Challenge 2: The Knapsack Problem

In this lesson, we will go over another famous dynamic programming problem, the Knapsack problem.

We'll cover the following

A thief has broken into a house; the house has many valuable goods but unfortunately, the thief only brought a knapsack with a limited capacity. Every good in the house has a value in dollars and weight in kilograms associated with it. The thief wants to maximize the utility of his trip and take back the goods that fit his knapsack and earn him the highest possible money.

Problem statement

Given a list of weights and a list of costs, find the optimal subset of things that form the highest cumulative price bounded by the capacity of the knapsack.

Input

As input, your function will get a list of `weights`, a list of `prices`, and an integer `capacity` denoting the total weight the knapsack can carry. `weights` and `prices` are aligned to each other, i.e., weight and price of the first object are given by `weights[0]` and `prices[0]` respectively.

Sample input

```
weights = [1, 2, 4, 6]
prices = [4, 2, 4, 7]
```

```
capacity = 7
```

Output

Your function will return the maximum possible profit, i.e., the sum of prices of goods selected based on `weights` and `prices` bound by `capacity`.

Sample output

```
knapsack(weights, prices, capacity) = 11
```

Coding challenge

You should not change the signature of the given function; however, you may create a new function with a different signature and call it from the provided function.

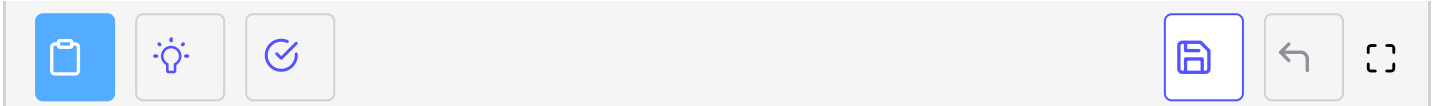
Try to think of a simple solution to this problem. Once you have implemented that, add the memoization to account for more complex problems.

You may check your simple solution approach first for correctness by testing with `stressTesting` set to `False`.

```
def knapsack(weights, prices, capacity):
  # your code goes here

  return 0

stressTesting = True
```



Hint 1 of 1

Remember in the optimal set of goods, a certain good may be present or it may be absent. Thus you have to explore these two possibilities for each good.

In the next lesson, we will look at some solutions to this problem.