AUFGABENSTELLUNG

ALLGEMEINE INFORMATIONEN

Die Aufgabenstellung ist als *Einzelarbeit* innerhalb von 2,5 Stunden zu lösen. Wie im realen Programmieralltag ist es erlaubt, das Internet zur Recherche zu verwenden. Bei Fragen zur Problemstellung, wende dich bitte an einen der KNAPP-Betreuer.

EINLEITUNG

KNAPP zählt zu den führenden Technologieunternehmen für Automatisierungslösungen und Software für Distribution und Produktion. Kunden auf der ganzen Welt aus unterschiedlichen Branchen wie zum Beispiel Gesundheitswesen, Onlinehandel, Lebensmittelhandel, Mode- und Textil-Branche bis hin zum Automotive-Bereich vertrauen auf die intelligenten Lösungen von KNAPP. Jede dieser Branchen hat dabei ihre Besonderheiten, denen KNAPP durch maßgeschneiderte Systemlösungen gerecht wird.

In einem Verteilzentrum werden einzelne Kundenaufträge in Kartons zusammengestellt. Am Ende der Bearbeitung wird das Paket für den Versand vorbereitet und die Etiketten aufgebracht.

Die fertigen Pakete werden im so genannten Versandbereich für die einzelnen Paketdienste auf Paletten für die Abholung durch Lastwägen bereitgestellt.

Um die Beladung der Paletten auf die LKW zu ermöglichen und auch Platz zu sparen gibt es Regeln zur Beladung der einzelnen Paletten.

Deine Aufgabe ist es nun, den Algorithmus für die Stapelung von Packstücken auf Paletten zu implementieren.

Die Information über alle Packstücke steht von Anfang an zur Verfügung. Diese müssen auf Paletten gestapelt werden. Dabei ist zu beachten, dass nur Packstücke für denselben Lastwagen auf eine Palette geladen werden, und dass auch die Einschränkungen der Paletten eingehalten werden da ansonsten zusätzliche Kosten entstehen.

Die Aufgabe ist eine Optimierungslösung, versuche zuerst die Aufgabe zu lösen und dein Ergebnis danach Schritt für Schritt zu verbessern.

