Problemlösung

Vorlesungsskript Digitalisierung und Programmierung

Prof. Dr. Nicolas Meseth

Inhaltsverzeichnis

Warum nutzen wir Computer?

Der wichtigste Grund für die Nutzung von Computern ist das Lösen von Problemen. Ob wir eine Route mit Google Maps planen, Online-Bestellungen bei DHL verfolgen oder eine KI wie ChatGPT um eine Empfehlung bitten – überall lösen Computer Probleme. Warum? Weil Computer zwei Eigenschaften besitzen, die für viele Probleme und deren Lösung vorteilhaft sind:

1. Computer machen keine Fehler. Wenn wir einem Computer einen Lösungsweg beibringen, wendet er ihn fehlerfrei auf neue Probleme an.
2. Computer sind unglaublich schnell. Ob einfache Schritte, komplexe Berechnungen oder die Verarbeitung großer Datenmengen – Computer lösen Probleme in einem Bruchteil der Zeit, die wir Menschen benötigen würden.

Diese beiden Eigenschaften ermöglichen es uns, mit Computern besonders solche Probleme effizient zu lösen, die wiederkehrend und in großer Zahl auftreten. Wir sprechen dann von **Automatisierung**.

In diesem Kapitel lernen wir, wie Computer Probleme strukturieren und lösen. Um den Begriff des Problems besser zu verstehen und seine Bedeutung im Kontext von Computern einzugrenzen, führen wir zunächst ein einfaches Modell ein.

Das Eingabe-Verarbeitung-Ausgabe-Modell

Das Eingabe-Verarbeitung-Ausgabe-Modell (EVA-Modell, s. Abbildung 1) ist ein grundlegendes Konzept der Informatik, das die Arbeitsweise von Computern vereinfacht erklärt. Es zeigt, wie Computer Probleme lösen: Sie wandeln Eingabedaten durch einen definierten Verarbeitungsprozess in gewünschte Ausgaben um.

Das Modell beschreibt ein Problem und dessen Lösung als eine Eingabe (*Input*), die dem Computer übergeben wird. Darauf folgt eine Verarbeitung (*Computation*) auf Basis dieser Eingabe, die wiederum eine Ausgabe (*Output*) erzeugt.

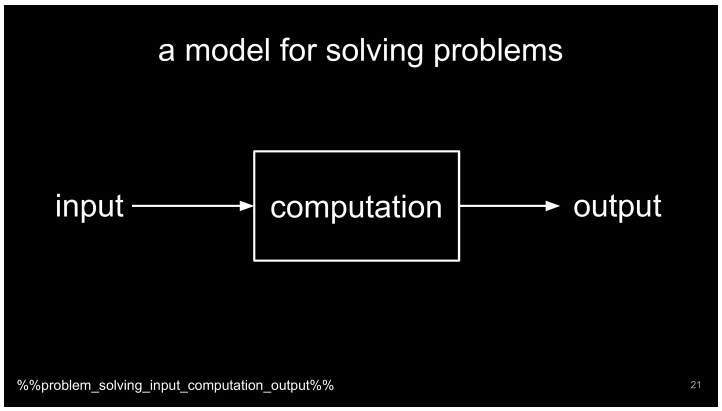


Abbildung 1: Das EVA-Modell besteht aus der Eingabe, der Berechnung und der Ausgabe.

Taschenrechner

Am Beispiel eines Taschenrechner lässt sich das EVA-Modell gut darstellen. Wir können uns bildlich vorstellen, wie ein Mensch die Eingabe tätigt und danach das Ergebnis abliest. Es ist wichtig zu verstehen, dass Eingabe und Ausgabe sehr unterschiedliche Formen annehmen können und keinesfalls nur über eine Tastatur erfolgen müssen.

Das Beispiel des Taschenrechners wird in Abbildung 2 anhand einer einfachen Addition zweier Zahlen konkreter verdeutlicht. Als Eingabe werden zwei Zahlen benötigt, die Berechnung

erfolgt durch eine Addition, dargestellt durch das Plussymbol. Die Ausgabe ist das Ergebnis – die Summe.

