# Titel

Namen der Gruppenmitglieder
&
Institutsname

**Kurzzusammenfassung**
Es soll ein möglichst optimales Lagersystem entstehen, wodurch man die bestmöglichste und kürzeste Route zu diversen Artikeln erhält. Dieses Lager ist systematisch angeordnet, sowohl, dass Kosten, als auch Zeit reduziert werden. Dabei sollte der menschliche Aspekt berücksichtigt werden, um mögliche Fehlgriffe zu vermeiden.

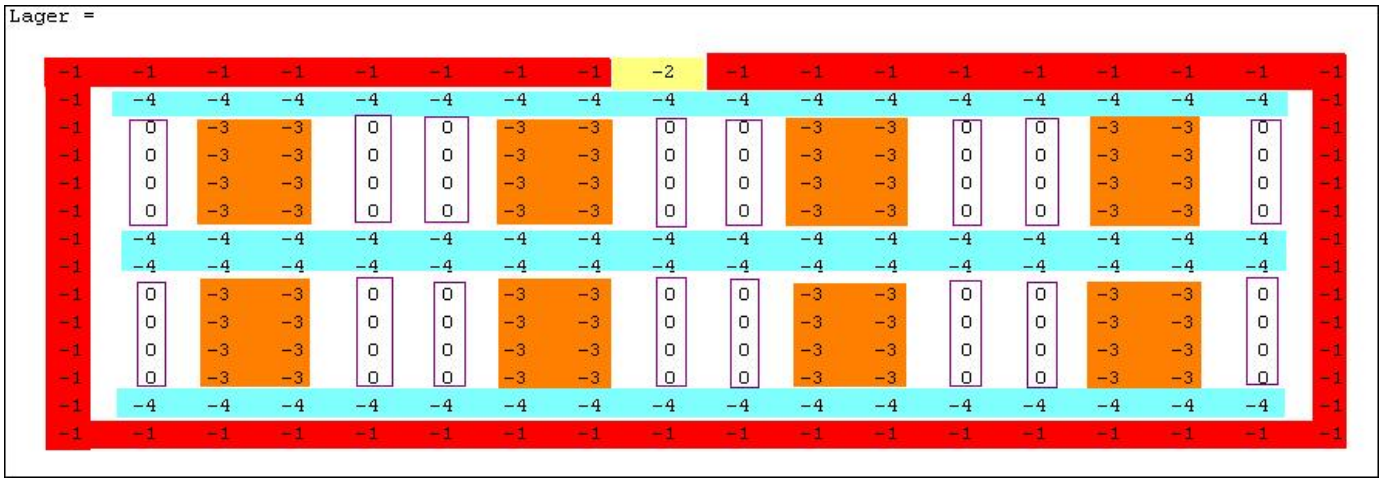
## 1 Aufgabenstellung

Der Versandhandel wächst im Gegensatz zum stationären Handel seit Jahr im zweistelligen Prozentbereich und wird somit auch immer wichtiger für Lösungsanbieter. Die Ware des Versandlagers muss zunächst aus dem Lager geholt werden und per Postpaket an den Kunden zu gesand werden. Aufgrund dieses Wachstums soll ein möglichst effizientes Lagersystem erstellt werden, welches die Route des Lagerarbeiters zu den gewünschten Waren optimiert, somit die Kosten-, sowie den Zeitaufwand reduziert.

Abbildung 1: Lieferant

Ein Lager besteht aus mehreren Gängen, dort stehen Regale in denen die Artikel des Warenhändlers gelagert werden.
Unsere Aufgabe bestand darin, die optimalste Lösung zur Warenanordnung zu finden, dabei werden die Faktoren Zeit, Kosten und Menschlichkeit berücksichtigt.

## 2 Daten




Legende: rot = Wand ; gelb = Tür; türkis = Quergang; orange = Regale; lila = Artikel
Der Raum beträgt ein rechteckiges Format, dabei sind die Regale reihenweise, horizontal angeordnet. Sowohl das Lager, als auch die Maße der Regale können beliebig erweitert und verkleinert werden.

