# Onlinealgorithmen zu Bearbeitung von Datenströmen

Aufträge zur Semesterbegleitenden Übung (FD1)
Sarina Mueller & Nicolas Kopp

## 1 Konzeption der Unterrichtseinheit:

### a) Begriffe & Vorwissen:

Die digitale Darstellung der Information ist eine Folge von diskreten Symbolen aus einem verabredeten Alphabet. Wenn man eine Information verschicken will, muss man für die Symbole eine physikalische Umsetzung finden, also muss man Symbole durch **Signale** oder Folgen von Signalen physikalisch umsetzen. Ein Beispiel dafür ist ein Lichtsignal an einer Strasse. Es setzt die Information, ob ein Fussgänger gerade sicher die Strasse überqueren kann, oder nicht (z.B. durch die Symbole “0” und “1” digital dargestellt), physikalisch mit dem grünen respektive roten Licht um. Die **Signalverarbeitung** hingegen beschäftigt sich unter anderem damit, aus einem empfangenen oder gemessenen Signal, oder einer Folge von Signalen, Informationen zu extrahieren, zu verarbeiten, abzuspeichern oder für die Weiterversendung vorzubereiten.

Im Wesentlichen ist also eine **Signalfolge** nichts anderes als ein Paket oder ein **Strom** von Informationsträgern. Signale können in zwei Hauptformen unterteilt werden: In **kontinuierliche** und **diskrete**. Kontinuierliche Signale sind fortlaufend beziehungsweise lückenlos, während diskrete Signale nur zu bestimmten Zeitpunkten bestehen. Ein **Datenstrom** ist ein Strom solcher Signale, der über die Zeit „fliesst“, und meistens ist unbekannt für wie lange. Oftmals wenn man es mit kontinuierlichen Daten zu tun hat, müssen jene in ein digitales/diskretes Format umgewandelt werden, da die allermeisten Computer mit digitalen Daten sehr viel besser umgehen können. Diesen Vorgang bezeichnet man als **Abtastung**, da man das kontinuierliche Signal sozusagen an bestimmten Zeitpunkten „abtastet“ und nur jene Proben behält. Dann kommt auch die **Quantisierung** ins Spiel, wobei man sich entscheiden muss, welchen diskreten Wert man einer Probe zuweist, da Computer auf endliche Wertemengen beschränkt sind. Nachdem das Signal in digitaler Form vorliegt, kann man sie mithilfe von Computern besser verarbeiten.

Wenn man es mit Datenströmen zu tun hat, welche in **Echtzeit** fliessen, dann werden sogenannte **Datenstromalgorithmen** unverzichtbar. Datenstromalgorithmen unterscheiden sich von herkömmlichen Algorithmen in dem, dass sie oftmals striktere Limitierungen bezüglich **Speicherkomplexität** und **Rechenzeit pro Eingabezeichen** haben, da es meistens unpraktisch oder sogar unmöglich ist, den gesamten Datenstrom abzuspeichern bevor man ihn verarbeitet. Sie haben also meistens jeweils nur einen Teil der Signalfolge zur Verfügung und können den auch in Echtzeit bearbeiten. Natürlich gibt es sehr viele verschiedene Algorithmen, ein Beispiel wären **Filteralgorithmen**, die **Störsignale** anhand eines **Schwellenwertes** herausfiltern können (Störsignale können zum Beispiel durch Hintergrundgeräusche bei einem Gespräch übers Mobiltelefon entstehen).

Die Signalverarbeitung und Datenstromalgorithmen sind grosse Themen mit vielen Begriffen und neuen Konzepten. Die Idee ist ein Grundverständnis aufzubauen wie heute mit den Unmengen an Daten umgegangen wird und erste Bausteine zu legen, um fortgeschrittenere Konzepte der Signalverarbeitung und Datenstromalgorithmen erkunden zu können.

Vorwissen: Aus dem Fachbereich der Informatik gehen wir davon aus, dass die lernenden grundlegende Programmierkenntnisse besitzen - sie sind mit Arrays (bestenfalls auch mehrdimensional) vertraut und wissen, wie man Schleifen und If/Else verwendet. Wir gehen aber davon aus, dass die SchülerInnen noch nicht viel Übung haben wenn es um das Entwerfen & Implementieren von eigenen Ideen geht. Ausserdem sollten sie schon mit einigen einfachen Konzepten der Statistik in Berührung gekommen sein. So sollte die Klasse bereits wissen, wie man den Durchschnitt berechnet. Weiteres Vorwissen in der Statistik wie den Median und auch die Standardabweichung könnte man hier ebenfalls sehr gut reaktivieren und in Übungsaufgaben einbauen. Aus der Mathematik wäre es sicher von Vorteil, wenn auch schon das Umgehen mit Summen bereits sitzt. Da das vorausgesetzte Vorwissen meist erst zu Beginn des Gymnasiums eingeführt wird, eignet sich dieses Material eher für die letzten beiden Jahre des Gymnasiums.

