# Problem 712: Minimum ASCII Delete Sum for Two Strings

## Problem Information

**Difficulty:** Medium
**Acceptance Rate:** 0.00%
**Paid Only:** No

## Problem Description

Given two strings

s1

and

s2

, return

the lowest

ASCII

sum of deleted characters to make two strings equal

.

Example 1:

Input:

s1 = "sea", s2 = "eat"

Output:

231

Explanation:

Deleting "s" from "sea" adds the ASCII value of "s" (115) to the sum. Deleting "t" from "eat" adds 116 to the sum. At the end, both strings are equal, and 115 + 116 = 231 is the minimum sum possible to achieve this.

Example 2:

Input:

s1 = "delete", s2 = "leet"

Output:

403

Explanation:

Deleting "dee" from "delete" to turn the string into "let", adds 100[d] + 101[e] + 101[e] to the sum. Deleting "e" from "leet" adds 101[e] to the sum. At the end, both strings are equal to "let", and the answer is 100+101+101+101 = 403. If instead we turned both strings into "lee" or "eet", we would get answers of 433 or 417, which are higher.

Constraints:

1 <= s1.length, s2.length <= 1000

s1

and

s2

consist of lowercase English letters.

## Code Snippets

### C++:

```
class Solution {
public:
int minimumDeleteSum(string s1, string s2) {

}
};
```

### Java:

```
class Solution {
public int minimumDeleteSum(String s1, String s2) {

}
}
```

### Python3:

```
class Solution:
def minimumDeleteSum(self, s1: str, s2: str) -> int:
```

### Python:

```
class Solution(object):
def minimumDeleteSum(self, s1, s2):
"""
:type s1: str
:type s2: str
:rtype: int
"""
```

### JavaScript:

```
/**
* @param {string} s1
* @param {string} s2
* @return {number}
*/
var minimumDeleteSum = function(s1, s2) {
```

```
};
```

**TypeScript:**

```
function minimumDeleteSum(s1: string, s2: string): number {

};
```

**C#:**

```
public class Solution {
public int MinimumDeleteSum(string s1, string s2) {

}
}
```

**C:**

```
int minimumDeleteSum(char* s1, char* s2) {

}
```

**Go:**

```
func minimumDeleteSum(s1 string, s2 string) int {

}
```

**Kotlin:**

```
class Solution {
fun minimumDeleteSum(s1: String, s2: String): Int {

}
}
```

**Swift:**

```
class Solution {
func minimumDeleteSum(_ s1: String, _ s2: String) -> Int {

}
```

```
}
```

**Rust:**

```
impl Solution {
pub fn minimum_delete_sum(s1: String, s2: String) -> i32 {

}
}
```

**Ruby:**

```
# @param {String} s1
# @param {String} s2
# @return {Integer}
def minimum_delete_sum(s1, s2)

end
```

**PHP:**

```
class Solution {

/**
* @param String $s1
* @param String $s2
* @return Integer
*/
function minimumDeleteSum($s1, $s2) {

}
}
```

**Dart:**

```
class Solution {
int minimumDeleteSum(String s1, String s2) {

}
}
```

**Scala:**

```
object Solution {
def minimumDeleteSum(s1: String, s2: String): Int = {

}
}
```

**Elixir:**

```
defmodule Solution do
@spec minimum_delete_sum(s1 :: String.t, s2 :: String.t) :: integer
def minimum_delete_sum(s1, s2) do

end
end
```

**Erlang:**

```
-spec minimum_delete_sum(S1 :: unicode:unicode_binary(), S2 ::
unicode:unicode_binary()) -> integer().
minimum_delete_sum(S1, S2) ->
.
```

**Racket:**

```
(define/contract (minimum-delete-sum s1 s2)
(-> string? string? exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Minimum ASCII Delete Sum for Two Strings
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/
```

```
class Solution {
public:
int minimumDeleteSum(string s1, string s2) {

}
};
```

### Java Solution:

```
/**
* Problem: Minimum ASCII Delete Sum for Two Strings
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int minimumDeleteSum(String s1, String s2) {

}
}
```

### Python3 Solution:

```
"""
Problem: Minimum ASCII Delete Sum for Two Strings
Difficulty: Medium
Tags: string, dp

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def minimumDeleteSum(self, s1: str, s2: str) -> int:
# TODO: Implement optimized solution
```

```
pass
```

### Python Solution:

```
class Solution(object):
def minimumDeleteSum(self, s1, s2):
"""
:type s1: str
:type s2: str
:rtype: int
"""
```

### JavaScript Solution:

```
/**
* Problem: Minimum ASCII Delete Sum for Two Strings
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {string} s1
* @param {string} s2
* @return {number}
*/
var minimumDeleteSum = function(s1, s2) {

};
```

### TypeScript Solution:

```
/**
* Problem: Minimum ASCII Delete Sum for Two Strings
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
```

```
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function minimumDeleteSum(s1: string, s2: string): number {

};
```

### C# Solution:

```
/*
* Problem: Minimum ASCII Delete Sum for Two Strings
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int MinimumDeleteSum(string s1, string s2) {

}
}
```

### C Solution:

```
/*
* Problem: Minimum ASCII Delete Sum for Two Strings
* Difficulty: Medium
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int minimumDeleteSum(char* s1, char* s2) {

}
```

### Go Solution:

```
// Problem: Minimum ASCII Delete Sum for Two Strings
// Difficulty: Medium
// Tags: string, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func minimumDeleteSum(s1 string, s2 string) int {

}
```

### Kotlin Solution:

```
class Solution {
fun minimumDeleteSum(s1: String, s2: String): Int {

}
}
```

### Swift Solution:

```
class Solution {
func minimumDeleteSum(_ s1: String, _ s2: String) -> Int {

}
}
```

### Rust Solution:

```
// Problem: Minimum ASCII Delete Sum for Two Strings
// Difficulty: Medium
// Tags: string, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn minimum_delete_sum(s1: String, s2: String) -> i32 {
```

```
}
}
```

### Ruby Solution:

```
# @param {String} s1
# @param {String} s2
# @return {Integer}
def minimum_delete_sum(s1, s2)

end
```

### PHP Solution:

```
class Solution {

/**
* @param String $s1
* @param String $s2
* @return Integer
*/
function minimumDeleteSum($s1, $s2) {

}
}
```

### Dart Solution:

```
class Solution {
int minimumDeleteSum(String s1, String s2) {

}
}
```

### Scala Solution:

```
object Solution {
def minimumDeleteSum(s1: String, s2: String): Int = {

}
```

```
}
```

### Elixir Solution:

```
defmodule Solution do
@spec minimum_delete_sum(s1 :: String.t, s2 :: String.t) :: integer
def minimum_delete_sum(s1, s2) do

end
end
```

### Erlang Solution:

```
-spec minimum_delete_sum(S1 :: unicode:unicode_binary(), S2 ::
unicode:unicode_binary()) -> integer().
minimum_delete_sum(S1, S2) ->
.
```

### Racket Solution:

```
(define/contract (minimum-delete-sum s1 s2)
(-> string? string? exact-integer?)
)
```