

<u>Help</u> shengtatng ∨

<u>Course</u> <u>Progress</u> <u>Dates</u> <u>Discussion</u> <u>Notes</u> <u>Calendar</u>





Next >



□ Bookmark this page

< Previous</pre>

Problem 2

1/1 point (ungraded)

Another way to transport the cows is to look at **every possible combination of trips and pick the best one**. This is an example of a brute force algorithm.

Implement a brute force algorithm to find the **minimum number of trips** needed to take all the cows across the universe in the function <code>brute_force_cow_transport</code>. The function returns a list of lists, where each inner list represents a trip and contains the names of cows taken on that trip.

Notes:

- Make sure not to mutate the dictionary of cows!
- In order to enumerate all possible combinations of trips, you will want to work with set partitions. We have provided you with a helper function called get_partitions that generates all the set partitions for a set of cows. More details on this function are provided below.

Assumptions:

- Assume that order doesn't matter. (1) [[1,2],[3,4]] and [[3,4],[1,2]] are considered equivalent lists of trips. (2) [[1,2],[3,4]] and [[2,1],[3,4]] are considered the same partitions of [1,2,3,4].
- You can assume that all the cows are between 0 and 100 tons in weight.
- All the cows have unique names.
- If multiple cows weigh the same amount, break ties arbitrarily.
- The spaceship has a cargo weight limit (in tons), which is passed into the function as a parameter.

Helper function get_partitions in ps1_partitions.py:

To generate all the possibilities for the brute force method, you will want to work with set partitions.

```
For instance, all the possible 2-partitions of the list [1,2,3,4] are [[1,2],[3,4]], [[1,3],[2,4]], [[2,3],[1,4]], [[1],[2,3,4]], [[3],[1,2,4]], [[4],[1,2,3]].
```

To help you with creating partitions, we have included a helper function [get_partitions(L)] that takes as input a list and returns a generator that contains all the possible partitions of this list, from 0-partitions to n-partitions, where n is the length of this list.

You can review more on generators in the Lecture 2 Exercise 1. To use generators, you must iterate over the generator to retrieve the elements; you cannot index into a generator! For instance, the recommended way to call get_partitions on a list [1,2,3] is the following. Try it out in ps1_partitions.py to see what is printed!

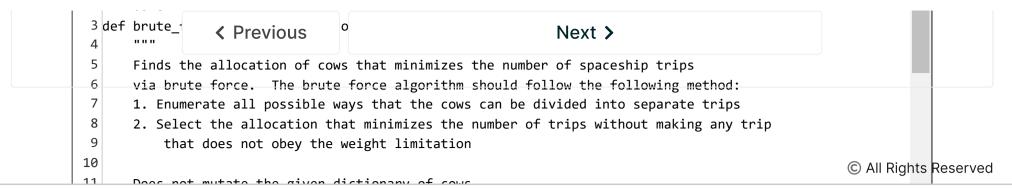
```
for partition in get_partitions([1,2,3]):
print(partition)
```

Example:

Suppose the spaceship has a cargo weight limit of 10 tons and the set of cows to transport is [{"Jesse": 6, "Maybel": 3, "Callie": 2, "Maggie": 5}].

The brute force algorithm will first try to fit them on only one trip, ["Jesse", "Maybel", "Callie", "Maggie"]. Since this trip contains 16 tons of cows, it is over the weight limit and does not work. Then the algorithm will try fitting them on all combinations of two trips. Suppose it first tries [["Jesse", "Maggie"], ["Maybel", "Callie"]]. This solution will be rejected because Jesse and Maggie together are over the weight limit and cannot be on the same trip. The algorithm will continue trying two trip partitions until it finds one that works, such as [["Jesse", "Callie"], ["Maybel", "Maggie"]].

The final result is then [["Jesse", "Callie"], ["Maybel", "Maggie"]]. Note that depending on which cow it tries first, the algorithm may find a different, optimal solution. Another optimal result could be [["Jesse", "Maybel"],["Callie", "Maggie"]].





edX

About

Affiliates

edX for Business

Open edX

Careers

News

Legal

Terms of Service & Honor Code

Privacy Policy

Accessibility Policy

Trademark Policy

<u>Sitemap</u>

Connect

<u>Blog</u>

Contact Us

Help Center

Security

Media Kit





Google play









© 2022 edX LLC. All rights reserved.

深圳市恒宇博科技有限公司 <u>粤ICP备17044299号-2</u>