ABLAUF

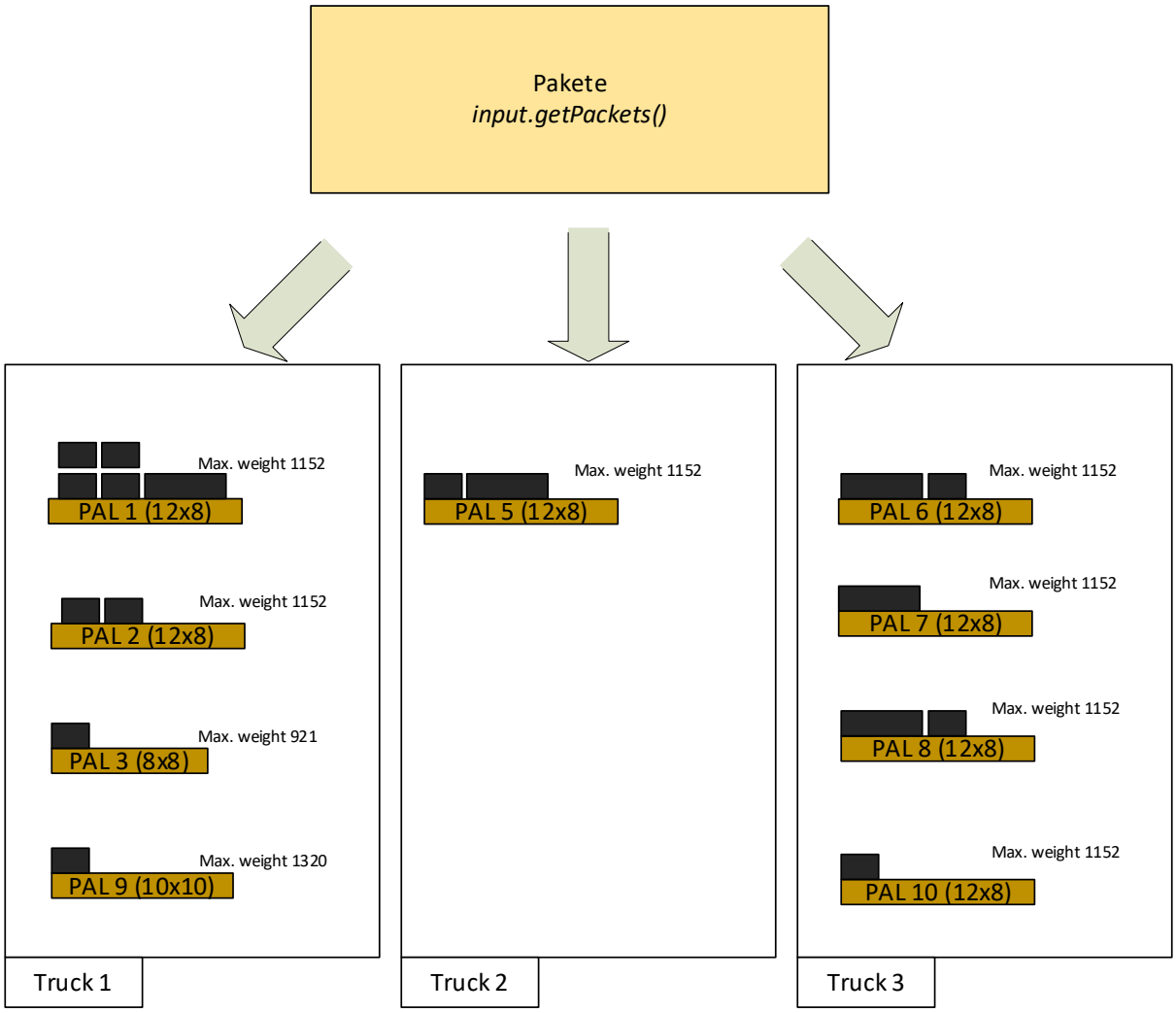


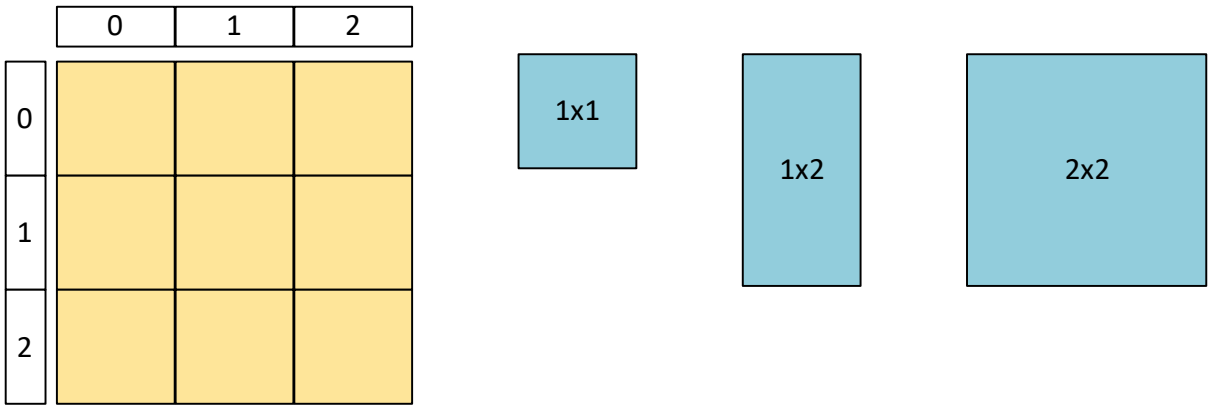
Abbildung 1 – Struktur

Für den Algorithmus stehen von Anfang an alle Packstücke (*input.getPackets()*) und die Palettentypen (*input.getPalletTypes()*) zur Verfügung.

Um Packstücke für einen LKW zu stapeln benötigt ihr mindestens eine Palette. Diese müsst ihr zuerst vorbereiten. Mit *warehouse.preparePallett()* erstellt ihr eine Palette eines bestimmten Typs und ordnet diese einem Lastwagen zu.

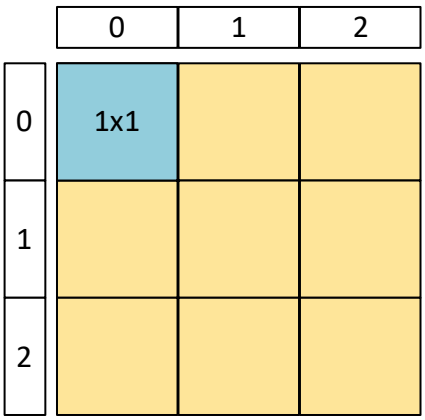
Nun könnt ihr unter Einhaltung des Maximalgewichts und der maximalen Stapelhöhe beliebig viele Pakte mittels *warehouse.putPacket()* auf diese Palette stapeln. Ihr legt das Paket dabei auf eine bestimmte Position und könnt das Paket auch drehen.

