Due Sunday Feb 6 11:59 PM EST

CSE 4256

Dr. Alan Weide

Spring Term 2022

Homework 3

Assignment

This assignment should be completed as a single Python source (.py) file modified from hw3-template.py (available on Carmen). Each question (including the Challenge Activities) has an associated TODO comment in the template file.

(1) Besides the syntax used to create them, what are three fundamental differences between a Python **set** and a Python **list**?

(2) Implement the function `all_unique(ls)` that, given a list of hashable objects `ls`, returns `True` whenever every element of `ls` is unique and `False` otherwise. **Hint**: this function can be written with a single line of code.

(3) Consider the following function `maybe_apply_all(s, f)`. Note the *default value* for the argument `f` is a lambda expression, which is essentially an anonymous function.

```python
def maybe_apply_all(s, f=lambda x: x + 5):
"""Applies function f to each element of s, updating s in place."""

  for x in s:
    x = f(x)
```

Is this function correct according to the docstring? If not, what's wrong?

(4) If it is not correct, write a correct implementation in the body of `apply_all(s, f)`.

(5) Implement the function `subset_sum(s, t)` that, given a set of integers `s` and a target `t`, returns `True` if and only if there is a subset of `s` whose sum is exactly `t`. That is, solve the Subset Sum Problem (or SSP). **Hint**: SSP is famously NP-complete, so there is no need

to spend time devising an efficient solution—the naive solution is just fine.[1] Recursion is a good tool for this job.

(6) Informally, a problem is in NP if there is an "easy" (*i.e.*, polynomial-time) algorithm to check whether a candidate solution to the problem is correct. Implement such a checker for SSP in the body of the function `is_subset_sum(sub, s, t)`.

Challenge Activities

Some homeworks (such as this one) will have additional challenge activities. These activities **do not contribute to your grade**, but they are problems that I find interesting or challenging.

(1) There are several polynomial-time approximation algorithms for SSP on its Wikipedia page. Implement one of them in the body of the function `subset_sum_approx(s, t)`.

(2) There exists a Pseudo-polynomial time algorithm to solve SSP using dynamic programming. Solve SSP using dynamic programming in the body of function `subset_sum_dyn(s, t)`. **Hint**: A Dictionary is a good tool for this job. While we haven't yet discussed Dictionaries in class, you can learn all sorts of things about them online (*e.g.*, at `https://docs.python.org/3/tutorial/datastructures.html#dictionaries`).

(3) There are several clever algorithms that improve on the runtime of the naive algorithm to solve SSP, often at the expense of space. One such algorithm is the Horowitz and Sahni algorithm. Implement that algorithm in the body of the function `subset_sum_h_s(s, t)`. You can look online for some more detailed notes about the algorithm.

Submission

To submit this assignment, upload your modified `hw3-template.py` file containing all of your code and question answers to the "Homework 3" assignment on Carmen. If there are test cases in your modified file, please comment them out before uploading it. As always, be sure to note all group members who contributed to the assignment and what those contributions were.

[1]That said, if you happen to come up with a novel efficient solution, you should publish it and win a Turing Award or something.