## b) Leitidee

Signalfolgen als Träger von Daten sind heute allgegenwärtig. Somit ist die Signalverarbeitung auch ein sehr breites Gebiet der Informatik und riesige Mengen an Daten werden mehr und mehr zu einem Problem. Ein Einstieg in die Datenstromalgorithmen soll erste Grundlagen in der Signalverarbeitung aufbauen und Konzepte näherbringen, die in unzähligen Teilgebieten von Signalverarbeitung relevant sind. Zudem können die Signale so gewählt werden, dass SchülerInnen einfache, aber effiziente Echtzeit-Algorithmen mehr oder weniger selbst entwickeln können. So wird nebenbei auch das Entwerfen eigener Programme geübt sowie das Optimieren, kritische Beurteilen und Bewerten jener.

## c) Dispositionsziele

Die SchülerInnen verwenden den Begriff des Signals und der Signalfolgen nach der Unterrichtssequenz als (physikalische) Träger von Informationen. Die SchülerInnen wenden verschiedene vorgestellte Grundkonzepte der Echtzeit-Datenstromalgorithmen an und entwerfen oder verbessern selbständig einfache Algorithmen, um gewünschte Merkmale aus Signalen in Echtzeit auslesen zu können.

## d) Operationalisierte Lernziele

Die SchülerInnen können aus dem Stand drei verschiedene Anwendungsbereiche von Datenstromalgorithmen & Signalverarbeitung aufzählen und in eigenen Worten begründen.

Die SchülerInnen geben bei einem gegebenen Datenstrom einen sinnvollen Vorschlag, was für Informationen oder statistische Merkmale sie aus dem Datenstrom extrahieren würden und wie sie versuchen würden das in Echtzeit zu schaffen.

# 2 Leitprogrammartige Unterrichtsunterlagen

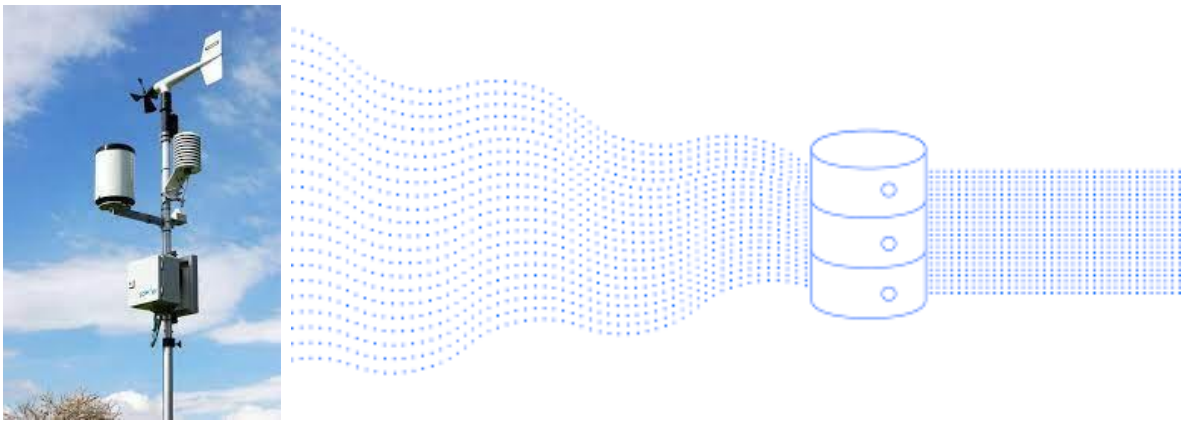
## Informierender Unterrichtseinstieg:

Um die Welt um uns herum besser zu verstehen, ist es oft notwendig, sie genau unter die Lupe zu nehmen. Beispielsweise sind unzählige Wetterstationen überall auf der Welt verteilt, die mit vielen teuren Messgeräten so viele Daten wie möglich sammeln. Solche Daten werden oft irgendwo hingeschickt, wo sie dann verarbeitet oder abgespeichert werden. Damit man Informationen oder Daten verschicken kann, setzt man die Informationen oder gemessenen Daten meistens physikalisch um, zum Beispiel in Radiowellen. Solche physikalischen Träger von Informationen nennt man auch Signale. Die können auch so einfach sein wie das bekannte Lichtsignal an Strassenkreuzungen. Aber wenn man sehr viele Informationen hat, wie bei der vorhin erwähnten Wetterstation, dann muss man ständig sehr viele Signale senden und dann auch empfangen. In diesen Fällen spricht man dann eher von sogenannten Echtzeit-Datenströmen, da fortlaufend neue Signale reinströmen und in fix begrenzter Zeit verarbeitet werden müssen. Solche Echtzeit-Datenströme sind heute überall, hinter jedem Online-Shop und jedem sozialen Netzwerk werden riesige Mengen an Daten in Echtzeit aufgenommen und verarbeitet. Das Sammeln oder Verarbeiten solcher riesigen Datenmengen fällt oft unter den Sammelbegriff Big Data. Big Data kann helfen, um aus riesigen Datenmengen Informationen zu gewinnen die man mit herkömmlichen Methoden früher nicht zur Verfügung gehabt hätte. So wird wie in den Wetterstationen zur Verbesserung des Verstehens des Klimas und deren Veränderung, in Zukunft vielleicht im grossen Stil Daten über Herzschläge, Schritte oder Aktivitäten von Menschen über Smartwatches gesammelt, um möglicherweise Gesundheitsrisiken frühzeitig zu erkennen.
Diese Fortschritte bringen aber auch Probleme mit sich, denn die Unmengen an Daten müssen dann auch sinnvoll verarbeitet werden, sonst nützen sie uns nur sehr wenig. Genau da kommen die Datenstromalgorithmen ins Spiel, und wir werden sehen, wie man mit deren Hilfe in Echtzeit sinnvolle Informationen aus Datenströmen und Signalen rausholen kann. Wir werden mit ein paar sehr einfachen Beispielen anfangen und dann auf schwierigere Probleme übergehen und unsere Datenstromalgorithmen schrittweise verbessern.