## 3 Zufallsdaten

Um unser Sortiersystem zu testen brauchten wir verschiedene Daten mit denen wir das Programm speisen konnten. Allerdings sollten diese Daten nicht nur zufällig sondern so realitätsnah wie möglich ermittelt werden. Das heißt, dass gewisse Artikel mit einer größeren Wahrscheinlichkeit als andere und Artikel mit verschiedenen Wahrscheinlichkeiten von Kombinationen mit anderen gekauft werden.

Um den ersten Schritt zu erreichen haben wir eine Array der Länge der Artikel erstellt in welcher zufällig verschieden Werte zugewiesen wurde. Diese Werte spiegeln die Wahrscheinlichkeiten wieder, wie oft der jeweilige Artikel gekauft wird. Anschließend wird die Summe der Wahrscheinlichkeiten genommen und dann eine zufallszahl in diesem Bereich ermittelt. Die Wahrscheinlichkeiten werden nacheinander geprüft und addieren bis die jetzige Summe größer oder gleich der Zufallszahl ist.

Währenddessen gibt es ein Array welches in Länge und Breite so groß ist wie die Artikelanzahl. Es beinhaltet die Wahrscheinlichkeit wie oft Artikel in Kombination mit anderen auftreten. Die Wahrscheinlichkeiten hierzu werden nach gleichem Prinzip erstellt so wie die Kombinationen selber (Artikel der zu dem ersten gekauft wurde. Wieder nach einer Zufallszahl wird ermittelt wie viele Artikel pro Bestellliste gekauft werden und danach entscheidend wird dieser Vorgang wiederholt. Als letztes werden die Bestelllisten bzw. ihre Werte (also die eingekauften Artikel) ausgegeben.

## 4 Komplexitätsuntersuchung

Unser Ansatz bestand darin, einen Vergleich zwischen unserem Algorithmus und einer alternativ Möglichkeit zu schaffen.
Dabei werden alle Möglichkeiten durchgegangen, indem man die Fakultät der vorhandenen Artikelanzahl (n) berechnet.
Ab einer Fakultät von 8 sieht man, dass die Kurve exponentiel wächst und somit die Möglichkeiten ab einer Fakultät von 8 sehr stark ansteigen.
Ab einer bestimmten Artikelanzahl (n) ist die zu berechnende Datenmenge so groß, dass sogar der schnellste Computer der Welt (Tianhe2) mehr Jahre für die Brechnung in Anspruch nimmt, als es Sterne in der Milchstraße gibt.

Anzahl aller Konfigurationen:
$f(n) = n!$
Annahme: Die Anzahl an Artikeln $n$ beträgt 100 und das Programm benötigt einen Flop zur Berechnung einer Konfiguration, so ergeben sich

$$\frac{9.3326 \cdot 10^{156}}{33 \cdot 10^{15} \cdot 60 \cdot 60 \cdot 60 \cdot 24 \cdot 365} =$$


$$8,97 \cdot 10^{129} \text{ Jahre}$$

zur Berechnung aller Konfigurationen auf einem Supercomputer. Zum Vergleich: Das Universum ist 13,75 Milliarden Jahre alt, also

$$13,75 \cdot 10^{9} \text{ Jahre.}$$

# 5 Vorgehensweise unseres Algorithmus

## 5.1 Benötigte Funktionen

Die Entfernung zwischen 2 Punkten im Lager musste berechnet werden. Hierfür wurde ein Algorithmus benötigt der Alle möglichen Lagen der beiden Punkte zueinander berücksichtigt. Diese 16 möglichen Lagen konnten erfolgreich bestimmt werden und somit ein Algorithmus programmiert werden der auch bei Hindernissen immer den optimalen weg fand.

## 5.2 Einsortierung

Vorbereitung:
Die Kaufhäufigkeit der Artikel wird berechnet. Außerdem wird die Wahrscheinlichkeit der Kombination zweier Artikel berechnet. Ist diese Wahrscheinlichkeit höher als ein vorgegebener Grenzwert sprechen wir von einer Verbindung.

