CS 341: Algorithms @ Waterloo **Last Updated:** July 5, 2021
Homework 4 **Due Date: July 19, 2021, 11:00pm**

You are allowed to discuss with others but are not allowed to use any references other than the course notes and the three reference books. Please list your collaborators for each question. You must write your own solutions. See the course outline for the homework policy.

There are totally 52 marks. The full mark is 50. This homework is counted 10% of the course grade.

---

1. **Programming Problem: Keyboard Edit Distance** (20 marks)

   In this problem, we are given two strings A and B, and we want to write a program to measure the "keyboard edit distance" between them. Specifically, starting from string A, we want to calculate the minimum cost of obtaining string B through a series of insertions, deletions, and single-character substitutions, where each operation has a prescribed cost. (Strictly speaking, it is not a "distance", because starting from string A and starting from string B could result in different costs.)

   A and B can only contain uppercase English letters, commas (`,`), periods (`.`), and spaces. The cost of inserting any character is I. The cost of deleting any character is D. The cost of changing one character to another depends on the distance between the two characters on the keyboard – the closer the two characters are, the more likely it is to mistype one as the other.

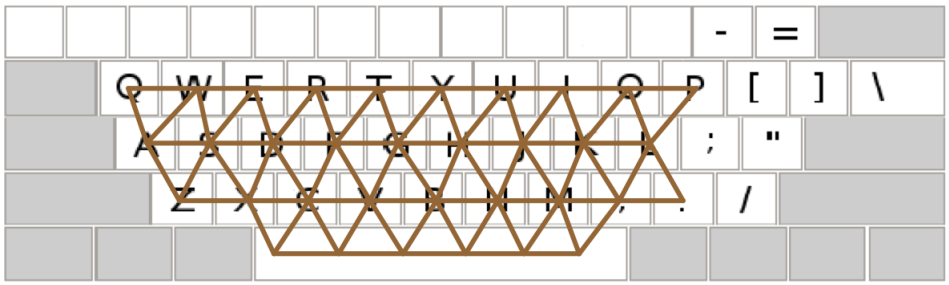

Figure 1: The QWERTY keyboard and the overlaying triangular grid. Note that some of the characters will not appear in the input strings, so they are not covered by the grid.

   Refer to Figure 1. For this problem, we use the QWERTY keybaord layout. We draw a triangular grid over it, where each vertex corresponds to a character (except the space which spans six vertices). The cost of changing one character to another is then given by the shortest distance between them on the grid. For example, the cost of changing from `'A'` to `'S'` is 1, the cost of changing from `'Z'` to `'P'` is 9, and the cost of changing from `' '` to `'.'` is 2.

   Find the minimum cost to obtain string B from string A through a series of insertions, deletions, and single-character substitutions.

   Input: The first line of input contains four integers: $|A|$ (the length of A), $|B|$, I, and D. The second line of input contains the string A. The third and final line of input contains the string B.

   A and B can only contain uppercase English letters, commas (`,`), periods (`.`), and spaces.

   It is guaranteed that $1 \leq |A|, |B| \leq 2000$ and $1 \leq I, D \leq 1000$.

Output: Output one single integer, the minimum cost of obtaining string B from string A.

Sample Input 1:

```
5 6 1000 1
ALICE
SPICES
```

Sample Output 1:

```
1002
```

Explanation: `ALICE` $\rightarrow$ `SLICE` $\rightarrow$ `SPICE` $\rightarrow$ `SPICES`. The cost is $1 + 1 + 1000 = 1002$.

Sample Input 2:

```
12 12 4 3
ASASASASASAS
SASASASASASA
```

Sample Output 2:

```
7
```

Explanation: `ASASASASASAS` $\rightarrow$ `SASASASASAS` $\rightarrow$ `SASASASASASA`. The cost is $3 + 4 = 7$.

