To do lijst Heuristieken:

Vragen:

* Liever uit het mapje? Hoe dan?
* Random algorithm, wat maak je random?
* Datastructuur
  + Lijst/matrix met pakketjes, of pakketjes ook als object?
    - Gewicht, volume, ratio
      * Lage ratio in duur schip
      * Hoge ratio in goedkoop schip
    - Boolean; op het moment in gebruik
  + Algoritme suggereert mogelijke samenstellingen?
  + Bijhouden per object/spaceship
    - Current load massa optellen
    - Current load volume optellen
    - Current number of pakketjes optellen
    - Current total costs berekenen
  + Optellen voor objects/spaceships samen
    - Aantal pakketjes mee optellen
    - Totale kosten optellen
    - Hoeveel geld is een pakketje extra meenemen waard?
      * Eigenlijk is een oplossing pas nuttig wanneer alle pakketjes mee kunnen
* Infrastructuur
* Github opzetten
  + Data -> in te laden?
  + Scripts
  + Classes
  + Calculations
    - Totalen
    - Max en min tabel
  + Other
    - To do list
  + Readme
    - Voorbeeld in slides eerste college van Wouter
  + Requirements.txt
    - Waarvoor?
* Onder- en bovengrens bedenken
  + Ondergrens voor 4 lege schepen meenemen: 1.466.179.000
  + Bovengrens voor 4 schepen met maximum payload mass: 1.522.419.000
* Oriënteren algoritmes
  + Waarmee wil je beginnen?
    - Op massa zo goed mogelijk indelen zonder max volume te overschrijden?
    - Op volume zo goed mogelijk indelen zonder max massa te overschrijden?
    - Op beide tegelijk letten?
    - Beginnen met grote chunks?
  + Verder?
    - Zoals bij het uitproberen voorbijkwam:
      * Als het verschil tussen het aanvul pakketje wat net past en het pakketje wat net niet past hetzelfde of kleiner is dan het verschil tussen het laatste pakketje wat je erbij hebt gedaan en het pakketje daarna: switch.
    - Onthouden wat so far de optimale oplossing is (en meteen afbreken als je die kosten overschrijdt.
* Oplossingsruimte:
  + Elke oplossing:
    - Hoeveel pakketjes (van de 100), en welke pakketjes?
    - 100 pakketjes te verdelen over 4 schepen:
      * 4^100?
  + Problemen:
    - 1 pakketje heeft 4 opties, maar ze hangen willekeurig samen
    - Het is so far onbekend hoeveel pakketjes er daadwerkelijk maximaal in de 4 schepen samen passen
    - \
* Classes gemaakt hebben
  + Parcel
  + Spaceship
  + Distribution
* 24/4/18
  + Classes inventory af vanuit main
  + Random algoritme schrijven
* In main alleen aanroepen, verder alles netjes in functies (opdelen in kleinere functies)
* Lege ruimte proberen op te vullen
* Inventory gebruiken

Is het handig om kleine dingen ook in losse functies te zetten? (zoals reset)

* Bijhouden welke van de oplossingen met max aantal is het goedkoopst?
* Greedy op basis van ratios

Swappen: 2 pakketjes, of meerdere, of random paar weghalen en weer verder

Hillclimber uitgaande van de 10 beste oplossingen van onze random en greedy

Hillclimber met 1% kans op accepteren verslechtering mag ook (hoe?)

Environment en model

Reset functie wil je in je inventory class

Overlap in functies ofwel in class ofwel in helpers zetten

Dingen zo veel mogelijk in functie in class zetten, en anders in kleinere functies

(grote functies niet groter dan 30 regels ofzo)

Beam search 🡪 mix depth/breadth search en hill climber

Iets verzinnen waardoor je al halverwege een beetje weet of je in de goede richting bent?

Commandline: welke cargolist, welke restricties: politieke constraint, alle schepen of niet

Main.py alleen maar aanroepende code obv command line, verder geen uitvoerende code

Visualisatie: progress bars met daarin zichtbaar aantal (welke pakketjes) en hoeveel er over is aan ruimte (en met visueel bewegende pakketjes enzo)

Punten:

Opdelen van code naar functies en visualisaties

Of extra algoritme erbij

Greedy algemener laten werken

Inventory ipv dict\_ x2

En ook hillclimber met inventory maken 🡪 goed uitdenken vtv

Zeg tegen pakketje: ga naar spaceship: functie maken die dat erachter in de class aanpast

Visualisatie maken om op terug te vallen??

Lower bound en upper bound voor d en e

Random schepen kiezen en random vullen

Random en greedy voor d en e aanpassen

Hill climber en nog een algoritme (misschien constructief met heuristiek die veel elimineert)

Experimentatie voor progressiepunten maar niet grote prioriteit

Repository: opschonen; calculations is niet helemaal duidelijk en de code is lelijk

Scripts: niet allemaal zijn vergelijkbaar; splitsen naar algoritmes en hulp

En visualisaties apart

Onnodige dingen weg

Voorbeeld in de mail, en readme in een folder is ook mooi

Comments en style

Runnen vanuit main moet smooth zijn, zowel vanuit visualisatie als uit command line

En wel daar errors afvangen

Misschien optie om wel of niet visualisatie te gebruiken

Heel duidelijke instructies geven: in readme

Misschien met flags -c = hier komt welke cargolist je wil

Maandag eind vd dag mailen met wat we hebben toegevoegd en waar we punten op zouden kunnen krijgen

PVA:

* Julia en ons mergen
  + Scripts bestaat wel nog, klopt dat?
  + Nu blijft de main door runnen, niet het aantal keer uit de command line
  + Result is type dict, generator is type generator
* Random en greedy werkend voor d en e
  + Aan class toevoegen: de ratio van schip (m/v) om daarop te sorteren voor greedy
  + Command line arguments voor welke schepen en welke cargolist
    - Check of cargolists met alleen nummers
  + In readme uitleg geven over command line
  + Errors geven voor alle fouten in commandline
* Lower en upper bound voor d & e berekenen
* Hillclimber maken
  + Hoeveel pakketjes eruit? (random 1-10?)
  + Hoe vaak?
* Meer splitsen naar functies
  + Overeenkomende stukken uit random en greedy filteren
* Comments en style synchroniseren
* Klopt het dat er maximaal 2 verschillende getallen qua aantallen van verschillende schepen zullen uitkomen?

1250 (lengte cargolist 3)

Upper:

Base cost duurste schip \* 1250 + fuelcost(ieder pakketje’s gewicht in het duurste schip)

Lower: (wat je zoekt)

Totaalvolume en totaalgewicht van cargo 3 verdelen over schepen, maar welke?

2 kleine vs 1 grote?

Yield zorgt dat ie in het midden stopt 🡪 opdelen?

Niet meer eruit halen dan erin zitten 🡪 met een lege kunnen beginnen

Loop binnen hillclimber

Prints ook in functie als groot

Switches zijn wel mooier

Import sys 🡪 sys.path (maar liever niet)

Reset functie binnen inventory

Id’s van schepen en parcels gelijk trekken met +1 en -1