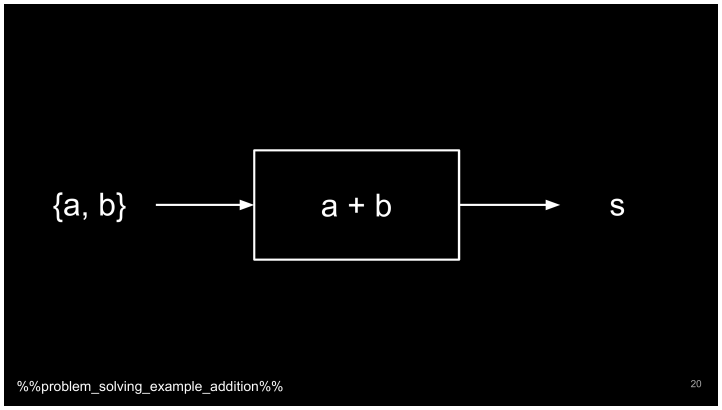


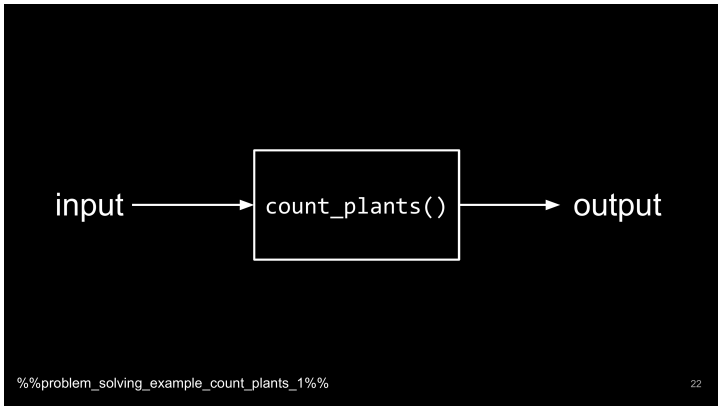
Abbildung 2: Das EVA-Modell für die Addition zweier Zahlen.

Dieses einfache Beispiel zeigt, dass wir verstehen müssen, wie Computer die drei Bestandteile des EVA-Modells umsetzen. Beim Taschenrechner sind Ein- und Ausgabe jeweils Zahlen. Diese Daten speichert der Computer in seinem Arbeitsspeicher. Dabei ist wichtig zu wissen, dass Computer auf der untersten Ebene ausschließlich Nullen und Einsen speichern. Wir müssen also verstehen, wie Computer Zahlen mithilfe dieser Binärzahlen darstellen können.

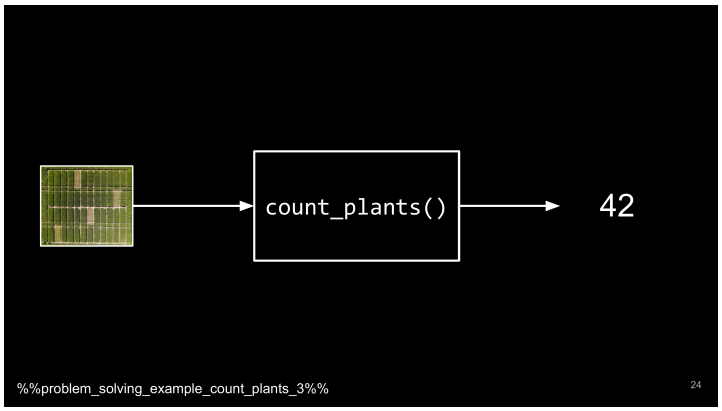
Was passiert bei der Berechnung in der Mitte des Modells? Eine Addition mag uns einfach erscheinen, doch auch hier müssen wir beachten, dass Computer mit Binärzahlen arbeiten. Es stellt sich also die Frage: Wie funktioniert eine Addition, wenn die Zahlen als Folge von Nullen und Einsen dargestellt sind? Auf die beiden Fragen zur **Repräsentation und der Verarbeitung von Informationen** im binären System werden wir im Laufe des Buches Antworten bekommen.

Pflanzen zählen

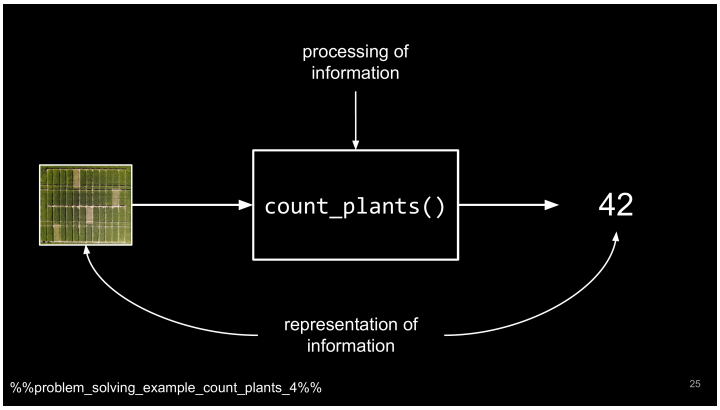
Betrachten wir ein weiteres Beispiel: Stell dir vor, du möchtest einen Computer nutzen, um Maispflanzen auf einer Drohnenaufnahme eines Ackers zu zählen. Diese Aufgabe ist für Menschen zwar einfach zu verstehen, wäre aber sehr zeitaufwändig auszuführen. Moderne Algorithmen ermöglichen es Computern, Objekte auf Bildern präzise zu lokalisieren und zu zählen. Nehmen wir an, wir haben für dieses Problem bereits eine Lösung entwickelt und ein Programm namens `count_plants` erstellt. Nun stellt sich die Frage: Wie sehen die Eingabe und die Ausgabe für dieses Problem aus? Was benötigt das Programm von uns, und was liefert es als Ergebnis?



