Note: Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. They are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 Planting Trees

This problem will guide you through the process of writing a dynamic programming algorithm.

You have a garden and want to plant some apple trees in your garden, so that they produce as many apples as possible. There are $n$ adjacent spots numbered 1 to $n$ in your garden where you can place a tree. Based on the quality of the soil in each spot, you know that if you plant a tree in the $i$th spot, it will produce exactly $x_i$ apples. However, each tree needs space to grow, so if you place a tree in the $i$th spot, you can’t place a tree in spots $i - 1$ or $i + 1$. What is the maximum number of apples you can produce in your garden?

(a) Give an example of an input for which:

- Starting from either the first or second spot and then picking every other spot (e.g. either planting the trees in spots 1, 3, 5... or in spots 2, 4, 6...) does not produce an optimal solution.
- The following algorithm does not produce an optimal solution: While it is possible to plant another tree, plant a tree in the spot where we are allowed to plant a tree with the largest $x_i$ value.

(b) To solve this problem, we’ll think about solving the following, more general problem: “What is the maximum number of apples that can be produced using only spots 1 to $i$?” Let $f(i)$ denote the answer to this question for any $i$. Define $f(0) = 0$, as when we have no spots, we can’t plant any trees. What is $f(1)$? What is $f(2)$?

(c) Suppose you know that the best way to plant trees using only spots 1 to $i$ does not place a tree in spot $i$. In this case, express $f(i)$ in terms of $x_i$ and $f(j)$ for $j < i$. (Hint: What spots are we left with? What is the best way to plant trees in these spots?)

(d) Suppose you know that the best way to plant trees using only spots 1 to $i$ places a tree in spot $i$. In this case, express $f(i)$ in terms of $x_i$ and $f(j)$ for $j < i$.

(e) Describe a linear-time algorithm to compute the maximum number of apples you can produce. (Hint: Compute $f(i)$ for every $i$. You should be able to combine your results from the previous two parts to perform each computation in $O(1)$ time).
2 Non-Prefix Code

As we have learned in lecture, the Huffman code satisfies the Prefix Property, which states that the bit string representing each symbol is not a prefix of the bit string representing any other symbol. One nice property of such codes is that, given a bit string, there is at most one way to decode it back to a sequence of symbols. However, this is not true anymore once we are working with codes that do not satisfy the Prefix Property. For example, consider the code that maps A to 1, B to 01 and C to 101. A bit string 101 can be interpreted in two ways: as C or as AB.

Your task is to, given a bit string $s$, determine how many ways one can interpret $s$. The mapping from symbols to bit strings of the code will be given to you as a dictionary $d$ (e.g., in the example, $d = \{ A : 1, B : 01, C : 101 \}$); you may assume that you can access each symbol in the dictionary in constant time. Your algorithm should run in time at most $O(n m \ell)$ where $n$ is the length of the input bit string $s$, $m$ is the number of symbols, and $\ell$ is an upper bound on the length of the bit strings representing symbols.

Please give a 3-part solution.

3 Equivalent Strings

We are given two strings $A, B$ of length $n, m$ respectively. These two strings can contain English characters a to z, as well the special character $?$. We say $A$ and $B$ are equivalent if it is possible to replace every instance of $?$ with a (possibly empty) string of English characters, such that the resulting strings (containing only English letters) are the exact same.

For example, “ab?” is equivalent to “a?cd”, since with the above replacements we can transform both strings into “abcd”. Similarly, “a?bc” is equivalent to “abc”, since we are allowed to replace $?$ with the empty string.

Give an efficient dynamic programming algorithm to determine if two strings are equivalent. Give a three-part solution.