrigiert werden kann. Jedoch ist in denselben Konfigurationen die Erkennung einer grösseren Anzahl von Fehlern möglich.

4.8 Aufgabe 11

Erläutern unter Verwendung der Zahlen aus dem Beispiel, wie viele zusätzliche Fehler gemacht werden können im Vergleich zu der Anzahl der Fehler, die korrigiert werden können, während der Empfänger noch in der Lage ist festzustellen, dass die Übertragung nicht korrekt ist.

Es ist möglich, dass diese 4 falschen Werte, zusammen mit 2 weiteren korrekten Werten, ein inkorrektes Polynom bilden. Das ergibt insgesamt 6 Punkte auf dem falschen Polynom.

Da aber insgesamt 7 Werte übermittelt wurden, bleibt mindestens ein Punkt, der nicht auf diesem inkorrekten Polynom liegt. Folglich lässt sich feststellen, dass ein Fehler vorliegt. Auch wenn die exakten korrekten Werte nicht mehr rekonstruiert werden können, ist es möglich, den Fehler zu erkennen.

Es bedeutet, dass die doppelte Anzahl an Werten fehlerhaft übertragen werden kann, verglichen mit der Anzahl der korrigierbaren Fehler, und es trotzdem noch möglich ist festzustellen, dass ein Fehler vorliegt.

4.9 Aufgabe 12

Vergleiche die Fehlerkorrekturtechnik des Kartentricks mit der Reed-Solomon-Codierung und erläutere die Unterschiede zwischen den beiden Ansätzen.

Beim Kartentrick müssen zusätzlich zur Länge der Nachricht n $2\sqrt{n}+1$ *Zahlen übermittelt werden, um einen Fehler zu korrigieren.*

Nehmen wir also zum Beispiel eine Nachricht der Länge 16. Dafür werden mit der Kartentrickcodierung 24 Zahlen übermittelt.

*Mit Reed-Solomon-Codierung wären es 16+2*1= 18 Zahlen.*

Die Grenze des Kartentricks liegt bei 3 Fehlern, wenn die im Buch behandelte Idee auf die Diagonalen erweitert wird. Allerdings erhöht sich der Faktor vor der Wurzel um das Dreifache. Im Gegensatz dazu kann die Reed-Solomon-Codierung theoretisch auf beliebig viele Fehlerkorrekturen erweitert werden

5 Schlussbemerkung

Du hast nun das Ende des Skripts zur Einführung in die Reed-Solomon-Codierung erreicht und eine grundlegende Einsicht in ihre Funktionsweise gewonnen. Es ist wichtig, sich in Erinnerung zu rufen, dass für die praktische Umsetzung in verschiedenen Anwendungen einige mathematische Anpassungen erforderlich sind, die wir in diesem Rahmen nicht vertieft behandelt haben. Ein Überblick über diese Anpassungen wurde am Anfang präsentiert. Du hast selbst festgestellt, dass bei der Rückgewinnung der Polynome gerundet wird oder bei der Bestimmung von Polynomen Gleichungssysteme gelöst werden müssen. Eine direkte Umsetzung dieser Methoden würde einerseits zu Ungenauigkeiten und andererseits zu extrem langer Rechenzeit führen. Wenn zusätzlich für die Fehlerpunkte bestimmt werden müsste, welche jetzt auf dem richtigen und welchen auf dem falschen Polynom liegen, würde selbst ein Computer nie fertig werden.

Bedenke dass z.B. für eine Minute Netflix-Streaming in HD-Qualität etwa 224 Millionen Bits (Einsen und Nullen) übertragen werden müssen

Diese Einsichten sollen dir dabei helfen, die Komplexität und die Herausforderungen in der Anwendung solcher Codierungsverfahren zu verstehen und gleichzeitig die Bedeutung effizienter Algorithmen und Optimierungsstrategien in der Informatik zu würdigen.

Zum Überprüfen ob du das Ziel erreicht hast kannst du diese Lernziele durchgehen:

1. **Ich kann die Grundidee der Reed-Solomon-Codierung erklären.** Dies beinhaltet das Verständnis, dass Daten als Koeffizienten von Polynomen kodiert werden, um eine erhöhte Robustheit gegenüber Fehlern bei der Übertragung zu erreichen.

2. **Ich kann grundlegende Operationen mit Polynomen durchführen.** Da ein Basiswissen über Polynome vorausgesetzt wird, solltest du in der Lage sein, einfache Rechenoperationen mit Polynomen zu verstehen und durchzuführen.

3. **Ich kann die Funktionsweise der Reed-Solomon-Codierung durch die Erstellung und Analyse von Python-Code erforschen und nachvollziehen.** Dieses Ziel hebt die praktische Anwendung des Gelernten hervor, indem du durch Programmieren in Python die Reed-Solomon-Codierung dekonstruieren und ihr Verständnis so vertiefen konntest.