Die erwartete **Ausgabe** lässt sich einfach beschreiben: Das Ergebnis der Zählung ist eine ganze Zahl. Die **Eingabe** für dieses Problem ist - anders als beim Taschenrechner - kein Tastendruck, sondern ein Bild. Damit der Computer das Bild verarbeiten kann, muss es dem Computer in digitaler Form bereitgestellt werden. Was das genau bedeutet, lernen wir in einem späteren Kapitel. Hier genügt es uns zu verstehen, *dass* wir das Bild digital abbilden müssen.

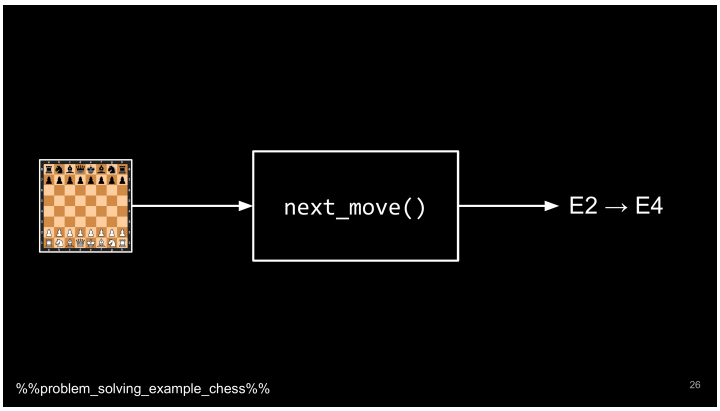


Wie gelangt das Bild in den Computer? Dies ist im Modell nicht näher definiert und für die Problembeschreibung auch nicht wesentlich. Das Bild muss lediglich irgendwie in den Arbeitsspeicher des Programms `count_plants` gelangen. Dies kann auf verschiedene Arten geschehen: Es kann von der Festplatte gelesen werden, über eine drahtlose Verbindung wie Bluetooth direkt übertragen und verarbeitet werden, oder das Programm `count_plants` läuft direkt auf der Drohne und greift unmittelbar auf deren Kamera zu. Die technische Umsetzung ist für unser Modell zunächst irrelevant. In einem späteren Kapitel werden wir uns damit befassen, wie Informationen genau übertragen und gespeichert werden. Genauso werden wir lernen, wie die benötigten Informationen für die Ein- und Ausgabe eines Programms in digitaler Form dargestellt werden können.



Schach spielen

Ein weiteres Beispiel zur Verdeutlichung des EVA-Modells ist das Schachspiel. Das Problem lässt sich einfach beschreiben: Der Computer soll auf Grundlage einer bestehenden Spielsituation den bestmöglichen nächsten Zug vorschlagen. Dieser Zug soll die Gewinnchancen maximieren.



Betrachten wir zunächst die Eingabe für dieses Problem: Wir können dem Computer nicht einfach ein physisches Schachbrett zeigen, sondern müssen überlegen, wie sich ein Schachbrett und die Position der Figuren in digitaler Form darstellen lassen. Dabei kann es durchaus mehrere Möglichkeiten geben, die uns ans Ziel führen.

Ein Schachbrett lässt sich etwa als Liste von 64 Feldern darstellen, die von oben links nach unten rechts durchnummeriert sind. Für jedes Feld speichern wir, ob es leer ist oder welche Figur darauf steht. Die Figuren werden durch Buchstaben dargestellt – zum Beispiel “R” für den Turm (Englisch: Rook) oder “N” für den Springer (Englisch: Knight). Für die Farben Schwarz und Weiß verwenden wir einfach 0 und 1. Diese Darstellungsform reduziert unser

Problem auf Listen, Zahlen und Buchstaben in digitaler Form. Für Computer ist das eine leicht zu verarbeitende Struktur, wie wir später noch sehen werden. Ein Beispiel für eine solche Kodierung zeigt Abbildung 3.