Schritt1:
Die Artikel werden nach ihrer Kaufhäufigkeit behandelt. Der meist gekaufte Artikel wird zu erst in das Lager eingesetzt.
Schritt2:

Es wird geprüft welches der der Tür nächste freie Lagerplatz ist. Alle Lagerplätze die näher liegen als die Entfernung des nächsten Lagerplatzes addiert zu einem Toleranzwert werden als mögliche Positionen für den Artikel angesehen.
Schritt3:
Jeder der möglichen Positionen wird ein Wert zugeordnet. Dieser setzt sich zusammen aus der Entfernung zum Eingang und der Entfernung zu Objekten zu denen der Artikel eine Verbindung hat.
Schritt4:
Der Artikel wird auf die Position mit dem tiefsten Wert gesetzt und somit Optimal einsortiert.

# 6 Route finden

Um die Effizienz des Warenverteilungs-Alhorithmus zu testen wurde ein Algorithmus benötigt mit welchem alle Artikel mit der Effizientesten Route verbunden werden.
Mit dem Algorithmus wurde das Ziel verfolgt einen Test zu finden der uns am Ende die benötigten Schritte eines Arbeiters für eine bestimmte Anzahl an Bestellungen mit variierenden Artikelzahlen ausgibt. Hierbei war zu beachten, dass wir einen möglichst genauen Algorithmus benötigen, damit die Ergebnisse des Vergleichs unserer verschiedenen Varianten des Verteilungsalgorithmus nicht durch ein ungenaues Ergebnis verfälscht werden.

## 6.1 Optimaler Weg

Der erste Algorithmus dient dazu die optimale Route auszugeben.
Das hierzu verwendete Programm legt zunächst eine Matrix an, in der die Entfernung aller im Lager befindlichen Artikel zu einander und zur Tür gespeichert sind. Im nächsten Schritt erstellt das Programm eine Matrix in der die Permutationen der Bestellung gespeichert sind, um eine Liste aller möglichen Reihenfolgen der Artikel zu erhalten.
An diese Liste der Routenmöglichkeiten wird nun noch vor und hinter jedem Weg die Tür gespeichert. Um nun die Länge der Wege zu überprüfen greift das Programm auf die Entfernung der in der Routenliste jeweils nebeneinander stehenden Artikel zu und addiert sie. Nach der Überprüfung der gesamten Wegeliste wird die kleinste Summe der Artikelentfernungen mit ihrer Reigenfolge der Artikel ausgegeben. Der Vorteil dieser Vorgehensweise ist, dass die optimale Route gefunden wird um eine Bestellung ab zu laufen. Es entsteht jedoch das Problem, dass der Algorithmus eine sehr große Laufzeit hat, da er alle Möglichkeiten überprüft.

## 6.2 Eigene Suche

Der zweite Algorithmus wurde entwickelt um trotz vieler Artikel pro Bestellung dennoch eine akzeptable Laufzeit zu garantieren.
Das Programm ordnet hierzu die Bestellung zunächst nach X-Koordinate der Artikel und teilt diese Liste in eine rechte/linke Liste.
Diese Listen ordnen wir nun auf der rechten Seite nach aufsteigenden Y-Werten und auf der linken nach absteigenden Y-Werten. So erzeugt das Programm eine Reihenfolge, nach welcher die Produkte in einer Kreisform aufgesammelt werden wobei der Start- und Endpunkt unsere Türe ist.
Der Algorithmus könnte noch verbessert werden indem man in jedem Regal, in welchem man schon einen Artikel abholt noch alle weiteren dort befindlichen Artikel, unabhängig von ihrer Position in der Liste, mitnimmt. Außerdem sollte man bei steigender Artikelzahl bzw. Lagergröße die Anzahl der Teillisten erhöhen,jedoch sollte die Anzahl der Teillsten gerade sein um die Grundidee der Kreisbewegung beizubehalten. Es ist zu erwarten, dass dieser Algoritmus bei kleinen Werten schlechtere Ergebnisse als der Algorithmus zur Findung des optimalen Weges auswirft. Jedoch sollte er bei größeren Werten besser werden, es ist dieser Trend zu erkennen jedoch konnten wir ihn aufgrund fehlender Rechenleistung nicht testen.

# 7 Ergebnisse