Hier ein Beispiel:



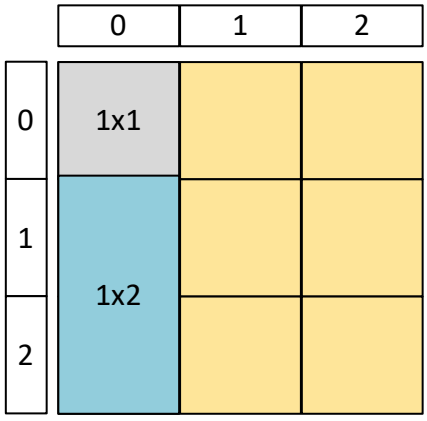
Auf eine Palette mit den Abmessungen 3x3 sind drei Packstücke zu stapeln. Es kann nun das erste Paket gestapelt werden:

putPacket(4711, 0,0, false)

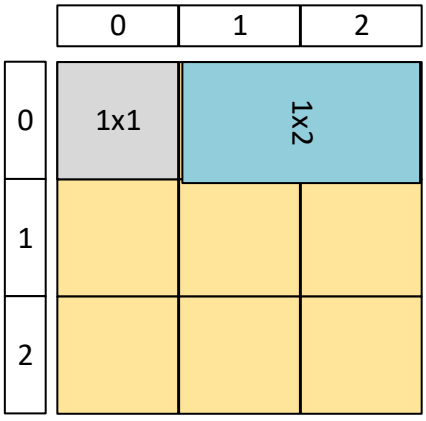


Das Packstück 4711 wird auf die Position 0,0 gelegt und dabei nicht gedreht. Für das zweite Packstück sind zum Beispiel sind folgende Varianten möglich:

putPacket(4712, 0, 1, false)

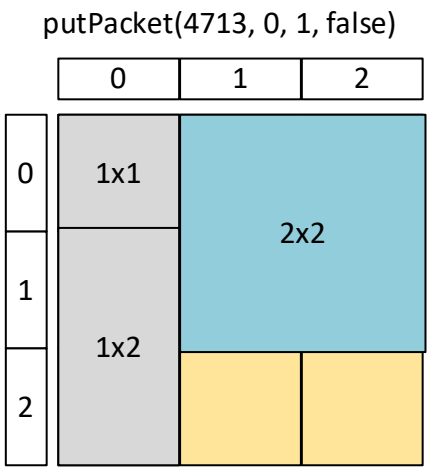


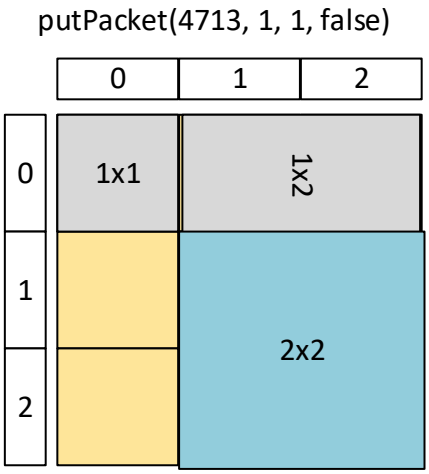
putPacket(4712, 1, 0, true)



Das Packstück 4712 kann an Position 0,1 ungedreht, oder an Position 2,0 gedreht gelegt werden.