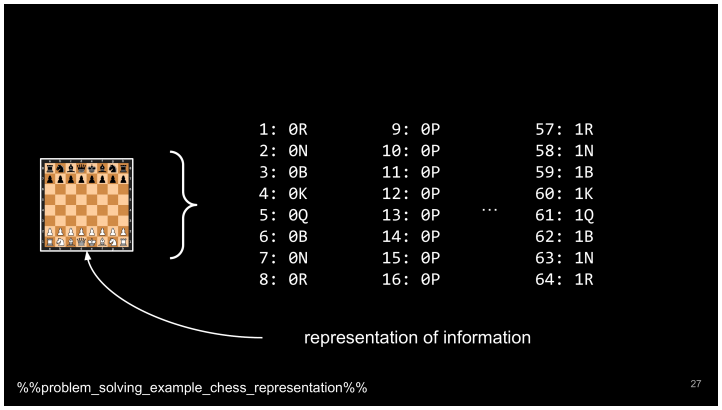


Abbildung 3: Beispiel für die Darstellung von Schachfiguren als Zahlen und Buchstaben.

Die Ausgabe, also der nächste Zug, lässt sich ebenfalls durch Zahlen und Buchstaben darstellen. Eine weit verbreitete Notation gibt zunächst die Koordinate des Ausgangsfelds an, von dem eine Figur gezogen werden soll, gefolgt von der Koordinate des Zielfelds. Ein Beispiel wäre der Zug von “E2 nach E4”. Statt “E2” und “E4” könnten wir ebenso die entsprechende Zahl zwischen 1 und 64 aus unserer Liste verwenden, um mit dem obigen Schema konsistent zu bleiben. Der Zug hieße dann “53 nach 37”.

Mit Computern chatten

Als drittes Beispiel betrachten wir die Verwendung von Chatprogrammen wie ChatGPT. Seit seiner Veröffentlichung im November 2022 hat es die Welt stark verändert und einen regelrechten KI-Hype ausgelöst. Ein großes Sprachmodell wie GPT-4, das hinter dem heutigen ChatGPT steckt, ist eine komplexe Software, die wir in diesem Buch nicht vollständig ergründen können. Das Schöne an Modellen wie dem EVA-Modell ist jedoch, dass sie komplexe Sachverhalte vereinfachen können – so auch bei Sprachmodellen. Das Problem, das Sprachmodelle lösen, lässt sich wie alle Probleme in unserem EVA-Modell einfach darstellen.

Dabei betrachten wir das Sprachmodell – oder ChatGPT – als Blackbox, ohne die internen Prozesse genauer zu definieren. Für unser Modell genügt es zu verstehen, dass wir eine Eingabe in Form einer Nachricht an ChatGPT benötigen und als Ausgabe eine Antwort erhalten. Auch hier stellt sich die Frage, wie wir beides digital repräsentieren können, damit ChatGPT damit arbeiten kann.

Klassischerweise bestehen sowohl Eingabe als auch Ausgabe einfach aus Texten – allerdings beherrschen moderne Sprachmodelle auch andere Eingabeformen wie Bilder oder gesprochene

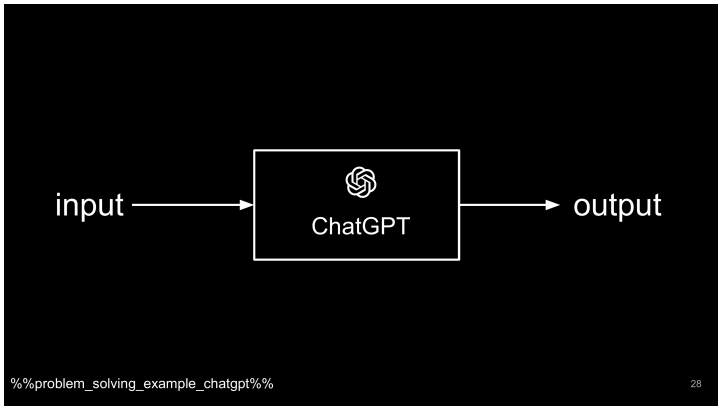
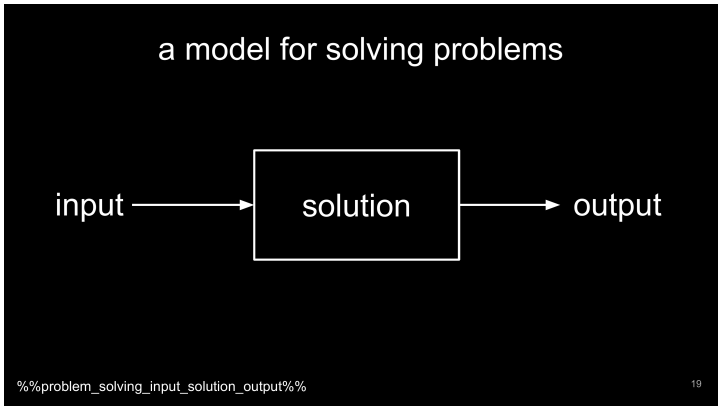


Abbildung 4: ChatGPT im EVA-Modell.