Unter anderem werden wir betrachten, wie wir mit wenig Speicherplätzen gewisse Informationen oder statistische Merkmale von Datenströmen gewinnen können und wie wir sicherstellen können, dass dies auch in Echtzeit möglich ist.

# Aufgabensammlung zu Datenstromalgorithmen und Signalverarbeitung

## 1. Wetterstation - Messungen in einem Fixen Zeitraum



In einer Wetterstation werden fortlaufend verschiedene Messungen durchgeführt, welche ausgewertet werden müssen. Dies kann eine grosse Menge an Messwerten, also Daten erzeugen, welche ausgewertet werden müssen. Aber wie unterscheiden sich nun die Messdaten einer Wetterstation von Resultaten eines Fussballturniers?

*Mögliche Antworten:*
*-Das Fussballturnier ist irgendwann fertig und man braucht meistens nur sehr wenige Daten abzuspeichern (zum Beispiel die Tore oder verteilten Karten).*

Eine Wetterstation produziert Messdaten, die über einen längeren Zeitraum reinströmen und es handelt sich meistens um eine sehr grosse Menge an Daten. Wobei es allein in einer Stunde der Fall sein kann, dass mehr Messwerte erzeugt worden sind als abgespeichert werden können. Würde die Station pro Sekunde 500 MB Daten speichern, wie viele wären es bereits nach einer Stunde (Wie viele Smartphones bräuchte man, um das abzuspeichern)?

*Lösung: Eine Stunde hat 3600 Sekunden, daher wären dies 3600x500MB = 1'800'000MB, was 1,8 TB sind. Das wären mehr als 14 Smartphones, wenn man annimmt, dass ein Smartphone 128 GB Speicherplatz hat.*

Zum Lösen solcher Probleme können sogenannte **Online-/Datenstromalgorithmen** eingesetzt werden. Sie unterscheiden sich von "klassischen" Algorithmen, oder auch **Offlinealgorithmen,** in dem**,** dass sie nicht auf die gesamte Eingabe gleichzeitig zugreifen können. Sie bekommen die Eingabe “stückweise”, und können aufgrund von Speicherproblemen nach einiger Zeit nicht mehr auf früher erhaltene Eingaben zurückgreifen. Die Aufgabe der Online/Datenstromalgorithmen ist es also, aus den stückweise erhaltenen Signalfolgen die wichtigsten Merkmale in Echtzeit rauszuholen, bevor das nächste Eingabestück eintrifft. Onlinealgorithmen eignen sich also, um eine grössere Menge an Daten, die auch ständig reinströmen können, zu verarbeiten.

Aufgabe 1.1
Ein Thermometer A liefert alle 30 Sekunden die Temperatur gemessen in Kelvin. Dabei handelt es sich um einzelne Float Werte.

a) Wie viele Werte sind dies an einem Tag?

b) Überlegen Sie sich einen Online-Algorithmus, der die maximale Temperatur pro Tag ausgibt, mit möglichst wenig Speicherplatz auskommt. Besprechen Sie Ihre Ideen mit Ihrem Tischnachbar und schreiben Sie den Algorithmus in Pseudocode auf.

c) Wie viele Variablen benötigen sie für Ihren Algorithmus und warum? Geht es noch besser? Wieviel Operationen benötigt er für den ersten Messwert? Wie viele Operationen müssen nach jeden weiteren gelesenen Messwert ausgeführt werden? Wenn wir zudem die minimale Temperatur auch noch wissen möchten, wie viel braucht er dann?

d) Wenn wir zudem die minimale Temperatur auch noch wissen möchten, wie viele Variablen braucht dann der Algorithmus insgesamt? Und wie viele Operationen werden pro Messwert ausgeführt (lesen von Werten, Wertzuweisungen und Vergleiche) ?

