**IMPLEMENTIERUNG**

**Polynomdarstellung**

Um in der Programmiersprache Python Polynome darzustellen, wird die Datenstruktur Tupel verwendet. In einem Tupel können mehrere Inhalte unveränderlich gespeichert werden. Da in der Charakteristik Zwei lediglich Zahlen aus dem Binärsystem Vorfaktoren der verschiedenen Monome sein können, werden die Tupel nur mit Nullen und Einsen befüllt. Dadurch wird bestimmt, ob ein bestimmtes Monom in dem Polynom vorkommt (in diesem Fall mit einer „1“ an der entsprechenden Stelle dargestellt) oder ob das Monom nicht in dem Polynom vertreten ist (Mit einer „0“ dargestellt).

In diesen Tupeln steht das Monom mit dem höchsten Exponent an der ersten stelle (entspricht der Stelle ganz links) und die Exponenten nehmen dann schrittweise nach rechts ab, sodass die letzte Stelle (ganz rechts) dem Monom mit dem Exponent „0“ entspricht. Diese Darstellung durch Tupel vereinfacht den Umgang mit Polynomen, da die ausführliche mathematische Schreibweise zusätzliche, unnötige Informationen wie die Variable „x“ oder das „+“-Zeichen. Für den Umgang mit Polynomen ist jedoch nur wichtig, welcher Vorfaktor an welcher Stelle des Polynoms steht, und diese beiden Informationen sind in der Schreibweise als binäre Tupel enthalten, wodurch sie sich anbietet für das Arbeiten mit Polynomen.

Um solch ein binäres Tupel in Python wieder in die mathematische Schreibweise eines Polynoms mit Vorfaktor, Variable „x“ und Exponent für jedes Monom darzustellen wurde die Funktion „TupleToPolynom“ erstellt. Wie der Name schon sagt, bekommt die Funktion ein Tupel als Funktionsparameter und gibt einen String zurück, der das vollständige Polynom umfasst. Dabei iteriert die Funktion über die einzelnen Elemente des Tupels und bestimmt daraus einen Teilstring. Bei der Erstellung dieser Teilstrings gibt es drei verschiedene Fälle. Bei dem letzten Monom eines Tupels, also dem Monom ganz rechts, wird der Teilstring lediglich eine „1“, da das Letzte Monom den Exponenten Null hat, und alles hoch Null gleich Eins ergibt. Bei dem vorletzten Monom wird der Teilstring zum einem „X“, da dieses Monom den Exponenten Eins hat. Für alle anderen Fälle, bei denen der Exponent größer als Eins ist, wird der Teilstring zu „X^n“, wobei n dem Exponenten des Monom entspricht. Im Zuge dieser Funktion wird das Tupel vor der Erstellung des Lösungsstrings einmal umgekehrt, sodass nun das letzte Monom an erster Stelle steht, das vorletzte an zweiter Stelle steht, und so weiter. Dadurch gewinnt man den Vorteil, dass jetzt die Position des Monoms in dem Tupel gleich dem Exponenten des Monoms entspricht. Da bei Tupeln, ähnlich wie bei Arrays das Zählen bei der Null anfängt steht nach der Umkehrung das erste Monom (was ursprünglich das Letzte war) an der Position Null in dem Tupel, was genau dem Exponenten dieses Monoms entspricht. All diese einzelnen Teilstrings werden dann aneinandergereiht, wobei zwischendrin immer ein „+“ beigefügt wird. So erhält man schließlich einen String der das vollständige Polynom, wie man es auch in der mathematischen Schreibweise aufschreiben würde.

Diese Funktion, die aus einem Tupel wieder ein Polynom macht, erfüllt keinen direkten Zweck, der für weitere Operationen wichtig ist, sondern diese Funktion dient lediglich dazu, aus einem Tupel eine für den Anwender lesbare Darstellung zu bieten, da die Tupel, vor allem bei steigender Länge unübersichtlich werden können.

**Hilfsfunktionen**

Im Laufe der Implementierung werden für einige Operationen Hilfsfunktionen benötigt. Diese sind in der Datei „AuxiliaryFunctions.ipynb“ gesammelt, da sie später von verschiedenen Stellen aus benötigt werden. Im Folgenden werden diese Funktionen genauer erläutert.