Sprache über ein Mikrofon. Wir sprechen dann von multimodalen KI-Modellen. Bei Bildern stehen wir vor demselben Repräsentationsproblem wie bei unserer Drohnenaufnahme. Bei der Sprache stellt sich neben der Repräsentation von Audioinhalten die Frage, wie wir gesprochene Worte überhaupt in eine digitale Form überführen können. Auch dazu erfahren wir im späteren Verlauf des Buches mehr.

Die Lösung des Problems

Anhand des EVA-Modells wird deutlich, dass wir dem Computer Informationen in Form von digitalen Daten bereitstellen müssen, mit denen er arbeiten kann. Was aber genau soll er damit machen? Hier kommt der mittlere Kasten des Modells ins Spiel – die eigentliche Lösung des Problems.



In der Informatik nennen wir die Beschreibung zur Lösung eines Problems einen **Algorithmus**. Ein Algorithmus ist eine Schritt-für-Schritt-Anleitung zur Problemlösung und ist zunächst

unabhängig von Computern. Das bedeutet, wir können die Lösung eines Problems ohne Bezug zu einem Computer beschreiben und nennen das einen Algorithmus.

Stellt euch dazu zum Beispiel eine IKEA-Aufbauanleitung vor. Sie beschreibt in sequenziellen Schritten, was zu tun ist, um das fertige Möbelstück zu bekommen. Die Eingabe besteht aus den mitgelieferten Teilen, Schrauben und dem benötigten Werkzeug für den Zusammenbau. Die Ausgabe ist das fertige Regal (oder ein anderes Möbelstück) – und das alles ganz ohne Computer.

Oder nehmt das Kochrezept eure Lieblingsessens. Auch ein Kochrezept ist ein Algorithmus: Die Eingabe besteht aus den Zutaten und Küchenutensilien, die Verarbeitung erfolgt durch die Schritt-für-Schritt-Anleitung, und die Ausgabe ist das fertige Gericht. Wie bei der IKEA-Anleitung ist der Algorithmus unabhängig von einem Computer – er beschreibt lediglich die Lösung des Problems "Wie koche ich dieses Gericht?".

Zu Beginn des Kapitels haben wir festgestellt, dass Computer aufgrund ihrer Geschwindigkeit und Fehlerfreiheit besonders gut zur Problemlösung geeignet sind – insbesondere bei häufig wiederkehrenden Problemen. Die beiden genannten Beispiele, das IKEA-Regal und das Kochrezept, eignen sich allerdings nicht für eine direkte Umsetzung durch Computer. Der Grund liegt in den analogen Eingaben (Baumaterial, Kochzutaten) und Ausgaben (Möbelstück, fertige Mahlzeit). Diese kann ein Computer nicht unmittelbar verarbeiten. Dafür wären Roboter nötig, die mit der physischen Welt interagieren können. Eine solche Automatisierung lohnt sich heute nur bei Aufgaben, die sehr häufig auftreten und ansonsten mit hohen Kosten verbunden sind – wie etwa in der Automobilindustrie, wo computergesteuerte Roboter in der Produktion zum Einsatz kommen.

Nehmen wir an, dass Haushaltsroboter in Zukunft erschwinglich werden und uns beim Kochen unseres Lieblingsgerichts helfen können. Wie vermitteln wir dann dem Roboter – im Grunde ein Computer mit Armen und Beinen – den Algorithmus für unser Rezept? Die Lösung liegt in der Programmierung: Wir erstellen ein **Programm** in einer computerverständlichen Sprache. Dieses Programm wandelt die Anweisungen aus unserem Kochbuch in Befehle um, die der Computer verstehen und ausführen kann. Genau genommen besteht ein Programm ebenfalls nur aus Informationen, und wir müssen herausfinden, wie wir diese Informationen digital darstellen können. Die Lösung besteht in der Verwendung einer **Programmiersprache** wie Python, die wir später kennenlernen werden.

