Problem 535: Encode and Decode TinyURL

Problem Information

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

Problem Description

Note: This is a companion problem to the

System Design

problem:

Design TinyURL

.

TinyURL is a URL shortening service where you enter a URL such as

https://leetcode.com/problems/design-tinyurl

and it returns a short URL such as

http://tinyurl.com/4e9iAk

. Design a class to encode a URL and decode a tiny URL.

There is no restriction on how your encode/decode algorithm should work. You just need to ensure that a URL can be encoded to a tiny URL and the tiny URL can be decoded to the original URL.

Implement the

Solution

class:

Solution()

Initializes the object of the system.

String encode(String longUrl)

Returns a tiny URL for the given

longUrl

.

String decode(String shortUrl)

Returns the original long URL for the given

shortUrl

. It is guaranteed that the given

shortUrl

was encoded by the same object.

Example 1:

Input:

url = "https://leetcode.com/problems/design-tinyurl"

Output:

"https://leetcode.com/problems/design-tinyurl"

Explanation:

Solution obj = new Solution(); string tiny = obj.encode(url); // returns the encoded tiny url. string ans = obj.decode(tiny); // returns the original url after decoding it.

Constraints:

1 <= url.length <= 10

4

url

is guranteed to be a valid URL.

Code Snippets

C++:

```
class Solution {
public:

// Encodes a URL to a shortened URL.
string encode(string longUrl) {

}

// Decodes a shortened URL to its original URL.
string decode(string shortUrl) {

}
};

// Your Solution object will be instantiated and called as such:
// Solution solution;
// solution.decode(solution.encode(url));
```

Java:

```
public class Codec {

// Encodes a URL to a shortened URL.
public String encode(String longUrl) {

}

// Decodes a shortened URL to its original URL.
public String decode(String shortUrl) {

}
}

// Your Codec object will be instantiated and called as such:
// Codec codec = new Codec();
// codec.decode(codec.encode(url));
```

Python3:

```
class Codec:

def encode(self, longUrl: str) -> str:
"""Encodes a URL to a shortened URL.
"""



def decode(self, shortUrl: str) -> str:
"""Decodes a shortened URL to its original URL.
"""



# Your Codec object will be instantiated and called as such:
# codec = Codec()
# codec.decode(codec.encode(url))
```

Python:

```
class Codec:

def encode(self, longUrl):
"""Encodes a URL to a shortened URL.

:type longUrl: str
:rtype: str
"""
```

```
def decode(self, shortUrl):
"""Decodes a shortened URL to its original URL.

:type shortUrl: str
:rtype: str
"""



# Your Codec object will be instantiated and called as such:
# codec = Codec()
# codec.decode(codec.encode(url))
```

JavaScript:

```
/**
* Encodes a URL to a shortened URL.
*
* @param {string} longUrl
* @return {string}
*/
var encode = function(longUrl) {

};

/**
* Decodes a shortened URL to its original URL.
*
* @param {string} shortUrl
* @return {string}
*/
var decode = function(shortUrl) {

};

/**
* Your functions will be called as such:
* decode(encode(url));
*/
```

TypeScript:

```
/**
* Encodes a URL to a shortened URL.
*/
function encode(longUrl: string): string {

};

/**
* Decodes a shortened URL to its original URL.
*/
function decode(shortUrl: string): string {

};

/**
* Your functions will be called as such:
* decode(encode(strs));
*/
```

C#:

```
public class Codec {

// Encodes a URL to a shortened URL
public string encode(string longUrl) {

}

// Decodes a shortened URL to its original URL.
public string decode(string shortUrl) {

}
}

// Your Codec object will be instantiated and called as such:
// Codec codec = new Codec();
// codec.decode(codec.encode(url));
```

C:

```
/** Encodes a URL to a shortened URL. */
char* encode(char* longUrl) {

}

/** Decodes a shortened URL to its original URL. */
char* decode(char* shortUrl) {

}

// Your functions will be called as such:
// char* s = encode(s);
// decode(s);
```

Go:

```
type Codec struct {

}


func Constructor() Codec {

}

// Encodes a URL to a shortened URL.
func (this *Codec) encode(longUrl string) string {

}

// Decodes a shortened URL to its original URL.
func (this *Codec) decode(shortUrl string) string {

}


/**
* Your Codec object will be instantiated and called as such:
* obj := Constructor();
* url := obj.encode(longUrl);
* ans := obj.decode(url);
*/
```

