Problem Set 8

All parts are due on November 15, 2019 at 6PM. Please write your solutions in the LaTeX and Python templates provided. Aim for concise solutions; convoluted and obtuse descriptions might receive low marks, even when they are correct. Solutions should be submitted on the course website, and any code should be submitted for automated checking on `alg.mit.edu`.

Please solve each of the following problems using **dynamic programming**. For each problem, be sure to define a set of subproblems, relate the subproblems recursively, argue the relation is acyclic, provide base cases, construct a solution from the subproblems, and analyze running time. Correct but inefficient dynamic programs will be awarded significant partial credit.

Problem 8-1. [15 points] **Peter Piper**

Peter Piper wants a peck of pickled peppers, so he planted a patch of n pepper plants in a line. The i^{th} pepper plant possesses a known positive integer p_i number of peppers. Peter Piper didn't plant his peppers properly, and now his poor plants are over-packed. Peter Piper plans to pick and pickle some peppers prematurely so the unpicked plants can prosper. For each plant, Peter Piper will either leave the plant **unpicked**, or **prune** the plant and pick all its peppers; but for each plant left unpicked, Peter Piper must prune both its neighboring plants to provide the unpicked plant space to prosper. Peter Piper prefers to pick as parsimoniously[1] as possible because he's picking prematurely. Please propose an efficient algorithm to help Peter Piper pick which plants to prune.

Problem 8-2. [15 points] **Coin Crafting**

Ceal Naffrey is a thief in desperate need of money. He recently acquired n identical gold coins. Each coin has distinctive markings that would easily identify them as stolen if sold. However, using his amateur craftsman skills, Cael can melt down gold coins to craft other golden objects. Ceal has a buyer willing to purchase golden objects at different rates, but will only purchase one of any object. Ceal has compiled a list of the n golden objects, listing both the positive integer **purchase price** the buyer would be willing to pay for each object and each object's positive integer **melting number**: the number of gold coins that would need to be melted to craft that object. Given this list, describe an efficient algorithm to determine the maximum revenue that Ceal could make, by melting down his coins to craft into golden objects to sell to his buyer.

[1] He wants to minimize the sum of p_i for all picked peppers, subject to his pruning requirement.

Problem 8-3. [15 points] **Building Blocks**

Saggie Mimpson is a toddler who likes to build block towers. Each of her blocks is a 3D rectangular prism, where each block b_i has a positive integer width w_i, height h_i, and length ℓ_i, and she has at least three of each type of block. Each block may be **oriented** so that any opposite pair of its rectangular faces may serve as its **top** and **bottom** faces, and the **height** of the block in that orientation is the distance between those faces. Saggie wants to construct a tower by stacking her blocks as high as possible, but she can only stack an oriented block b_i on top of another oriented block b_j if the dimensions of the bottom of block b_i are strictly smaller[2] than the dimensions of the top of block b_j. Given the dimensions of each of her n blocks, describe an $O(n^2)$-time algorithm to determine the height of the tallest tower Saggie can build from her blocks.

Problem 8-4. [15 points] **Diffing Data**

Operating system Menix has a `diff` utility that can compare files. A **file** is an ordered sequence of strings, where the i^{th} string is called the i^{th} **line** of the file. A single **change** to a file is either:

- the insertion of a single new line into the file;
- the removal of a single line from the file; or
- swapping two adjacent lines in the file.

In Menix, swapping two lines is cheap, as they are already in the file, but inserting or deleting a line is expensive. A **diff** from a file A to a file B is any sequence of changes that, when applied in sequence to A will transform it into B, under the conditions that any line may be swapped at most once and any pair of swapped lines appear adjacent in A and adjacent in B. Given two files A and B, each containing exactly n lines, describe an $O(kn + n^2)$-time algorithm to return a diff from A to B that minimizes the number of changes that are **not swaps**, assuming that any line from either file is at most k ASCII characters long.

[2] If the bottom of block b_i has dimensions $p \times q$ and the top of block b_j has dimensions $s \times t$, then b_i can be stacked on b_j in this orientation if either: $p < s$ and $q < t$; or $p < t$ and $q < s$.

Problem 8-5. [40 points] **Princess Plum**

Princess Plum is a video game character collecting mushrooms in a digital haunted forest. The forest is an $n \times n$ square grid where each grid square contains either a tree, mushroom, or is empty. Princess Plum can move from one grid square to another if the two squares share an edge, but she cannot enter a grid square containing a tree. Princess Plum starts in the upper left grid square and wants to reach her home in the bottom right grid square[3]. The haunted forest is scary, so she wants to reach home via a **quick path**: a route from start to home that goes through at most $2n - 1$ grid squares (including start and home). If Princess Plum enters a square with a mushroom, she will pick it up. Let k be the maximum number of mushrooms she could pick up along any quick path, and let a quick path be **optimal** if she could pick up k mushrooms along that path.

(a) [15 points] Given a map of the forest grid, describe an $O(n^2)$-time algorithm to return the number of distinct optimal quick paths through the forest, assuming that some quick path exists.

(b) [25 points] Write a Python function `count_paths(F)` that implements your algorithm from (a). You can download a code template containing some test cases from the website. Submit your code online at `alg.mit.edu`.

```
def count_paths(F):
    '''
    Input:  F | size-n direct access array of size-n direct access arrays
              | each F[i][j] is either 't', 'm', or 'x'
              | for tree, mushroom, empty respectively
    Output: p | the number of distinct optimal paths in F
              | starting from (0,0) and ending at (n-1,n-1)
    '''
    p = 0
    ##################
    # YOUR CODE HERE #
    ##################
    return p
```

[3] Assume that both the start and home grid squares are empty.