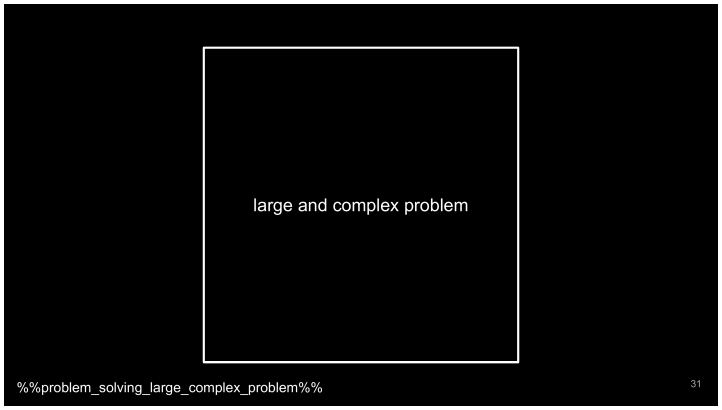
Problemlösungsstrategien

Die konkrete Lösung und der zugehörige Algorithmus sehen für jedes Problem unterschiedlich aus. Das Erkennen von Pflanzen folgt einer anderen Logik als die Entscheidung, welche Schachfigur als nächstes gezogen werden soll. Dennoch gibt es universelle Lösungsstrategien, die auf viele Probleme anwendbar sind, um sie möglichst effizient zu lösen. Im Folgenden betrachten wir drei dieser Strategien.

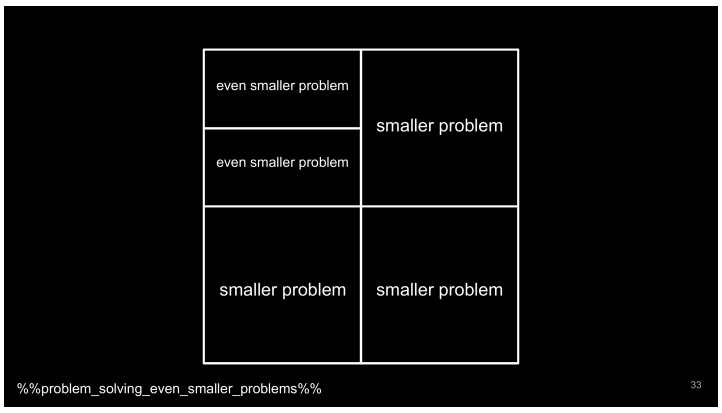
Problemzerlegung (*Problem Decomposition*)

Eine universelle Strategie zur Lösung komplexer Probleme ist die Zerlegung in kleinere Schritte oder Teilprobleme. Jedes dieser Teilprobleme ist unterschiedlich und erfordert einen spezifischen Lösungsansatz. Nehmen wir als Beispiel das Zählen von Pflanzen auf einer Drohnenaufnahme. Dieses komplexe Problem lässt sich, ausgehend von der Eingabe – dem digitalen Bild – in drei Teilprobleme zerlegen:

1. Pflanzen auf dem Bild lokalisieren
2. Lokalisierte Pflanzen klassifizieren: Maispflanze oder nicht?
3. Identifizierte Maispflanzen zählen



Jedes dieser Teilprobleme erfordert einen eigenen Algorithmus. Dabei ist es möglich, die Teilprobleme noch weiter zu zerlegen, um sie besser bearbeiten zu können.



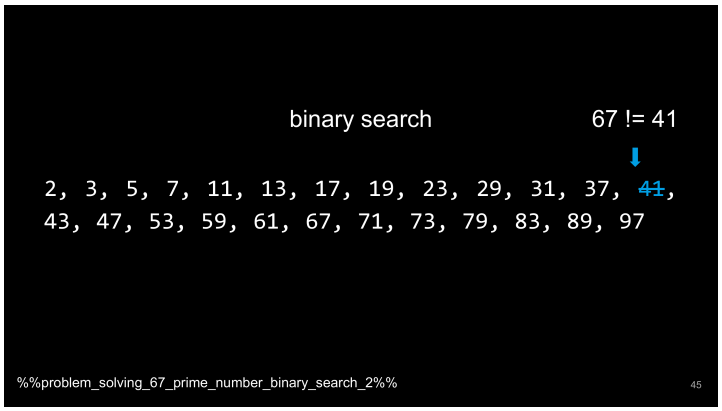
Die Identifizierung sinnvoller Teilprobleme erfordert ein ausgeprägtes analytisches Denkvermögen. Dies ist besonders wichtig im Umgang mit Computern. Wie wir beim Erlernen der Programmierung sehen werden, ist die Zerlegung eines Problems in kleine, lösbare Schritte der Schlüssel zur Beherrschung seiner Komplexität.

Teile und Herrsche (*Divide and Conquer*)

Die "Teile und Herrsche"-Strategie ist ein Ansatz zur Lösung komplexer Probleme, bei dem wir das Hauptproblem schrittweise in immer kleinere Teilprobleme zerlegen, bis diese einfach zu lösen sind. Dabei gehen wir rekursiv vor: Wir teilen das ursprüngliche Problem in kleinere Teile, diese kleineren Probleme wiederum in noch kleinere Teile und so weiter. Die Rekursion endet, wenn die Probleme so klein sind, dass sie sich nicht weiter aufteilen lassen und die Lösung direkt ersichtlich ist.

