Problem Session 8

Problem 8-1. Sunny Studies

Tim the Beaver needs to study for exams, but it's getting warmer, and Tim wants to spend more time outside. Tim enjoys being outside more when the weather is warmer: specifically, if the temperature outside is t integer units above zero, Tim's happiness will increase by t after spending the day outside (with a decrease in happiness when t is negative). On each of the n days until finals, Tim will either study or play outside (never both on the same day). In order to stay on top of coursework, Tim resolves never to play outside more than two days in a row. Given a weather forecast estimating temperature for the next n days, describe an $O(n)$-time dynamic programming algorithm to determine which days Tim should study in order to increase happiness the most.

Problem 8-2. 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.

Problem 8-3. 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[1] 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.

[1] 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 8-4. 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[2]. 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`.

```python
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
```

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