## *Lösung:*

a) *2×60×24 = 2880*

b) *Lösungsansatz: Da wir an der oberen Aufgabe alleine am Maximalwert interessiert sind, können wir alle neuen Daten ignorieren welchen kleiner sind als der momentane Maximalwert der bereits eingelesenen Daten. Wir brauchen einen zusätzlichen Speicherplatz für das Minimum aus ähnlichen Gründen.*

```
def max_funct(I:InputStream):
    #nehme erste Wert als max
    max = I.readline()
    for x in range(2879):
        # jeder Wert  wird als temp eingelesen.
        temp = I.readline()
        # Falls grösser als max wird max erstzt
        if temp > max:
            max = temp
    return max
```

c) *Um die Maximale Temperatur zu bestimmen, benötigen wir nur eine Variable plus je nach dem eine temporäre Variable, falls wir den Input Wert zwischenspeichern müssen. Für den ersten Messwert benötigt es eine Operation, für alle anderen Messwerten drei Operationen benötigt.*

d) *Wenn wir zusätzlich auch noch die minimale Temperatur bestimmen möchten, dann benötigen wir eine zusätzliche Variable, also zwei bis drei. Für den ersten*

*Messwert brauchen wir dann zwei Operationen. Für alle anderen werden drei bis vier Operationen benötigt.*

Aufgabe 1.2:

Das Luftfeuchtigkeitsmessgerät liefert alle 30 Sekunden den Messwert der aktuellen Luftfeuchtigkeit. Nun wollen wir wissen, wie hoch die durchschnittliche Luftfeuchtigkeit pro Tag ist.

a) Wie könnte man den Durchschnitt berechnen, wenn wir n+1 Speicherplätze zur Verfügung haben? Wobei n die Anzahl Messwerte sind.

b) Wenn n aber sehr gross ist, muss man so sehr sehr viele Daten speichern. Geht das noch besser? Kann der Durchschnitt mit k vielen Variablen bestimmt werden, wobei k unabhängig von der Stromlänge n ist? Überlege dir einen Online-Algorithmus und schreibe in Pseudo Code auf.

*Lösung:*

*a) Lösungsansatz: Speicherplätze können verwendet werden, um die Daten abzuspeichern und nach einer Stunde die Messdaten anschliessend aufaddieren, um am Ende des Tages durch die Anzahl zu teilen.*

```
def average_funct(I:InputStream):
    avg = 0
    for x in range(2880):
        avg = avg + I.readline()/2880
    return avg
```

*b) Es geht tatsächlich mit einem Speicherplatz fürs Abspeichern des Resultats plus zwei temporäre Variablen zur Berechnung. Dazu werden vier Operationen pro Messwert benötigt.*
*Da wir wissen, dass wir insgesamt 2880 Werte messen werden pro Tag, können wir jeden Messwert durch 2880 teilen und in einer Variable addieren. Mit diesem Algorithmus hat man zwar am Ende des Tages einen Durchschnittswert, aber auch nicht vorher. Während des Tages hat man keinen aktuellen Durchschnittswert.*

## 2. Wetterstation – Messungen in einem Unbestimmten Zeitraum

Bis jetzt hatten wir einen fixen Zeitraum mit einer fixen Anzahl Messungen. Nun nehmen wir an, wir möchten gerne wissen, wie sich die gesuchten Werte nicht nur am letzten Tag oder Stunde verhält, sondern dass wir zu jedem Zeitpunkt den aktuellen Durchschnittswert haben, seit Tagesbeginn, seit Jahresbeginn… und nicht erst am Ende des Messzeitraumes, welches unbekannt sein kann.

### Aufgabe 2.1:

Überlegen Sie sich für die Algorithmen aus dem vorherigen Abschnitt, ob Sie sie auch verwenden können, um…

a) Um fortlaufend die höchste Temperatur seit Jahresbeginn anzugeben?
b) Um fortlaufend die durchschnittliche Luftfeuchtigkeit seit Jahresbeginn anzugeben?
c) Worin unterscheiden sich diese beiden Aufgaben?
d) Nehmen wir an, wir haben den Durchschnitt nach zwei Werten, haben aber die einzelnen Werten nicht gespeichert. Wenn wir nun einen dritten Wert messen, wie können wir den neuen Durchschnitt berechnen?
e) Wie können wir von einem Durchschnitt von 1000 Werten und einem neuen Messwert den Durchschnitt von 1001 Messwerten berechnen?
f) Können wir nun mit Hilfe der Teilaufgabe d) und e) einen Algorithmus finden, welcher nach jeder Eingabe eines neuen Messwertes den aktuellen Durchschnitt ausgibt, ohne alle Messwerte abzuspeichern?