Kotlin:

```
class Codec() {
// Encodes a URL to a shortened URL.
fun encode(longUrl: String): String {

}

// Decodes a shortened URL to its original URL.
fun decode(shortUrl: String): String {

}
}

/**
* Your Codec object will be instantiated and called as such:
* var obj = Codec()
* var url = obj.encode(longUrl)
* var ans = obj.decode(url)
*/
```

Swift:

```
class Codec {
// Encodes a URL to a shortened URL.
func encode(_ longUrl: String) -> String {

}

// Decodes a shortened URL to its original URL.
func decode(_ shortUrl: String) -> String {

}
}

/**
* Your Codec object will be instantiated and called as such:
* let obj = Codec()
* val s = obj.encode(longUrl)
* let ans = obj.decode(s)
*/
```

Rust:

```
struct Codec {

}

/**
* `&self` means the method takes an immutable reference.
* If you need a mutable reference, change it to `&mut self` instead.
*/
impl Codec {
fn new() -> Self {

}

// Encodes a URL to a shortened URL.
fn encode(&self, longURL: String) -> String {

}

// Decodes a shortened URL to its original URL.
fn decode(&self, shortURL: String) -> String {

}
}

/**
* Your Codec object will be instantiated and called as such:
* let obj = Codec::new();
* let s: String = obj.encode(strs);
* let ans: VecVec<String> = obj.decode(s);
*/
```

Ruby:

```
# Encodes a URL to a shortened URL.
#
# @param {string} longUrl
# @return {string}
def encode(longUrl)

end
```

```
# Decodes a shortened URL to its original URL.
#
# @param {string} shortUrl
# @return {string}
def decode(shortUrl)

end



# Your functions will be called as such:
# decode(encode(url))
```

PHP:

```
class Codec {
/**
* Encodes a URL to a shortened URL.
* @param String $longUrl
* @return String
*/
function encode($longUrl) {

}

/**
* Decodes a shortened URL to its original URL.
* @param String $shortUrl
* @return String
*/
function decode($shortUrl) {

}
}

/**
* Your Codec object will be instantiated and called as such:
* $obj = Codec();
* $s = $obj->encode($longUrl);
* $ans = $obj->decode($s);
*/
```

Scala:

```
class Codec {
// Encodes a URL to a shortened URL.
def encode(longURL: String): String = {

}

// Decodes a shortened URL to its original URL.
def decode(shortURL: String): String = {

}
}

/**
* Your Codec object will be instantiated and called as such:
* var obj = new Codec()
* val s = obj.encode(longURL)
* val ans = obj.decode(s)
*/
```

Solutions

C++ Solution:

```
/*
* Problem: Encode and Decode TinyURL
* Difficulty: Medium
* Tags: string, hash
*
* 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:

// Encodes a URL to a shortened URL.
string encode(string longUrl) {
}

// Decodes a shortened URL to its original URL.
string decode(string shortUrl) {

}
};

// Your Solution object will be instantiated and called as such:
// Solution solution;
// solution.decode(solution.encode(url));
```

Java Solution:

```
/**
* Problem: Encode and Decode TinyURL
* Difficulty: Medium
* Tags: string, hash
*
* 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 Codec {

// Encodes a URL to a shortened URL.
public String encode(String longUrl) {

}

// Decodes a shortened URL to its original URL.
public String decode(String shortUrl) {

}
}

// Your Codec object will be instantiated and called as such:
// Codec codec = new Codec();
// codec.decode(codec.encode(url));
```

Python3 Solution:

```
"""
Problem: Encode and Decode TinyURL
Difficulty: Medium
Tags: string, hash

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 Codec:

def encode(self, longUrl: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Codec:

def encode(self, longUrl):
"""Encodes a URL to a shortened URL.

:type longUrl: str
:rtype: str
"""



def decode(self, shortUrl):
"""Decodes a shortened URL to its original URL.

:type shortUrl: str
:rtype: str
"""



# Your Codec object will be instantiated and called as such:
# codec = Codec()
# codec.decode(codec.encode(url))
```