Wegschneiden von Nullen

Um führende Nullen von Polynomen, bzw. von binären Tupeln wegzuschneiden, wurde die Funktion „cut\_zeros\_left“ implementiert, die dafür sorgt, dass das gegebene Tupel mit der ersten Eins, die in dem Tupel vorhanden ist, beginnt, und alle vorherstehenden Stellen abgeschnitten werden.

Um diese führende Nullen abzuschneiden, wird über das binäre Tupel iteriert, und überprüft, ob die erste Stelle eine Null ist. Sollte diese Bedingung wahr sein, wird die erste Stelle abgeschnitten. Dieser Vorgang wird so lange wiederholt, bis die erste Stelle eine Eins ist.

Für den Sonderfall, dass ein leeres Tupel in die Funktion gegeben wird, muss eine zusätzliche Bedingung eingeführt werden, da in einem leeren Tupel keine Eins gefunden werden kann, und somit eine Endlosschleife entstehen würde. Dieser Fall kann einfach abgefangen werden, in dem in der Schleifenbedingung geprüft wird, ob das Tupel leer ist, und die Schleife nur durchlaufen wird, sollte das Tupel nicht leer sein.

Erste Einser-Stelle finden

Die Funktion „find\_pos\_of\_first\_one“ dient dazu, die erste Stelle eines Polynoms, von links aus, zu finden die mit einer Eins befüllt ist. Diese Funktion wird im späteren Verlauf benötigt, um den Grad einer Funktion ermitteln zu können.

Um diese Stelle bestimmen zu können, wird über die Stellen des gegebenen Tupels iteriert. Sobald eine Eins gefunden wurde, wird der Index zurückgegeben, bei dem sich die Iteration zu dem Zeitpunkt befindet. Dieser Index stellt dann die Position der ersten Eins des Polynoms dar.

Sollte keine Eins gefunden werden, wird von der Funktion „None“ zurückgegeben. Dieses Ergebnis bedeutet dann, das das Tupel leer sein muss, da es lediglich aus Nullen besteht.

Grad eines Polynoms bestimmen

Der Grad eines Polynoms wird bestimmt durch den höchsten Exponent, der in dem jeweiligen Polynom zu finden ist. Um nun, mittels der Funktion „find\_degree“, den Grad eines Polynoms bestimmen zu können, muss zunächst die Erste Einser-Stelle des Polynoms ermittelt werden. Dies ist durch die eben aufgeführte Hilfsfunktion „find\_pos\_of\_first\_one“ möglich.

Anhand dieser Stelle, welche das Monom mit dem höchsten Exponent darstellt, kann nun der Grad des Polynoms ermittelt werden. Dazu muss von der Gesamtlänge des Polynoms die Zahl, der Index der Ersten Eins abgezogen werden. Dadurch erhält man die Länge des Polynoms, wenn man von der ersten Eins aus anfängt zu zählen. Von dieser Länge muss nun noch eins abgezogen werden, da das letzte Monom den Exponent Null hat.

Als Ergebnis erhält man den höchsten Exponenten, der in dem Polynom zu finden ist, und dies entspricht dem Grad des Polynoms.

Polynom auf Nullstellen überprüfen

Mittel der Funktion „has\_root“ ist es möglich ein Polynom darauf zu prüfen, ob es ein Nullstelle hat. Ein Polynom hat genau dann eine Nullstelle, wenn man in dem Polynom für die Variable „x“ etwas einsetzen kann, wodurch das Polynom zu Null wird. In der Charakteristik Zwei gibt es hierbei lediglich die beiden Möglichkeiten Null und Eins, die in das Polynom eingesetzt werden können.

Die Null stellt genau dann eine Nullstelle dar, wenn das kleinste Monom den Vorfaktor Null hat. Denn in alle Anderen Monome des Polynoms wird eine Null eingesetzt, wodurch sich all diese zu Null addieren. Dadurch muss nur das niedrigste Monom betrachtet werden, da dies durch den Exponent von Null nicht von der für die Variable eingesetzte Zahl abhängt. Demnach gilt, wenn die letzte Stelle des Tupels eine Eins ist, stellt die Null keine Nullstelle dar, und wenn die letzte Stelle des Tupels eine Null ist, ist die Null eine Nullstelle des Polynoms.