2. **Written Problem: Sketching Data with Outliers** (12 marks)

We now examine how allowing outliers changes the data sketching problem from HW3. Recall that our input in this problem is a list of data points (x_i, y_i) with increasing x_i and an error tolerance ϵ. In the original problem, we were required to partition the x-coordinates into intervals from s_j to f_j and to compute a value h_j for each interval so that

$$\max_{i:s_j \le i \le f_j} |h_j - y_i| \le \epsilon.$$

In the variant we now consider, we can proclaim some subset of the data points to be outliers. We then remove the outliers from the data set, and solve the original problem on the remaining points. If $A \subset \{1, \ldots, n\}$ is the set of outliers, this is equivalent to requiring that

$$\max_{i:s_j \le i \le f_j \text{ and } i \notin A} |h_j - y_i| \le \epsilon. \tag{1}$$

The problem would be too easy if we were allowed to proclaim every point an outlier. Rather, we are told that we can label at most k points outliers. Our goal is to find a set A of size at most k and a partition of the input into intervals that satisfy (1) while using as few intervals as possible.

Design the fastest algorithm that you can to solve this problem. Prove the correctness and analyze the time complexity.

(Hint: You may begin by considering the following simpler variant of this problem: how can we find the smallest set A and value h so that

$$\max_{i:1 \le i \le n, i \notin A} |h - y_i| \le \epsilon.$$

Once you can solve this, you should be able to solve the whole problem by dynamic programming.)

3. **Written Problem: Covering Set** (10 marks)

Given an undirected graph $G = (V, E)$, a subset $S \subseteq V$ is a covering set if for every vertex $v \in V - S$ there exists $u \in S$ such that $uv \in E$ (i.e. every vertex $v \in V - S$ has a neighbor in S). We are interested in finding a covering set of minimum total weight, but this problem is NP-hard in general graphs. Show that this problem is easy in trees.

Input: A tree $T = (V, E)$ in the adjacency list representation, a weight w_v for each vertex $v \in V$.

Output: A covering set $S \subseteq V$ that minimizes the total weight $\sum_{v \in S} w_v$.

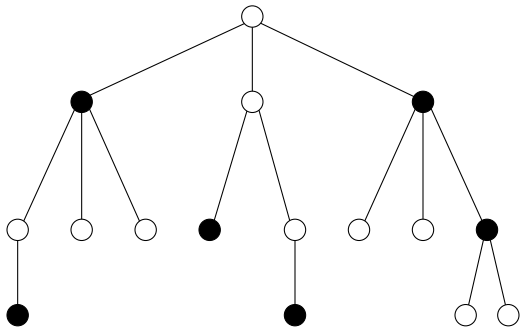

Figure 2: The black vertices form a minimum covering set assuming all weights are one.

Design the fastest algorithm that you can to solve this problem. Prove the correctness and analyze the time complexity.

4. **Written Problem: Running to Meetings** (10 marks)

A professor has many meetings to go to every day. In fact, it is impossible to get to all of them. We will figure out how to get our professor to as many meetings as possible.

Our problem has two inputs: a list of meetings with their locations and a map of campus. We will model the campus as a weighted graph $G = (V, E, l)$, where V is the set of possible locations of meetings and l gives the distance between vertices. We will assume that the time required to get from vertex a to vertex b equals the distance between them, $l(a, b)$. Meeting i is specified by a vertex, a_i, the time that the meeting starts, s_i, and the time that meeting ends, f_i.

Design an algorithm that takes as input the graph G and the list of meetings, and returns a subset of meetings for the professor to attend. The professor must be able to get from one meeting to the next. That is, if the professor goes from meeting i to meeting j, it must be the case that $f_i + l(a_i, a_j) \leq s_j$.

You should assume that the professor's first meeting is with a coffee shop at vertex a_0, that the professor can always get to this meeting, and that it ends at time f_0. Design the fastest algorithm that you can that gets the professor to as many meetings as possible. Prove the correctness and analyze the time complexity.