香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

G262 - Removing Letters

Casper Wong {gasbug}
2026-02-14

Background

Problem idea by gasbug

Preparation by gasbug

Problem Restatement

- Given 3 strings of uppercase letters
- In each operation, you can remove all occurrences of any uppercase letter X from any 2 of the 3 strings
- Find the minimum number of operations needed to make the 3 strings equal **after sorting the characters within each string alphabetically**

Subtasks

Task	Attempts	Max	Mean	Std Dev	Subtasks				
G262 - Removing Letters	131	100	23.877	37.681	11: 59	22: 35	14: 29	21: 27	32: 23

First solved by **ywgs217** (Cheung Suet Nam) at **20m 0s**

Subtask Constraints

Subtask	Points	Constraints
1	11	$S_2 = S_3$ $\lvert S_1 \rvert = \lvert S_2 \rvert = \lvert S_3 \rvert = 1$
2	22	$S_2 = S_3$ $\lvert S_1 \rvert = 2$ $\lvert S_2 \rvert = \lvert S_3 \rvert = 1$
3	14	$S_2 = S_3$ Each letter appears in a string at most once
4	21	Each letter appears in a string at most once
5	32	No additional constraints

Subtask 1 (11%)

- $S_2 = S_3$, all 3 strings only consist of one letter, what are the possible cases?
- Case 1
  - $S_1 = S_2$
  - E.g. $S_1 = S_2 = S_3$ = “A”
  - How many operations do we need?
- Case 2
  - $S_1 \neq S_2$
  - E.g. S_1 = “A”, $S_2 = S_3$ = “B”
  - How many operations do we need?

Subtask 1 (11%)

- $S_2 = S_3$, all 3 strings only consist of one letter, what are the possible cases?
- Case 1
  - $S_1 = S_2$
  - E.g. $S_1 = S_2 = S_3$ = “A”
  - **No operations needed**
- Case 2
  - $S_1 \neq S_2$
  - E.g. S_1 = “A”, $S_2 = S_3$ = “B”
  - **2 operations needed**

Subtask 2 (22%)

- $S_2 = S_3$, $|S_2| = |S_3| = 1$, $|S_1| = 2$
- Lets list all the cases again!
- Case 1
  - S_1 does not contain S_2
  - Case 1a
    - Both letter in S_1 are the same
    - E.g. S_1 = “AA”, $S_2 = S_3$ = ”B”
    - How many operations do we need?
  - Case 1b
    - The letters in S_1 are different
    - E.g. S_1 = “AB”, $S_2 = S_3$ = “C”
    - How many operations do we need?

Subtask 2 (22%)

- $S_2 = S_3$, $|S_2| = |S_3| = 1$, $|S_1| = 2$
- Lets list all the cases again!
- Case 1
  - S_1 does not contain S_2
  - Case 1a
    - Both letter in S_1 are the same
    - E.g. S_1 = “AA”, $S_2 = S_3$ = ”B”
    - **2 operations needed**
  - Case 1b
    - The letters in S_1 are different
    - E.g. S_1 = “AB”, $S_2 = S_3$ = “C”
    - **3 operations needed**

Subtask 2 (22%)

- $S_2 = S_3$, $|S_2| = |S_3| = 1$, $|S_1| = 2$
- Lets list all the cases again!
- Case 2
  - S_1 contains S_2 once
  - E.g. S_1 = “AB”, $S_2 = S_3$ = “A”
  - How many operations do we need?
- Case 3
  - S_1 contains S_2 twice
  - E.g. S_1 = “AA”, $S_2 = S_3$ = “A”
  - How many operations do we need?

Subtask 2 (22%)

- $S_2 = S_3$, $|S_2| = |S_3| = 1$, $|S_1| = 2$
- Lets list all the cases again!
- Case 2
  - S_1 contains S_2 once
  - E.g. S_1 = “AB”, $S_2 = S_3$ = “A”
  - **1 operation needed**
- Case 3
  - S_1 contains S_2 twice
  - E.g. S_1 = “AA”, $S_2 = S_3$ = “A”
  - **2 operations needed**

Observation: Letters are independent

- Each operation only affects one letter
- The occurrences of other letters remain unchanged
- E.g. S_1 = “ABB”, S_2 = “AABB”, S_3 = “A” and we remove ‘A’ all 3 strings with 2 operations
- S_1 = “BB”, S_2 = “BB”, S_3 = “”
- The occurrences of ‘B’ in all strings remain unchanged
- We can handle letter by letter!

Subtask 3 (14%)

- $S_2 = S_3$, each letter appears in a string at most once
- For each letter C:
  - If C exists in all 3 strings:
    - How many operations do we need?
  - If C only exist in S_2 and S_3:
    - How many operations do we need?
  - If C only exist in S_1:
    - How many operations do we need?
  - If C doesn't exist in any of the 3 strings:
    - How many operations do we need?

Subtask 3 (14%)

- $S_2 = S_3$, each letter appears in a string at most once
- For each letter C:
  - If C exists in all 3 strings:
    - **No operations needed**
  - If C only exist in S_2 and S_3:
    - **1 operation needed**
  - If C only exist in S_1:
    - **1 operation needed**
  - If C doesn't exist in any of the 3 strings:
    - **No operations needed**

Subtask 4 (21%)

- S_2 may not equal S_3
- For each letter C:
  - If C exists in all 3 strings:
    - How many operations do we need?
  - If C exists in 2 strings:
    - How many operations do we need?
  - If C exists in 1 string:
    - How many operations do we need?
  - If C doesn't exist in any of the 3 strings:
    - How many operations do we need?

Subtask 4 (21%)

- S_2 may not equal S_3
- For each letter C:
  - If C exists in all 3 strings:
    - **No operations needed**
  - If C exists in 2 strings:
    - **1 operation needed**
  - If C exists in 1 string:
    - **1 operation needed**
  - If C doesn't exist in any of the 3 strings:
    - **No operations needed**

Subtask 5 (21%)

- Each letter may appear any number of times
- Since each operation remove **all** occurrences of the letter from 2 strings
- The exactly occurrences doesn't matter, we only need to know if the occurrences are same across the 3 strings
- For each letter C:
  - If the occurrence of C is equal in all 3 string:
    - How many operations do we need?
  - Else:
    - How many operations do we need?

Subtask 5 (21%)

- Each letter may appear any number of times
- Since each operation remove **all** occurrences of the letter from 2 strings
- The exactly occurrences doesn't matter, we only need to know if the occurrences are same across the 3 strings
- For each letter C:
  - If the occurrence of C is equal in all 3 string:
    - **No operations needed**
  - Else:
    - **2 operations** to remove C in all 3 strings

Subtask 5 (21%)

- Each letter may appear any number of times
- For each letter C:
  - If the occurrence of C is equal in all 3 string:
    - **No operations needed**
  - Else:
    - **2 operations** to remove C in all 3 strings
    - Wait... Is this optimal?
    - If C doesn't exist in at least 1 string, we only need **1 operation**!

Subtask 5 (21%)

- To check number of occurrences of C in S
- We can either create a function
- Or simply use s.count(c) in python

```
def num_of_occurrences(c, s):
    ans = 0
    for i in s:
        if i == c:
            ans = ans + 1
    return ans
```

Tips

- You have to be very careful in order to solve case handling tasks
- You may want to ask yourself
  - Is that all the cases? ← e.g. separate case 1 into 1a and 1b in subtask 2
  - Is the handling of each case correct? ← e.g. subtask 5