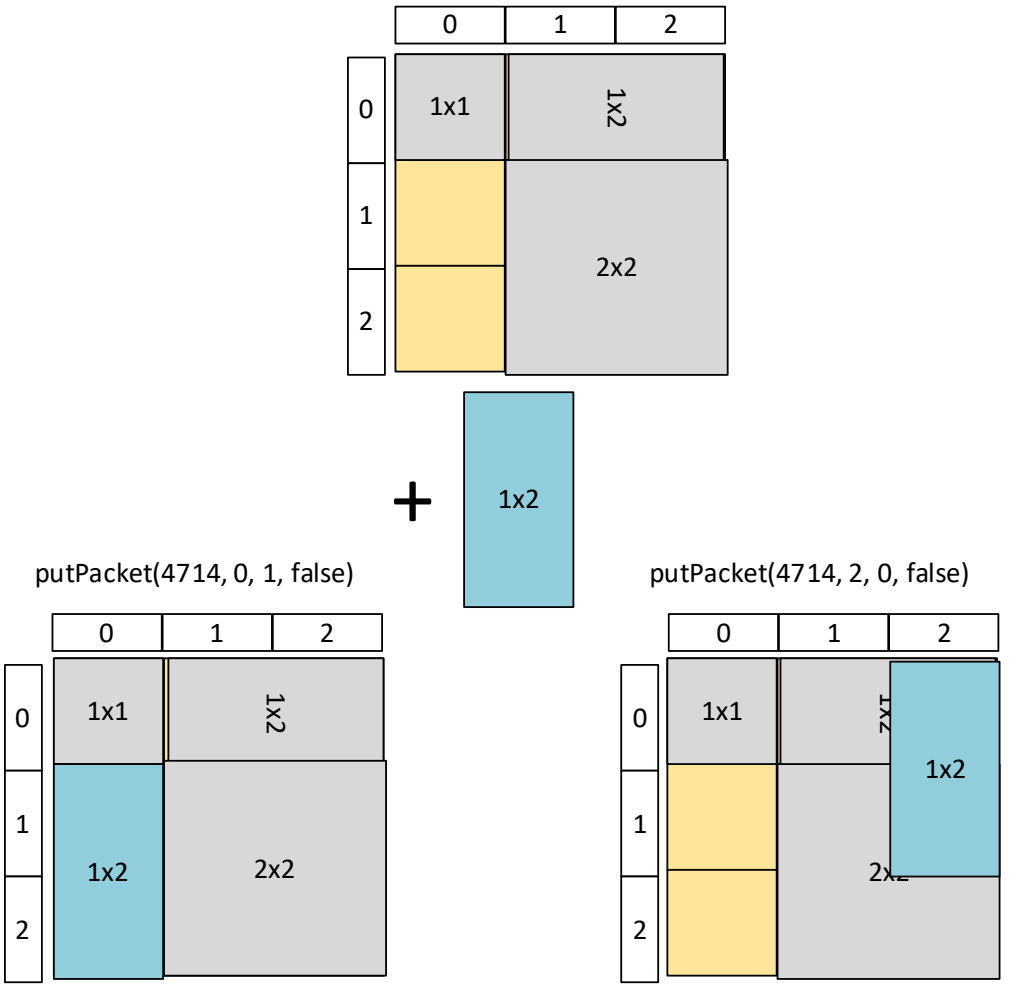
Das letzte Packstück kann zum Beispiel auf folgende Positionen gelegt werden:





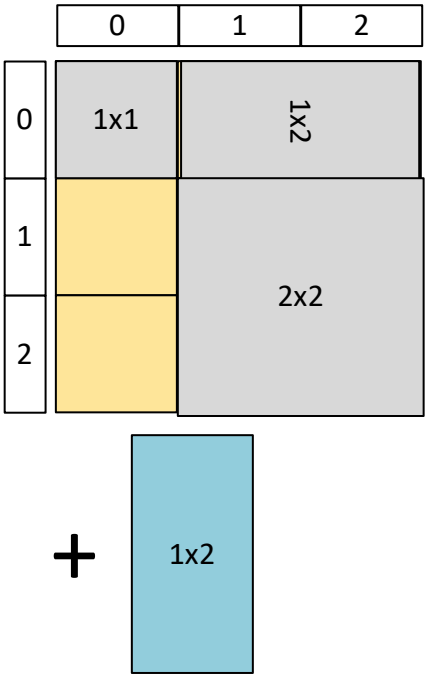
Wenn die Packstücke wie oben beschrieben abgelegt wurden, ist die Stapelhöhe 1 und damit die Anzahl der Ebenen (Layer) ist 1.