### *Lösung 2.1:*

*a) Ja, der Algorithmus des Lösungsvorschlages funktioniert immer noch.*

*b) Nein, in diesem Fall funktioniert der Algorithmus nicht mehr.*

*c) Der Unterschied besteht darin, dass das Maximum unabhängig von der Anzahl Messwerten der Durchschnitt aber nicht. Wenn wir die Formeln betrachten, dann wird für erstere die Anzahl Werte nicht benötigt, jedoch für letztere.*

*d) Sei $d_2$ der Durchschnitt von zwei uns mittlerweile unbekannten Werten $x_1$ und $x_2$ also ist $d_2 = (x_1 + x_2) / 2$. Nun messen wir einen dritten Wert $x_3$. Der Durchschnitt von drei Werten wäre $d_3 = (x_1 + x_2 + x_3) / 3$, wir kennen aber $x_1$ und $x_2$ nicht mehr, sondern nur $d_2$.Wir brauchen also eine Formel für den Durchschnitt. Wir benötigen eine Formel in der nur $x_3$ und $d_2$ vorkommen. Wir können die Formel umformen zu*

$$d_3 = ((x_1 + x_2) / 2 + (x_1 + x_2) / 2 + x_3) / 3$$
$$d_3 = (d_2 + d_2 + x_3) / 3$$
$$d_3 = (2d_2 + x_3) / 3$$

*Für diese Formel kennen wir nun alle Werte und können somit den Durchschnitt von drei Werten berechnen.*

*e) Hier können wir im Prinzip gleich vorgehen wie in Teilaufgabe d):*
*Der Durchschnitt von 1001 Werten wäre:*
$d_{1001} = (x_1 + x_2 + ... + x_{1000} + x_{1001}) / 1001$
*nun sind uns aber wiederum die Werte $x_1, x_2, .., x_{1000}$ nicht bekannt. Dafür kennen wir $d_{1000}$ welches von der Form ist:*
$d_{1000} = (x_1 + x_2 + ... + x_{1000}) / 1000$ *daher*
$1000d_{1000} = (x_1 + x_2 + ... + x_{1000})$ *und somit*
$d_{1001} = (1000d_{1000} + x_{1001}) / 1001$

*f)* *Lösungsansatz:*
*Die Berechnungen der vorherigen Aufgabe kann verallgemeinert werden: Wenn $d_n$ der Durchschnitt von n Werten ist, so ist $n \times d_n = (x_1 + x_2 + \ldots + x_n)$ da $d_n = (x_1 + x_2 + \ldots + x_n)/n$ ist. Wenn wir nun einen neuen Wert messen, $x_{n+1}$, dann können wir den neuen Durchschnitt berechnen mit $d_{n+1} = (n \times d_n + x_{n+1})/(n+1)$. Dies können wir so in einen Online-Algorithmus umwandeln (siehe Pseudocode). Für diesen Algorithmus benötigen wir zwei dauerhafte Speicherplätze für Durchschnitt und Anzahl berücksichtigter Werte. Sowohl als auch zwei weitere temporäre Variablen. Pro Messwert benötigen wir sechs bis sieben Operationen.*

```
def average_funct(I:InputStream):
    avg = I.readline()
    count = 1
    for x in I:
        avg = (avg*count + I.readline())/(count + 1)
        count = count + 1
    return avg
```

Was genau haben wir jetzt in der letzten Aufgabe gemacht, wenn wir es allgemein betrachten? Wir haben das Resultat genommen, dass wir für m Werte erhalten haben. Haben die Operation umgekehrt, welche von der Anzahl abhängig war, den neuen Wert aufgenommen und die neu Anzahl Operation angewendet.

# 3. Online Candy Shop – Signale höherer Dimension

Bis jetzt besteht unser Signal/ Datenfluss aus einer Abfolge von einzelnen Werten also in der Form von f(x)=y wobei x der Messzeitpunkt ist und y einen einzelnen Wert. Aber ein Datenfluss kann auch aus einer Abfolge von Tupels, also ( a , b) oder aber auch Arrays oder mehrdimensionalem Array sein. Nehmen wir das Beispiel eines Videos als Signal, so kann das Bild, welches am Zeitpunkt x aufgenommen wurde, als Messwert y betrachtet werden.

Nun schauen wir uns erst ein einfacheres Beispiel an. Nehme sie an, ein Online Candy Shop verkauft verschiedene Bonbon Sorten. Die Webseite ist so programmiert, dass die Daten im sogenannten Registrierkasse Modell übermittelt werden.
Schauen wir es noch anhand eines Beispiels an:

