

You have a keyboard layout as shown above in the XY plane, where each English uppercase letter is located at some coordinate, for example, the letter **A** is located at coordinate **(0,0)**, the letter **B** is located at coordinate **(0,1)**, the letter **P** is located at coordinate **(2,3)** and the letter **Z** is located at coordinate **(4,1)**.

Given the string word, return the minimum total distance to type such string using only two fingers. The distance between coordinates **(x1,y1)** and **(x2,y2)** is **|x1 - x2| + |y1 - y2|**.

Note that the initial positions of your two fingers are considered free so don't count towards your total distance, also your two fingers do not have to start at the first letter or the first two letters.

**Example 1:**

**Input:** word = "CAKE"

**Output:** 3

**Explanation:**

Using two fingers, one optimal way to type "CAKE" is:

Finger 1 on letter 'C' -> cost = 0

Finger 1 on letter 'A' -> cost = Distance from letter 'C' to letter 'A' = 2

Finger 2 on letter 'K' -> cost = 0

Finger 2 on letter 'E' -> cost = Distance from letter 'K' to letter 'E' = 1

Total distance = 3

**Example 2:**

**Input:** word = "HAPPY"

**Output:** 6

**Explanation:**

Using two fingers, one optimal way to type "HAPPY" is:

Finger 1 on letter 'H' -> cost = 0

Finger 1 on letter 'A' -> cost = Distance from letter 'H' to letter 'A' = 2

Finger 2 on letter 'P' -> cost = 0

Finger 2 on letter 'P' -> cost = Distance from letter 'P' to letter 'P' = 0

Finger 1 on letter 'Y' -> cost = Distance from letter 'A' to letter 'Y' = 4

Total distance = 6

**Example 3:**

**Input:** word = "NEW"

**Output:** 3

**Example 4:**

**Input:** word = "YEAR"

**Output:** 7

**Constraints:**

* 2 <= word.length <= 300
* Each word[i] is an English uppercase letter.