Wird nun ein weiteres Packstück gestapelt, so gibt es (unter anderem) folgende Möglichkeiten:

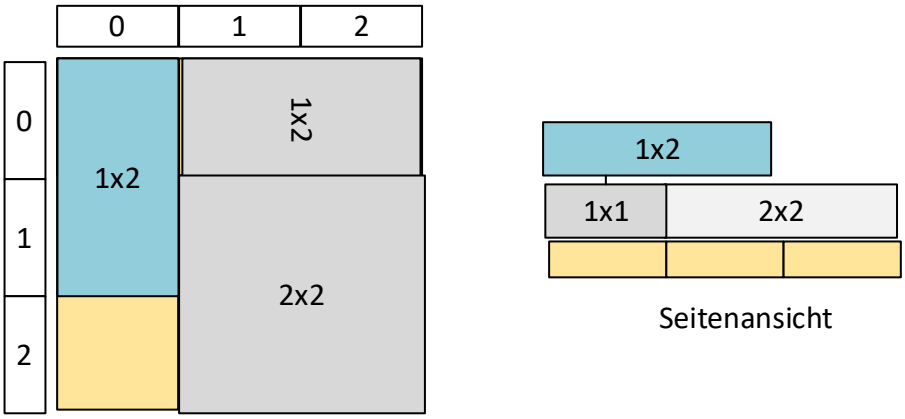


Bei der Lösung auf der linken Seite wird die unterste Ebene voll ausgenützt, das ist daher eine gute Lösung. Bei der Lösung auf der rechten Seite wird eine neue Ebene begonnen. Beide Lösungen sind gültig.

Ihr beginnt eine neue Ebene, indem ein Packstück auf eine Position gelegt wird wo darunter bereits ein Packstück liegt. Hierbei reicht es bereits, wenn ein Teil des Packstücks auf einem Packstück darunter aufliegt. Die Palette ist immer stabil, es wird nicht geprüft ob Löcher entstehen:



putPacket(4714, 0, 0, false)



Bei dem Beispiel darüber entsteht auf Ebene 1 auf Position 0,1 ein Loch. Das Stapeln ist gültig, jedoch unter Umständen nicht die beste Variante.

Die Pakete werden immer von oben abgelegt, es besteht danach keine Möglichkeit mehr eine entstandene Lücke zu füllen.

Anmerkung:
In diesem Beispiel wurden nur die Abmessungen erläutert, das Maximalgewicht muss zusätzlich beachtet werden.

Nach Aufruf der Funktionen *preparePallet* und *putPacket* sind die Daten in der Lösung abgelegt, es sind keine weiteren Funktionsaufrufe mehr notwendig.

WICHTIGE METHODEN

warehouse.preparePallet(int truckId, PalletType type)

Bereitet eine Palette eines Typs für de Beladung für einen LKW vor.

Warehouse.putPacket(Pallet pallet, Packet packet
, int x, int y, boolean rotated)

Stapelt ein Paket auf eine Palette auf die angegebene Position.
Die Muss-Kriterien sind einzuhalten.

ERLÄUTERUNGEN

Lastwagen (Truck)

Mit einem Lastwagen werden die Pakete weitertransportiert. Der Lastwagen ist keine eigene Entität sondern ist über die TruckId am Packstück und der Palette dargestellt.

Palettentyp (PalletType)

Ein Palettentyp beschreibt den Typ einer Palette, die beladen werden kann. Ein Palettentyp hat

- Id, eindeutig für den Palettentyp
- Länge (Length)
- Breite (Width)
- Maximale Höhe (MaxHeight)
- Maximales Gewicht (MaxWeight), maximales Gewicht der Pakete auf der Palette

Palette (Pallet)

Auf einer Palette werden die Packstücke gesammelt, um dann in einen Lastwagen verladen zu werden.

Eine Palette hat

- Id, eindeutig im Lager
- Typ (PalletType)
- TruckId, Id des Lastwagens dem die Palette zugeordnet wurde

Für die Palette gilt

- Es dürfen nur Packstücke für den Lastwagen gestapelt werden, dem diese Palette zugeordnet ist.

Packstück (Packet)

Ein Packstück ist die Einheit, die an einen Kunden versandt wird und die bestellten Produkte des Kunden beinhält. Ein Packstück muss auf eine Palette verladen werden.

Ein Packstück hat

- Id, eindeutig für jedes Packstück
- Länge (Length)
- Breite (Width)
- Gewicht (Weight)
- TruckId, LKW mit dem das Packstück weitertransportiert wird.

Für jedes Packstück gilt:

- Jedes Packstück hat eine Höhe von 1.
- Jedes Packstück ist ein Quader.
- Jedes Packstück passt zumindest auf einen Palettetyp, jedes Packstück kann damit versendet werden.
- Jedes Packstück kann rotiert werden.
- Jedes Packstück ist über die gesamte Fläche voll belastbar.