**Beispiel**:
Bei unserem Candy Shop trifft für jede Bestellung ein Satz von Tupel ein die wie folgt aussehen: (123, 4), (133, 1). Dies kann man wie folgt verstehen: Das Bonbon mit der Identifikationsnummer 123 wurde in einer Bestellung vier Mal bestellt und das Bonbon mit der Identifikationsnummer 133 wurde einmal bestellt. Die erste Zahl des Tupels identifiziert also das Produkt und die zweite gibt die Stückzahl an.
Nun können wir pro Bonbon entweder ein Zähler verwenden oder ein Array, der zählt wie viele Bonbons verkauft worden sind:
Z. B. wenn der Zähler $A_{123}[i] = 234$ ist und das Tupel (123,4) eintrifft, so ist er neu $A_{123}[i+1] = 238$. Wir können auch die Zähler durch einen Array ersetzen, dies könnte wie folgt aussehen: $A_{123}$=[0, 10, 14, 15 …, 234, 238]

Ein Datensatz sieht formal wie folgt aus:

$$(j, I_i)\ ,\ I_i>0\ ;\ \ A_j[\ I_i] = A_j[\ I_{i-1}] + I_i$$

## Aufgabe 3.1:

Im Beispiel haben wir die Anzahl verkaufter Bonbons gezählt. Nehmen wir die Zähler vom vorherigen Beispiel. Wie finden wir mit Hilfe dieser Zähler heraus, wie viele Bonbons wir pro Sorte noch auf Lager haben, vorausgesetzt wir kennen die Anfangszahl?

## *Lösung 3.1:*

*Wir speichern die Zähler der Bonbon in einem Array ab, so dass das Bonbon mit Nummer 123 im 123 Feld des Arrays ist, dasselbe machen wir mit der Anfangszahl. Nun können wir die Verkauften Bonbons vom Anfangsbestand abziehen und erhalten die Anzahl Verbleibende Bonbons.*

## Aufgabe 3.2:

Diese Art von Zähler erfasst Informationen über verschiedene Produkte aus einem Datenstrom mit einem Array der Länge der Anzahl Produkte – was für ein Candy Shop immer noch sehr lange sein kann. Jedoch, jede Reduktion von Daten auf einen geringeren Speicherplatz führt zu Verlust von Informationen. Beurteile, ob folgende

Informationen aus den Zählern gewonnen werden können. Und falls nicht, welche Informationen sonst noch gespeichert werden sollten.

a) Welches Produkt wurde am meisten verkauft?
b) Wie hoch ist die Stückzahl des Produktes X im Schnitt pro Verkauf?

c) Wie gross ist die durchschnittliche Stückzahl, in der ein Produkt bestellt wird. (Beachte nur die Fälle, in denen ein bestimmtes Produkt verkauft wurde)

## *Lösung 3.2:*

*a) Diese Information kann aus den Zählern gewonnen werden.*
*b) Diese Information kann nicht allein aus den Zählern gewonnen werden. Dazu würden wir noch einen Zähler benötigen, welche die Anzahl Verkäufe zählt. Hierzu würden wir eine Variable für die Anzahl Bestellungen benötigen.*

*c) Auch diese Information kann nicht aus den Zählern gewonnen werden. Hierfür bräuchte es ein Zähler pro Produkt, der die Anzahl Verkäufe zählt, in dem dieses Produkt vorhanden gewesen war. Hierzu würden wir ein Array benötigen, in dem für jedes Produkt die Anzahl Bestellungen gezählt werden.*

# 4. Der fehlende Gast

Ein Alien Superstar will ein grosses Fest für das ganze Universum organisieren. Jeder Bewohner des Universums ist eingeladen und hat eine eindeutige Zahl zwischen 1 und k zugewiesen (der Alien Superstar hat die Zahl 0). Wobei k die Gesamtanzahl der Bewohner des Universums ist.

Leider weiss er nicht genau wie gross k ist, aber das interessiert den Alien Superstar auch nicht wirklich, vielmehr möchte er wissen, ob alle gekommen sind. Er hatte die Idee, dass er am Eingang, wo alle Gäste hineinströmen, einfach eine Liste hinhängt, wo alle Gäste ihre zugewiesene Nummer eintragen können, am Schluss könne er dann schauen, ob eine Zahl fehle. Er hat zum Glück eine Freundin, die ihm davon abriet, denn es ist unmöglich so eine riesige Liste zu führen, geschweige denn nach der fehlenden Zahl darin zu suchen, oder es würde zumindest eine Ewigkeit dauern. Sie hatte aber eine clevere Idee. Sie wollte aber nicht gleich alles verraten und gab nur als Tipp, dass er sich die Gausssche Summenformel zu Hilfe nehmen soll.

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Am Tag des Festes stand er mit einem Lächeln im Gesicht am Eingang, da er wusste, dass er sich nur 2 Variablen merken muss. Aber er musste jeden neu ankommenden Gast um seine zugewiesene Zahl fragen.

## Aufgabe 4.1:

a) Was hat er sich gemerkt und wie viel musste er rechnen (Anzahl Rechenoperationen)?

Als niemand mehr in der Schlange war, hatte er die beiden Variablen x und y und stellte mit Erschrecken fest, dass $\frac{x\cdot(x+1)}{2}$ nicht gleich gross wie y war. Er war ausser sich, aber eine vertrauenswürdige Quelle versicherte ihm, dass nur genau ein Gast nicht gekommen war.

b) Welche Nummer hatte der fehlende Gast?