Anders als bei der Problemzerlegung sind die Teilprobleme beim Divide and Conquer-Ansatz dadurch gleichartig und stellen nur kleinere Instanzen des ursprünglichen Problems dar. Die einzelnen Lösungen für jedes Teilproblem werden dann schrittweise wieder zusammengeführt, um die Gesamtlösung zu erhalten. Ein klassisches Beispiel ist die Sortierung einer langen Liste von Zahlen: Wir teilen die Liste immer wieder in der Mitte, bis nur noch einzelne Zahlen übrig sind, und fügen diese dann in sortierter Reihenfolge wieder zusammen.

Ein anderes Beispiel ist die binäre Suche in einer sortierten Liste. Hier betrachten wir das Element in der Mitte der Liste und vergleichen es mit dem gesuchten Element. Da die Liste sortiert ist, können wir entscheiden, in welchem Teil der Liste wir weitersuchen müssen. Im zweiten Schritt suchen wir nur in diesem Teil weiter und haben damit das Problem halbiert. Die Natur des Problems bleibt dabei gleich, und wir können erneut genauso verfahren – so lange, bis wir nur noch ein Element übrig haben, das entweder das gesuchte Element ist oder nicht.



Verteile und Parallelisiere (*Distribute and Parallelize*)

Manche Probleme lassen sich effizienter lösen, wenn mehrere Personen gleichzeitig daran arbeiten. Anstatt eine einzelne Person mit der gesamten Aufgabe zu betrauen, verteilen wir die Arbeit auf mehrere Schultern und arbeiten parallel an der Lösung. Allerdings eignet sich nicht jedes Problem für diesen Ansatz.

Ein gutes Beispiel ist das Aufräumen eines Zimmers: Eine einzelne Person müsste nacheinander verschiedene Bereiche aufräumen, während mehrere Personen gleichzeitig unterschiedliche Ecken des Raums in Angriff nehmen können. Je größer das Zimmer, desto mehr Personen werden benötigt, um es in der gleichen Zeit aufzuräumen. Ein Gegenbeispiel, bei dem diese Strategie nicht funktioniert, ist das Lösen einer mathematischen Gleichung. Hier müssen die einzelnen Rechenschritte aufeinander aufbauen, weshalb das Problem nicht gleichzeitig an mehrere Personen übergeben werden kann, die unabhängig daran arbeiten.

Die Strategie des Verteilens und Parallelisierens – im Englischen *Distribute and Parallelize* – funktioniert nach einem klaren Prinzip: Wir zerlegen ein großes Problem in Teile, die unabhängig voneinander gelöst werden können. Diese Teile weisen wir dann verschiedenen Ressourcen zu – zum Beispiel mehreren Personen oder Computern. Jede Ressource arbeitet an ihrem Teilproblem und erzeugt ein eigenes Ergebnis. Dabei gehen wir davon aus, dass sich alle Teilergebnisse am Ende zu einer Gesamtlösung zusammenfügen lassen. Ähnlich wie beim Divide and Conquer-Ansatz sind die Teilprobleme meist gleichartig.

Um dieses Konzept greifbar zu machen, schauen wir uns ein konkretes Beispiel an, das wir mit dem EVA-Modell analysieren: das Zählen aller Wörter in einem Buch.

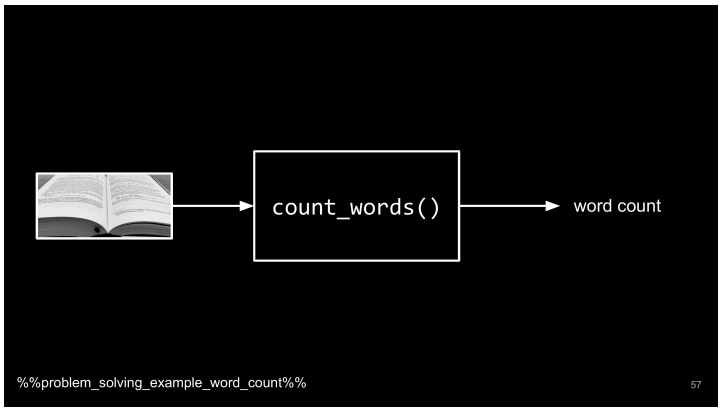


Abbildung 5: Das Wörterzählen-Problem im EVA-Modell

Je nach Umfang des Buches kann dies eine mühsame Aufgabe sein, besonders für Menschen. Ein Computer bewältigt ein einzelnes Buch dank seiner hohen Verarbeitungsgeschwindigkeit problemlos. Allerdings lässt sich das Problem beliebig erweitern – etwa wenn wir statt eines Buches alle Texte im Internet oder sämtliche Wikipedia-Artikel analysieren möchten. In solchen

Fällen wird die Aufgabe auch für Computer aufwendig und zeitintensiv. Eine Lösung besteht darin, mehrere Computer parallel einzusetzen.

