Problem 2268: Minimum Number of Keypresses

Problem Information

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

Problem Description

You have a keypad with

9

buttons, numbered from

1

to

9

, each mapped to lowercase English letters. You can choose which characters each button is matched to as long as:

All 26 lowercase English letters are mapped to.

Each character is mapped to by

exactly

1

button.

Each button maps to

at most

3

characters.

To type the first character matched to a button, you press the button once. To type the second character, you press the button twice, and so on.

Given a string

s

, return

the

minimum

number of keypresses needed to type

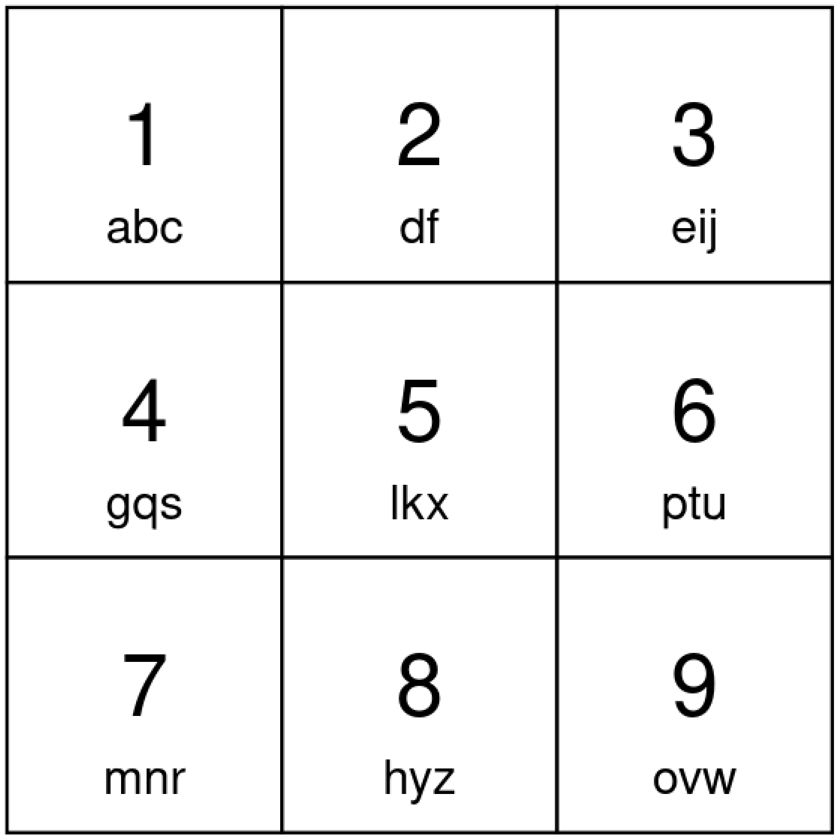
s

using your keypad.

Note

that the characters mapped to by each button, and the order they are mapped in cannot be changed.

Example 1:



Input:

s = "apple"

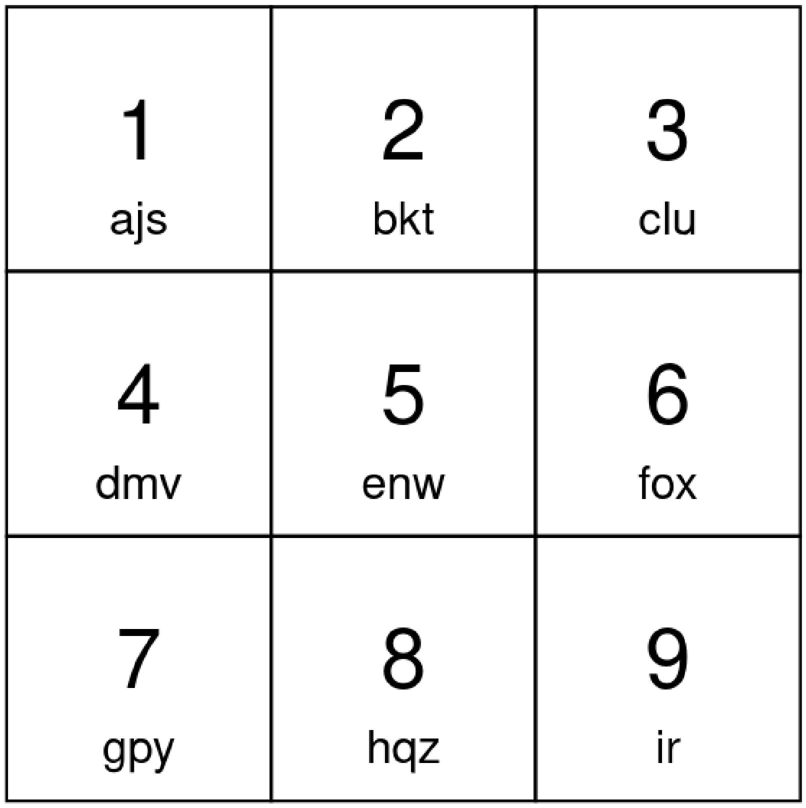
Output:

5

Explanation:

One optimal way to setup your keypad is shown above. Type 'a' by pressing button 1 once. Type 'p' by pressing button 6 once. Type 'p' by pressing button 6 once. Type 'l' by pressing button 5 once. Type 'e' by pressing button 3 once. A total of 5 button presses are needed, so return 5.