*Lösung 4.1*

*a) Er hat sich die Anzahl der Gäste gemerkt, die bereits angekommen sind, wir nennen diese mal x. Und die Summe der jeweiligen zugewiesenen Zahlen der angekommenen Gäste merkt er sich auch noch, die nennen wir y. So kann er am Schluss überprüfen mit Hilfe der Gaussschen Summenformel ob $\frac{x\cdot(x+1)}{2}$ den gleichen Wert ergibt wie y. Wenn ja, dann weiss er, dass alle Gäste mit den Zahlen von 1 bis x anwesend sind.*

*Er braucht auch nicht viel zu rechnen, jeweils eine Addition für die Variable x, und eine Addition für die Variable y, für jeden neuen Gast, der ankommt.*

*x wird zu x + 1*

*y wird zu y + id, wobei id die eindeutige Zahl ist, die jeweils jedem Bewohner des Universums zugeordnet ist.*

b) *Da anscheinend nur ein Gast fehlt, muss das heissen, dass nur eine einzige Zahl in der Summe fehlt. Somit muss der Gast genau die Zahl haben, die der Differenz zwischen* $\frac{x\cdot(x+1)}{2}$ *und y entspricht.*

# 5. Zusammenfassung

Zur Berechnung und Speicherung von globalen Merkmalen von Datenströmen können Online-/Stream Algorithmen verwendet werden. Dabei ist der benötigte Speicherplatz unabhängig von der Länge des bisherigen Datenstroms. Dabei können gewünschte Merkmale nach jeder neuen Eingabe aktualisiert werden.

Datensätze und Signale können unterschiedliche Formate haben. Es kann sich dabei um einen einzelnen Wert handeln, ein Array von Werten oder auch ein mehrdimensionaler Array an Werten wie zum Beispiel Bilder. Wie beim Beispiel einer Wildlife Kamera, können ein Signal oder ein Datenstrom auch über längere Zeit keine Informationen enthalten, die wir speichern möchten. Hierfür können Schwellenwert eingesetzt werden, um nur Daten abzuspeichern, die von Interesse sind.

# 6. Kontrollfragen

1. Was ist ein Datenstrom?
2. Wie unterscheiden sich Onlinealgorithmen von Offlinealgorithmen?
3. Spielt es für Merkmale wie Minimum und Maximum, ob wir die Gesamtanzahl der Datensätze kennen oder nicht?
4. Wenn ich den Durchschnitt für n Werte habe und n kenne, und einen neuen Wert erhalte, wie kann ich den neuen Durchschnitt berechnen?
5. Sollte ich zuerst entscheiden, was ich speichern möchte oder welche Merkmale mich interessieren?
6. Wenn ich meine 1000 Lamas nummeriert habe, zähle ich aber nur 999, muss ich sie zuerst der Reihe nach ordnen, um herauszufinden, welches Lama fehlt?

Lösung:

1. *Ein kontinuierlicher Fluss an Datensätzen. Wessen Ende nicht zwingend bekannt sein muss.*
2. *Offline-Algorithmen kennen von Anfang an alle Eingabewerte. Online-Algorithmen bekommen die Eingabe stückweise und reagieren auf jede Eingabe.*
3. *Nein, es spielt keine Rolle. Wenn $x_i$ das Maximum ist, ist der Wert das Maximum, egal wie viele andere Werte existieren. Solange kein anderer Wert grösser ist.*
4. *In dem ich den alten Durchschnitt mit n multiplizieren, den neuen Wert dazu zähle und den erhaltenen Wert durch n+1 teile.*
5. *Es macht Sinn, sich zuerst zu überlegen, welche Merkmale man haben möchte. Sonst könnte es sein, dass man diese Merkmale gar nicht mehr berechnen kann, da die benötigten Informationen nicht in den gespeicherten Daten enthalten sind.*
6. *Ja, es geht schneller, indem ich die vorhandenen Lamas nicht nur zähle, sondern ihre Nummer aufaddiert. Diese addierte Nummer kann von 500500 subtrahiert werden und wir erhalten die Nummer des verlorenen Lamas.*

## 7. Kontrollaufgaben

1. Vervollständigen sie den folgenden Pseudocode! Es kommt jede neue Sekunde ein neuer Wert von einer Smartwatch über einen Datenstrom rein. Und zwar handelt es sich dabei immer entweder um eine Null oder eine Eins. Eine Null bedeutet, dass in der gerade vergangenen Sekunde kein Herzschlag von der Smartwatch verzeichnet wurde. Bei einer Eins hingegen wurde ein Herzschlag verzeichnet. Sie sollen jetzt einen Datenstromalgorithmus entwerfen, der jede Minute (also immer nach 60 gesammelten Werten) den Puls in Herzschlägen pro Minute ausgibt.

```
def heart_rate(I:Inputstream):

    while (not(I.is_empty())):
        counter = 0
        heart_beats = 0
        while(counter< ):



        print("Der Puls ist:" + str(heart_beats) + "pro Minute")
```

