SP 2023 DP

Question 1

Problem statement

Once upon a time, in a mystical land known as Linguaville, there lived two wise scribes named Orin and Elara. Orin and Elara were tasked with safeguarding ancient texts, each containing a unique and magical incantation. These incantations were the key to preserving the harmony of Linguaville, and the scribes took their duty very seriously.

The prosperity of Linguaville was intrinsically linked to the harmony of its inhabitants. The more harmonious the incantations, the more abundant the crops, the more joyful the celebrations, and the greater the peace throughout the land. Orin and Elara were not just scribes; they were stewards of the land's well-being.

One fateful day, a sinister sorcerer cast a spell of discord over Linguaville. As a result, the more dissonance there was among the incantations, the more the land suffered. The skies darkened, the rivers ran dry, and crops withered. The once-vibrant town square became a gloomy and desolate place.

Orin and Elara knew that the only way to reverse the spell was to reunite the incantations, and they needed to do it in the minimum number of steps. The longer the process took, the more the land would suffer.

With each passing day, Linguaville's despair deepened. The scribes understood the urgency of their task. "What is the minimum number of steps we need to take to make these scrolls identical and restore harmony to Linguaville as swiftly as possible?" they pondered.

Your task, as an apprentice scribe, is to help Orin and Elara determine the minimum number of steps required to make the two scrolls, represented as strings word1 and word2, identical. In one step, you can delete exactly one character from either scroll.

The fate of Linguaville rests on your shoulders. Can you find the quickest path to reunite the incantations and restore harmony to the land before it's too late?

Class Definition

Class: `Question1`

Method: `minSteps`

The class `Question1` includes the method `minSteps` which accepts two `String` parameters and returns an `int`.

```
//java
public int minSteps(String word1, String word2)

//cpp
public int minSteps(string word1, string word2)

#python
def minSteps(word1: str, word2: str) -> int
```

Constraints

- `word1` and `word2` length can be up to 500 characters, for each string.
- `word1` and `word2` consist of only lowercase English letters.

Examples

Example 0

```
Question1 q1 = new Question1();
System.out.println(q1.minSteps("sea","eat")); // Returns: 2
```

We can delete 1 character in each string, to make both becomes `ea`.

Example 1

```
Question1 q1 = new Question1();
System.out.println(q1.minSquareGems("leetcode","etco")); // Returns: 4
```

Deletes 4 characters in first string, to becomes `etc`

Test Cases

Input	Expected Output
“sea”,"eat”	2
“leetcode”,"etco”	4
“pp”,"itm”	5
“nw”,"for”	5
“bbs”,"vjkc”	7
“ueg”,"samq”	
“dqdzw”,"fzcmop”	
“witim”,"gkyvfy”	
“balhqtzfka”,"rxfdithj”	
“apsuummteb”,"nodfftho”	
“luisgmhiihskdpj”,"fdthfuczbanhqswfpw”	
“jzvjinkpvlzfjzk”,"jpxsbpupirervvqktm”	
“zvrywlucuzvbsaezeaxa”,“cwzpnxqiregjztcznereui”	
“chroblesmqczebeuqixm”,"rnzdqrxnbshuauxixkxwsg”	
“xsugyizivpfyywnldyejjcvpv”,"tqhkdhxbeobrdmzwlyltuyxau”	
“rcdzsvfldsyqjbqmtoshnsxtv”,"vzmcicdcyoymecjwwcvkvbihm”	

Please note that the expected outputs for some test cases are not provided.

Question 2

Problem Statement

In the bustling city of Numerville, there was a renowned adventurer named Alaric. Alaric was famous for his incredible journeys into the treacherous Numeral Mountains, a mysterious and ancient range filled with hidden secrets and dangers. Alaric's expeditions were unlike any other; he was on a quest to find the most valuable treasures buried deep within the mountains, treasures known as "The Numeral Stones."

The Numeral Stones were special crystals, each representing a number from a unique set of integers. These stones were scattered throughout the mountains, and legends said that when combined in a specific order, they held unimaginable power and knowledge.

Alaric had discovered an old map that hinted at the location of these stones, but the map also revealed an intricate challenge. To access the stones, he needed to navigate a series of m magical gates, each requiring a unique numeric code to unlock. These gates were guarded by ancient spirits that challenged him with mathematical puzzles.

The first gate was known as the "Gate of Multiplication." It demanded that Alaric, who was now accompanied by a group of skilled companions, solve a complex equation to prove his worthiness.

Alaric's team had a collection of precious Numeral Stones with values represented by the array "nums." They also had a set of multipliers, each corresponding to a specific operation they could perform. These multipliers were like enchanted scrolls, and using them, Alaric could add or subtract values from the Numeral Stones to unlock the gate. However, there was a catch: Alaric had to be strategic.

With each operation, Alaric could choose one stone from either the beginning or the end of their collection, apply the corresponding multiplier, and then remove the stone. The goal was to maximize the total value of the stones by the time they reached the Gate of Multiplication.

