Program the Pivot: Step 2: Instructions

Help

When you're ready to submit your solution, go to the assignments list.

Step Two (Pivot All the Way to Glory)

This will build on step one, where you have mastered the single step of pivoting by programming it. We will now go all the way and build a rudimentary LP solver. Ready to go? Test cases and inputs for this assignment are available as a bundle: zip or tar.gz .

Specification

The goal of this assignment is to read in a **feasible dictionary** (same format as step 1), and **repeatedly pivot** using the same pivoting code you wrote in Step #1, until it gets to a final dictionary with an optimal answer or an unbounded dictionary. You are expected to get step one correct before you attempt this assignment.

Output format for Step Two

If an optimal solution is found, the output is a text file with two lines as shown below:

```
[Line 1] Optimal solution obtained (we will accept an answer that differs from ours by upto 0.1)
[Line 2] Number of pivoting steps that were needed to get to a final dictionary.
```

If we detected that the dictionary was unbounded, the output is a text file with a single line:

```
UNBOUNDED
```

Examples

Example-1: We obtained an optimum of 14.1213 in 11 pivoting steps (which means that we will have seen 12 dictionaries including the one we start off with). The output file will contain

```
14.1213
11
```

Admin Help

Example-2: We obtained an unbounded answer in 15 pivoting steps. The output file will be

```
UNBOUNDED
```

Unit Tests for Step Two

The unit tests for this assignment are available in the bundle under the directory `unitTests` Just like in part one, you will find 10 inputs dict1..dict10. The solutions for these are as follows:

Dictionary	Optimal Objective Obtained	Number of Pivoting Steps
dict1	7	3
dict2	4	1
dict3	3	2
dict4	28	3
dict5	60	4
dict6	UNBOUNDED	
dict7	6	1
dict8	6.72952	2
dict9	0.27272	2
dict10	9.33227	18

Input Files for Step Two

The input files to use for the various parts are available under the directory `assignmentParts` We have five input cases for step two, each one larger than the other.

Part Name	Input File to Execute
1	`part1.dict`
2	`part2.dict`
3	`part3.dict`
4	`part4.dict`
5	`part5.dict`