*Lösung & Erklärung: Wir zählen die Anzahl Werte die reinkommen mit der Variable counter. Jedes mal, wenn ein Wert rein kommt schauen wir ob in der letzten Sekunde ein Herzschlag verzeichnet wurde oder nicht, wir zählen nur die Anzahl verzeichneten Herzschläge mit der Variable heart_beats. Sobald wir 60 Werte gesammelt haben, sprich, 60 Sekunden verstrichen sind geben wir den Puls aus und danach, wird counter und heart_beats wieder auf 0 gesetzt und wir beginnen wieder von Neuem.*

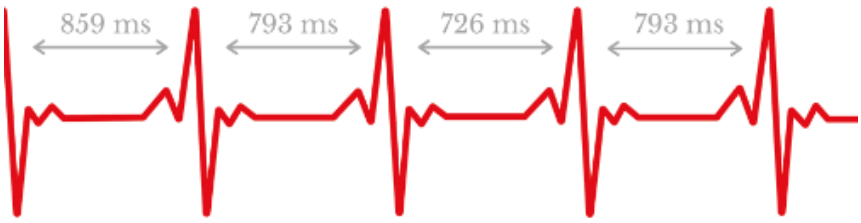
```
def heart_rate(I:Inputstream):

    while (not(I.is_empty())):
        counter = 0
        heart_beats = 0
        while(counter<60):
            heart_beat = I.pop()
            if heart_beat == 1:
                heart_beats +=1

            counter += 1

        print("Der Puls ist:" + str(heart_beats) + "pro Minute")
```

2. Die sogenannte Herzfrequenzvariabilität ist ein Indikator für das Stresslevel, deswegen kann es sinnvoll sein, jenen zu überwachen. Um die Herzfrequenzvariabilität zu bestimmen, misst man die Zeit zwischen 2 Herzschlägen und schaut, wie stark dieser Zeitabstand variiert.




Genauer möchten wir den Root-mean squared error berechnen, der ein Mass für diese Variation der Abstände zwischen zwei Herzschlägen geben kann. Dieses Mal bekommen wir alle 10 Millisekunden ein neues Update, ob gerade in den letzten 10ms ein Herzschlag verzeichnet wurde, und wir wollen wieder immer jede Minute den RMSE (in Milisekunden) der Abstände zwischen zwei Herzschlägen überprüfen.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

$\hat{y}_i$ Ist der Durchschnitt der zeitlichen Abstände zwischen zwei nachfolgenden Herzschlägen, $y_i$ ist der i-te gemessene Abstand und n ist hier die Gesamtanzahl der gemessenen Abstände zwischen 2 Herzschlägen im Zeitinterval von einer Minute.

Wie würden sie vorgehen? Welche Werte müssen sie Zwischenspeichern und welche nicht? Können wir im Vorhinein wissen, wie viele Werte wir abspeichern müssen?

Wir möchten noch zusätzlich Einbauen, dass das Programm eine Warnung ausgibt, wenn der berechnete Wert unter 30ms fällt.

*Lösung:*
*Um den RMSE zu berechnen über ein Zeitfenster von 60s = 60000ms, müssen wir (da wir alle 10ms einen neuen Herzschlagwert bekommen) jeweils 6000 Werte «abwarten», bevor wir den RMSE berechnen können. Wir müssen aber keinesfalls alle Werte abspeichern. Sobald wir eine 1 sehen, zählen wir die folgende Anzahl Nullen bis wieder eine 1 kommt, dann Speichern wir die Anzahl Nullen zwischen den zwei Einsen ab und setzen den Zähler wieder auf Null. Wir können also im Vorhinein nicht genau wissen wie viele Werte wir abspeichern müssen, aber wir können sicher sein, dass es sicher weniger als 6000 sind, sehr wahrscheinlich viele weniger, da niemand alle 10ms ein Herzschlag hat. Sobald wir ins gesamt 6000 Werte bekommen haben, multiplizieren wir alle Werte mit 10 (da eine Null einen Abstand von 10ms bedeutet und wir ms als Einheit haben wollen), berechnen den Durchschnitt aller Zahlen in der Liste, also der durchschnittliche Zeitliche Abstand zwischen zwei Herzschlägen in ms. Danach berechnen wir die Differenz zwischen diesem Durchschnitt und den jeweiligen Werten in der Liste, quadrieren diese Differenz, teilen durch die Länge der Liste und nehmen zum Schluss die Wurzel. Wir haben den RMSE pro Minute, jetzt müssen wir nur noch schauen ob dieser Wert jeweils unter 30ms fällt, falls ja, schreiben wir mit print() eine Warnung auf die Konsole.*