Example 2:



1 ajs	2 bkt	3 clu
4 dmv	5 enw	6 fox
7 gpy	8 hqz	9 ir

Input:

s = "abcdefghijkl"

Output:

15

Explanation:

One optimal way to setup your keypad is shown above. The letters 'a' to 'i' can each be typed by pressing a button once. Type 'j' by pressing button 1 twice. Type 'k' by pressing button 2 twice. Type 'l' by pressing button 3 twice. A total of 15 button presses are needed, so return 15.

Constraints:

1 <= s.length <= 10

5

s

consists of lowercase English letters.

Code Snippets

C++:

```
class Solution {
public:
int minimumKeypresses(string s) {

}
};
```

Java:

```
class Solution {
public int minimumKeypresses(String s) {

}
}
```

Python3:

```
class Solution:
def minimumKeypresses(self, s: str) -> int:
```

Python:

```
class Solution(object):
def minimumKeypresses(self, s):
"""
:type s: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {number}
*/
var minimumKeypresses = function(s) {

};
```

TypeScript:

```
function minimumKeypresses(s: string): number {

};
```

C#:

```
public class Solution {
public int MinimumKeypresses(string s) {

}
}
```

C:

```
int minimumKeypresses(char* s) {

}
```

Go:

```
func minimumKeypresses(s string) int {

}
```

Kotlin:

```
class Solution {
fun minimumKeypresses(s: String): Int {

}
}
```

Swift:

```
class Solution {
func minimumKeypresses(_ s: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn minimum_keypresses(s: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} s
# @return {Integer}
def minimum_keypresses(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return Integer
*/
function minimumKeypresses($s) {

}
}
```

Dart:

```
class Solution {
int minimumKeypresses(String s) {

}
}
```

Scala:

```
object Solution {
def minimumKeypresses(s: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec minimum_keypresses(s :: String.t) :: integer
def minimum_keypresses(s) do

end
end
```

Erlang:

```
-spec minimum_keypresses(S :: unicode:unicode_binary()) -> integer().
minimum_keypresses(S) ->
.
```

Racket:

```
(define/contract (minimum-keypresses s)
(-> string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Number of Keypresses
* Difficulty: Medium
* Tags: string, greedy, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
```

```
class Solution {
public:
int minimumKeypresses(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Number of Keypresses
* Difficulty: Medium
* Tags: string, greedy, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int minimumKeypresses(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Number of Keypresses
Difficulty: Medium
Tags: string, greedy, hash, sort

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

class Solution:
def minimumKeypresses(self, s: str) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

```
class Solution(object):
def minimumKeypresses(self, s):
"""
:type s: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Number of Keypresses
* Difficulty: Medium
* Tags: string, greedy, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

/**
* @param {string} s
* @return {number}
*/
var minimumKeypresses = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Number of Keypresses
* Difficulty: Medium
* Tags: string, greedy, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
```

```
*/

function minimumKeypresses(s: string): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Number of Keypresses
* Difficulty: Medium
* Tags: string, greedy, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int MinimumKeypresses(string s) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Number of Keypresses
* Difficulty: Medium
* Tags: string, greedy, hash, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int minimumKeypresses(char* s) {

}
```

Go Solution:

```
// Problem: Minimum Number of Keypresses
// Difficulty: Medium
// Tags: string, greedy, hash, sort
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func minimumKeypresses(s string) int {

}
```

Kotlin Solution:

```
class Solution {
fun minimumKeypresses(s: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func minimumKeypresses(_ s: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Number of Keypresses
// Difficulty: Medium
// Tags: string, greedy, hash, sort
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn minimum_keypresses(s: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {Integer}
def minimum_keypresses(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return Integer
*/
function minimumKeypresses($s) {

}
}
```

Dart Solution:

```
class Solution {
int minimumKeypresses(String s) {

}
}
```

Scala Solution:

```
object Solution {
def minimumKeypresses(s: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec minimum_keypresses(s :: String.t) :: integer
def minimum_keypresses(s) do

end
end
```

Erlang Solution:

```
-spec minimum_keypresses(S :: unicode:unicode_binary()) -> integer().
minimum_keypresses(S) ->
.
```

Racket Solution:

```
(define/contract (minimum-keypresses s)
(-> string? exact-integer?)
)
```