Alaric knew that he needed to be exceptionally clever and make the best choices to pass through the gate. His companions looked to him for guidance, and the fate of their expedition and the quest for the Numeral Stones depended on his decisions.

Your task, as a fellow adventurer, is to help Alaric and his companions determine the maximum score they can achieve after performing exactly m operations. The journey through the Numeral Mountains is perilous, and every decision counts. Can you guide them to success and unlock the secrets of the Numeral Stones?

Class Definition

Class: `Question2`

Method: `maxScore`

The class `Question1` includes the method `maxScore` which accepts two `int` array parameters and returns an `int`.

```
//java
public int maxScore(int[] nums, int[] mul)

//cpp
public int maxScore(vector<int> nums, vector<int> mul)

#python
def maxScore(nums: List[int], mul: List[int]) -> int
```

Constraints

- 1 ≤ `mul.length` ≤ 300
- `mul.length` ≤ `nums.length` ≤ 100000 (10^5)
- -1000 ≤ `nums[i]`, `mul[i]` ≤ 1000

Examples

Example 0

```
Question2 q2 = new Question2();
System.out.println(q2.maxScore([1,2,3], [3,2,1])); // Returns: 14
```

An optimal solution is as follows:

1. Choose from the end, [1,2,3], adding 3 * 3 = 9 to the score.
2. Choose from the end, [1,2], adding 2 * 2 = 4 to the score.
3. Choose from the end, [1], adding 1 * 1 = 1 to the score.

The total score is 9 + 4 + 1 = 14.

Example 1

```
Question2 q2 = new Question2();
System.out.println(q2.maxScore(1)); // Returns: 0
```

An optimal solution is as follows:

1. Choose from the start, [-5,-3,-3,-2,7,1], adding -5 * -10 = 50 to the score.
2. Choose from the start, [-3,-3,-2,7,1], adding -3 * -5 = 15 to the score.
3. Choose from the start, [-3,-2,7,1], adding -3 * 3 = -9 to the score.
4. Choose from the end, [-2,7,1], adding 1 * 4 = 4 to the score.
5. Choose from the end, [-2,7], adding 7 * 6 = 42 to the score.

The total score is 50 + 15 - 9 + 4 + 42 = 102.

Hint: Pretty similar to Alice/Bob candy (or stone) picking game we have discussed during workshop. This Alice Bob candy picking game was also mentioned in Announcements.

Test Cases

Input	Expected Output
[1, 2, 3] [3, 2, 1]	14
[-5, -3, -3, -2, 7, 1] [-10, -5, 3, 4, 6]	102
[-688, -977, -729] [-160, -312, -983]	1297919
[985, -682, 839, -81, -206, -905, 668, 577, 999, 929, -763, -852, 631, 118, 990, 1, -251, -354, 57, -491, -725, -786, 514, -968, 142] [-771, -131, 290, 796, 906, -751, -355, 889, -981, 692, 642, -543, -270, -372, -778]	3944466
[940, 64, 733, 177, -181, -397, -728, -829, -499, 984] [-623, -699, 666, -295]	
[-14, 487, 193, 950, 817, 281, 795, -799, 726, 65, 817, -127, -498, 231, 620, -667, 414] [-410, 768, 277, -515, 119, 92, 331, -598, -233, 895, 743, -726, 98, -166, -770, 651, -532]	
[651, 610] [941, 479]	
[18, 778, -544, 224, 153, 755, -743, -97, 837, -699, 728, 110, -84, 130, -2, -448, 755, 866, 200, 996, -124] [-180, 645, 953, 260, -627, 133, -77, -374, -597, 912, -296, -983, -712, 430, 535, 562, -687]	

Input	Expected Output
[-327, -895, 578, 448, -681, 41, -912, 101, 794, -308, -514, -777] [245, -527, 55, 337]	
[667, 407] [562]	
[-433, 259, 321, 421, 845, -422, -531, -851, -925, 797, 139, -750, -981, -690, 670, -167, -165, -571, 611, -321, -537, -431, -65, 601] [-460, -396, -730, 801, 805, 497, -754, -354, -177, 279, -612, -718, 255, -189]	
[-354, -630, -876, 798, -344, 301, -164, 132, -917, 248, -646, -412, 331, -456, -975, 893, 435, -733, 562, 670, 677, -204, -935, 845, 625] [-300, 936, -57, -419, 432, 506, -458, -341, -218, 165, -367, -71, -841, 737, -47, 407, 41]	
[261] [-245]	
[156, -276, 637, 123, -319, -18, 653, 104, -766, -993] [356, 417, 405, -252, 867, -96, 507]	
[371, -632, -573, -749, 742, 517, -285, -991, 133, 914, 474, -925, 491, 753, -343, 378, 431, -566, 734, -952, 139] [-311, -653, 700, 422, 528, 685, 460, 6, -458, 0, 992]	
[-728, -701, 913, 105, 84, -507, -880, -434] [-460, -605, -942]	
[-717] [688]	
[-293, -853, -600, 859, 959, -911, 606, -738, -310, -234] [672, 539, 928, -635]	

Please note that the expected outputs for some test cases are not provided.