In Abbildung 6 sehen wir beispielhaft die Verteilung der Buchseiten auf vier Studenten. Jeder erhält einen gleichen Anteil, wodurch sich die Bearbeitungszeit im Optimalfall auf ein Viertel reduziert. Bei Computern können wir analog vorgehen und mehrere Rechner gleichzeitig mit Teilen der Seiten betrauen. Diese Rechner werden in einem Netzwerk verbunden und von einem zentralen Computer gesteuert, der die Teilergebnisse am Ende zusammenführt. Ein solches System nennen wir Rechencluster, bestehend aus Arbeitern – den *Worker Nodes* – sowie einer Steuereinheit, die in solchen Systemen als *Driver* oder *Name Node* bezeichnet wird.

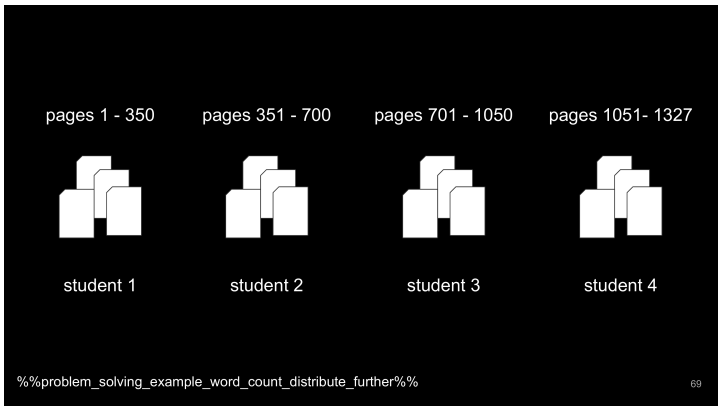


Abbildung 6: Verteiltes Wörterzählen

Durch die Verteilung und parallele Ausführung kann das EVA-Modell wie in Abbildung 7 angepasst und detaillierter dargestellt werden. Statt eines einzelnen Prozesses `count_words` laufen nun n parallele Prozesse, die jeweils einen Teil des Buches durchsuchen. Die Aufteilung erfolgt zu Beginn durch den `split`-Prozess, während das Zusammenführen der Teilergebnisse – in diesem Fall das Addieren der Teilsummen – durch den `merge`-Schritt erfolgt.

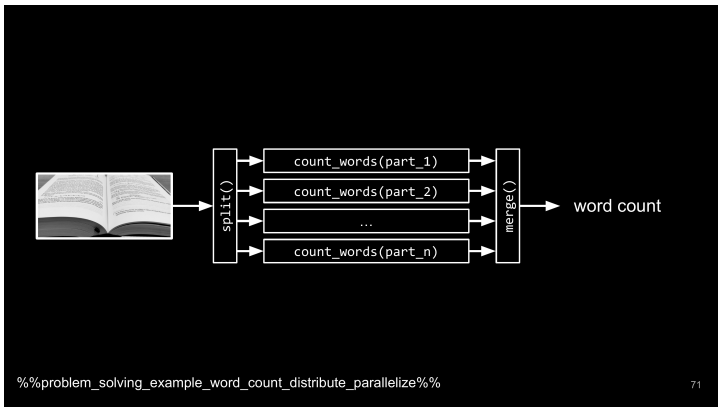


Abbildung 7: Das parallelisierte Wörterzählen im EVA-Modell

In diesem Kapitel haben wir uns mit dem EVA-Modell auseinandergesetzt - einem fundamentalen Konzept für die computergestützte Problemlösung. Dieses Modell bietet uns einen strukturierten Rahmen, der aus drei wesentlichen Komponenten besteht:

- Eingabe (E): Die zu verarbeitenden Daten oder Informationen
- Verarbeitung (V): Der Kern der Problemlösung durch Algorithmen
- Ausgabe (A): Das Ergebnis der Verarbeitung in nutzbarer Form

Die Verarbeitung als zentrales Element des Modells ist dabei der Ort, an dem die eigentliche Problemlösung stattfindet. Hier kommen Algorithmen zum Einsatz - präzise Handlungsanweisungen, die Schritt für Schritt zur Lösung führen. Die genaue Natur dieser Algorithmen, ihre charakteristischen Eigenschaften und wie wir sie entwickeln können, werden wir im nächsten Kapitel detailliert betrachten.