ALLGEMEINES

- Alle Pakete stehen von Anfang an zur Verfügung.
- Die Reihenfolge in der gestapelt wird ist beliebig.
- Es gibt keine Beschränkung der Anzahl der Paletten.
- Die Ergebnisdatei wird automatisch durch die Funktionen
    - *preparePallet*
    - *putPacket*

  erstellt.

MUSS-BEDINGUNGEN

- Kein Packstück darf über die Palette hinausragen.
- Es dürfen nur Packstücke auf eine Palette gelegt werden, die für den Lastwagen bestimmt sind, dem die Palette zugeordnet ist.
- Die maximale Stapelhöhe für die Palette muss eingehalten werden.
- Das maximale Gewicht der Packstücke auf einer Palette muss eingehalten werden.
- Ein Packstück kann nur einmal abgelegt werden.

BEWERTUNGSSCHEMA

- Es werden nur auf den Bewertungsserver hochgeladene Ergebnisse gewertet.
- Die Resultate werden am Server, mit den dort hinterlegten Algorithmen, errechnet.
- Die Punkte errechnen sich aus
    - der Anzahl der gebauten Paletten (weniger ist besser)
    - dem Gesamtvolumen der Paletten (weniger ist besser)
    - dem Abgabezeitpunkt (früher ist besser)

Die Gewichtung zwischen Abgabezeitpunkt und Kosten ist dabei so gewählt, dass in der Regel eine bessere Lösung auch bei späterer Abgabe ein besseres Ergebnis bedeutet.

Bei mehreren Abgaben (Uploads) zählt immer die beste Abgabe zu dem Zeitpunkt, an dem diese getätigt wurde. Wenn Du nach dem Upload einer Lösung weiterarbeitest und durch Deine Änderungen das Ergebnis schlechter wird, bleiben spätere Abgaben unberücksichtigt.

ABGABEMODUS

Zur Beurteilung des Ergebnisses muss die vom KNAPP-Code automatisch erzeugte Datei `upload2021.zip` über die Abgabeseite hochgeladen werden. In diese Datei wird dein Source automatisch mitverpackt. Dieser Source muss das Ergebnis erzeugt haben. KNAPP behält sich hier Kontrollen vor.

Du kannst alle Code Teile modifizieren, die Ergebnisdatei (`warehouse-operations.csv`) wird von uns interpretiert und darf daher im Format nicht verändert werden.

UPLOAD WEBSITE

Unmittelbar nach dem Upload erhältst du ein detailliertes Feedback zu deiner Abgabe.

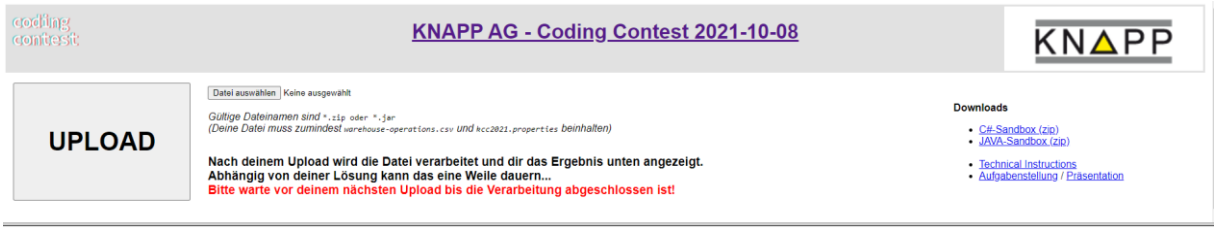


Abbildung 2: Upload-Server (Beispiel)

Im Feedback siehst du die Probleme, die noch in deiner Abgabe vorhanden sind, und deinen Score für diesen Upload. Je *niedriger* dieser Wert ist, desto besser ist das Ergebnis.

Abbildung 3 Abgabe - Details (Beispiel)

TIPPS

- Versuche mit deiner Software das Ergebnis zuerst mit einfachen Strategien zu verbessern und arbeite danach an weiteren Optimierungen.
- Schau dir den von uns vorgegeben Code an und prüfe, ob du Teile wiederverwenden oder erweitern kannst.
- Du kannst alle Teile des Source-Codes verändern. Die Abgabedatei muss jedoch dem vorgegebenen Format entsprechen.
- Lade öfters hoch - es wird nur die beste Abgabe bewertet.

CODE-DETAILS

- Name und Institution setzen
- Einsprungspunkt: `Solution::Run()`
- Die Klassen im Package `core` sind KNAPP Hilfsklassen und sind für die Lösung nicht von Bedeutung