Padam Mini-Hackathon #1

Knapsack problem

Friday, November 20th 2020

Problem statement

You are given `n` objects. Object `i` (for `1≤i≤n`) has a value v_i and a weight w_i.

You also have a bag that can carry a total weight up to `capacity`.

Your objective is to choose some of these `n` objects so that the sum of their values is as high as you can, and the sum of their weights is not bigger than `capacity`.

Resources

You should have received a zip file containing the relevant resources for you to participate. Unzip this file in your hackathon directory on your PC. You'll notice the following files:

- **main.py & helpers.py** : You should NOT modify these files. You can have a look at them but it will not help you.
- **instances.json** [only useful if you do NOT code in python] : This file contains 10 different instances (or example) of the problem. The format of each instance is detailed in the Input part of this pdf.
- **solutions_example.json** [only useful if you do NOT code in python] : This file contains an example solution to help you understand what kind of output is expected from your program.
- **knapsack_solver.py** [only useful if you DO code in python] : If you do the Hackathon in Python, you should do it IN THIS FILE ONLY. It contains only one function (which is documented) and your program should be in it. See Input for more information.

You are given 10 different sets of items (i.e. 10 variations of the problem) to solve. Depending on the output of your program, you'll be given a score, and teams will be ranked based on this score. Read Running your program and getting a score for more information on that.

Input

- **If you code in Python3:**
  - You only have to code in the knapsack_solver.py file, in the `get_knapsack_solution` function.
  - This function takes three input parameters:
    - weights: a list of n integers, representing the weight of each object.
    - values: a list of n integers, representing the values of each object.
    - capacity: an integer, representing the maximum amount of weight your bag can handle.
  - Note that the lists 'weights' and 'values' have the same length, which is the amount of objects you have to choose from.

- **If you code in any other programming language:**
  - The file instances.json contains some variations of the problem. The file is a mapping between the "id" of an instance and the instance itself. An instance of the problem is one dictionary containing the following key/value pairs:
    - "items": The amount of items in the problem (ranging from 50 to 10000).
    - "capacity": The maximum capacity your bag can handle.
    - "values": The list of values of each object. The length of this list is exactly equal to the amount of items.
    - "weights": The list of weights of each object. The length of this list is exactly equal to the amount of items.
    - "optimal_value" [useless for your hackathon]: The maximum attainable sum of values with respect to the given constraints. This is only used by our checker to compute your score (see Running your program and getting a score)
  - The file is small enough so you can open it in your IDE without crashing your PC.

Output

- **If you code in Python3:**
  - The code you'll implement in `get_knapsack_solution` should return only a list of boolean (True or False). If the i-th item of this list is True, that means you have chosen this object in your bag and its value and weight will be counted by our checker. Make sure that the objects you choose do not have a weight higher than the maximum capacity!

- **If you code in any other programming language:**
  - Your program should create a json file containing a mapping of solutions (corresponding to the instances given in Input). Each element of this mapping should be a pair with the following key/value:
    - key: Identifier of the instance you're giving a solution to.
    - value: list of integers. The list should have a length equal to the number of items of the instance, and its elements can only be 0s or 1s. If the i-th item of this list is a 1, that means you have chosen this object in your bag and its value and weight will be counted by our checker. Make sure that the objects you choose do not have a weight higher than the maximum capacity!
  - Opening the file solutions_example.json may help you understand the expected format.

Running your program and getting a score

- The only dependency of the checker script is `termcolor`. Run `pip install --user termcolor` and you should be good to go.
- Use our `python main.py` command to run our checkers and score computing script on your solutions. Run `python main.py --help` for more information.
  - IMPORTANT NOTE: The main.py command creates and maintains a file named `personal_bests` automatically. Everytime you run the command, it will check whether you improved your best score on some instance or another and update the personal bests file accordingly. You should not modify this file manually. **Before 11:15AM, send an email containing this file to alexis@padam.io AND louis@padam.io.**

    - For each instance, the command will tell you if your solution is correct (e.g. respects given constraints) and whether you improved or not.
    - You might find the optimal value on some instances of the problem. In that case (congratulations!) the command will tell you about it and it will not rerun the checkers on these instances, unless you pass the `--force` flag. The command should output instances where you have found the best possible value. That makes you a great thief!
    - The best possible score is 100. Getting a score of 99.5 does not guarantee you to win this contest!

- **If you code in Python3:**
    - As you can see in the given `get_knapsack_solution` function, the example algorithm will choose 1% of the objects at random and return the list.
        - Run `python main.py --all` to run this dummy algorithm and you should see its output for each variation of the problem, as well as the global score obtained by this strategy.

- **If you code in any other programming language:**
    - Once you have a working json output file, run `python main.py --all --solutions <your_json_file>` to get a score.
    - You can also run `python main.py --all --solutions solutions_example.json` in order to see an example output of the command.

Links & other information

- Link to the gather room: https://gather.town/app/HDD3eQ5q39earC2D/padam
- Link to the meet room: https://meet.google.com/mxh-rvuw-jyc
- If you encounter any problem, do not hesitate to contact Alexis or Louis using Slack or Gather. We will probably be able to help you in case there is something you do not understand or if you encounter a bug in the scripts we provided, although we will not give you hints about what may be a good or bad algorithm.

GOOD LUCK HACKERZ!