Für die Eins hingegen müssen andere Bedingungen überprüft werden. Hierfür ist insbesondere die folgende Rechenregel wichtig, die aufgrund des Binärsystems gilt: „1 + 1 = 0“. Daraus ergibt sich, dass wenn die Summe der Einser gerade ist, wird das Ergebnis Null, da sich immer zwei Einser gegenseitig auslöschen. Dadurch gilt, dass die Eins eine Nullstelle ist, sollte die Anzahl Einser im binären Tupel gerade sein. Somit ist bei ungerader Anzahl Einser im Tupel die Eins keine Nullstelle.

In der Implementierung wurden die eben aufgeführten Bedingungen für Nullstellen folgendermaßen umgesetzt. Da für die Null lediglich die letzte Stelle entscheidend ist, ob sie eine Nullstelle für das Polynom ist, wird überprüft, ob die letzte Stelle eine Null ist. Sollte diese Bedingung stimmen, gibt die Funktion True zurück, da in diesem Fall eine Nullstelle vorliegt. Um die Eins zu überprüfen, wird stellenweise über das binäre Tupel iteriert und die Anzahl der vorkommenden Einser gezählt. Sollte diese Anzahl gerade sein, gibt es eine Nullstelle und somit gibt die Funktion „True“ zurück. Sollten diese beiden aufgeführten Bedingungen nicht eingetreten sein, hat das Polynom keine Nullstelle und die Funktion gibt „False“ zurück.

**Arithmetische Operationen**

Da die Charakteristik Zwei auf dem Binärsystem basiert, verhalten sich die arithmetischen Operationen innerhalb von Körpern der Charakteristik Zwei analog zu den Berechnungen im Binärsystem. Im Folgenden wird die Implementierung der Arithmetischen Rechenoperationen für binäre Tupel, welche Polynome der Charakteristik Zwei darstellen, genauer erläutert.

Addition von Polynomen

Bei der Addition von Polynomen in Körpern der Charakteristik Zwei gelten die Rechenregeln wie bei der Addition im Binärsystem. Diese lauten folgendermaßen. 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1 und 1 + 1 = 0 (mit Übertrag 1).

Um nun diese Addition nun in Python zu implementieren, werden der Funktion „AddPolynoms“ zunächst zwei Polynome übergegeben, die in Form von binären Tupeln dargestellt werden. Um diese beiden binäre Tupel zu addieren, wird über die Stellen der Tupel iteriert, und jede Stelle einzeln, nach den oben gegebenen Rechenregeln, binär addiert. Die Ergebnisse werden dann schrittweise dem Lösungspolynom hinzugefügt.

Bevor die Iterative Addition der Stellen der Polynome durchgeführt werden kann, muss jedoch sichergestellt werden, dass auch jeweils die passenden Stellen miteinander addiert werden. Als Voraussetzung dafür, dass immer die passenden Stellen miteinander addiert werden, gilt, dass die beiden Polynome, beziehungsweise die beiden Tupel, gleichviele Stellen haben. Um sicherzustellen, dass die beiden Tupel gleichviele Stellen haben, wird bevor der iterativen Addition geprüft, welches Tupel das längere der beiden ist, und im Anschluss wird das kürzere Tupel von links aus mit Nullen ergänzt, bis es so viele Stellen wie das jeweils andere hat. Diese Nullen dienen lediglich zur Richtigen Zuordnung der Stellen bei der Addition. Zudem ändern sie nichts an den Rechnungen, da die Zahl Null das neutrale Element in der Addition darstellt. Demnach wird durch diese Auffüllung mit Nullen das Ergebnis der Addition nicht beeinflusst.

Schließlich wird am Ende das Lösungspolynom in Form eines binären Tupels zurückgegeben.

Multiplikation von Polynomen

Generell mathematisch

Implementierung

Reduktion mit Relation

Polynomdivision