JavaScript Solution:

```
/**
* Problem: Encode and Decode TinyURL
* Difficulty: Medium
* Tags: string, hash
*
* 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
*/

/**
* Encodes a URL to a shortened URL.
*
* @param {string} longUrl
* @return {string}
*/
var encode = function(longUrl) {

};

/**
* Decodes a shortened URL to its original URL.
*
* @param {string} shortUrl
* @return {string}
*/
var decode = function(shortUrl) {

};

/**
* Your functions will be called as such:
* decode(encode(url));
*/
```

TypeScript Solution:

```
/**
* Problem: Encode and Decode TinyURL
* Difficulty: Medium
* Tags: string, hash
```

```
*
* 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
*/

/**
* Encodes a URL to a shortened URL.
*/
function encode(longUrl: string): string {

};

/**
* Decodes a shortened URL to its original URL.
*/
function decode(shortUrl: string): string {

};

/**
* Your functions will be called as such:
* decode(encode(strs));
*/
```

C# Solution:

```
/*
* Problem: Encode and Decode TinyURL
* Difficulty: Medium
* Tags: string, hash
*
* 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 Codec {

// Encodes a URL to a shortened URL
public string encode(string longUrl) {
}

// Decodes a shortened URL to its original URL.
public string decode(string shortUrl) {

}
}

// Your Codec object will be instantiated and called as such:
// Codec codec = new Codec();
// codec.decode(codec.encode(url));
```

C Solution:

```
/*
* Problem: Encode and Decode TinyURL
* Difficulty: Medium
* Tags: string, hash
*
* 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
*/

/** Encodes a URL to a shortened URL. */
char* encode(char* longUrl) {

}

/** Decodes a shortened URL to its original URL. */
char* decode(char* shortUrl) {

}

// Your functions will be called as such:
// char* s = encode(s);
// decode(s);
```

Go Solution:

```
// Problem: Encode and Decode TinyURL
// Difficulty: Medium
// Tags: string, hash
//
// 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

type Codec struct {

}


func Constructor() Codec {

}

// Encodes a URL to a shortened URL.
func (this *Codec) encode(longUrl string) string {

}

// Decodes a shortened URL to its original URL.
func (this *Codec) decode(shortUrl string) string {

}


/**
* Your Codec object will be instantiated and called as such:
* obj := Constructor();
* url := obj.encode(longUrl);
* ans := obj.decode(url);
*/
```

Kotlin Solution:

```
class Codec() {
// Encodes a URL to a shortened URL.
fun encode(longUrl: String): String {

}
// Decodes a shortened URL to its original URL.
fun decode(shortUrl: String): String {

}
}

/**
* Your Codec object will be instantiated and called as such:
* var obj = Codec()
* var url = obj.encode(longUrl)
* var ans = obj.decode(url)
*/
```

Swift Solution:

```
class Codec {
// Encodes a URL to a shortened URL.
func encode(_ longUrl: String) -> String {

}

// Decodes a shortened URL to its original URL.
func decode(_ shortUrl: String) -> String {

}
}

/**
* Your Codec object will be instantiated and called as such:
* let obj = Codec()
* val s = obj.encode(longUrl)
* let ans = obj.decode(s)
*/
```

Rust Solution:

```
// Problem: Encode and Decode TinyURL
// Difficulty: Medium
// Tags: string, hash
//
```

```
// 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

struct Codec {

}

/**
* `&self` means the method takes an immutable reference.
* If you need a mutable reference, change it to `&mut self` instead.
*/
impl Codec {
fn new() -> Self {

}

// Encodes a URL to a shortened URL.
fn encode(&self, longURL: String) -> String {

}

// Decodes a shortened URL to its original URL.
fn decode(&self, shortURL: String) -> String {

}
}

/**
* Your Codec object will be instantiated and called as such:
* let obj = Codec::new();
* let s: String = obj.encode(strs);
* let ans: VecVec<String> = obj.decode(s);
*/
```

Ruby Solution:

```
# Encodes a URL to a shortened URL.
#
# @param {string} longUrl
# @return {string}
```

```
def encode(longUrl)

end

# Decodes a shortened URL to its original URL.
#
# @param {string} shortUrl
# @return {string}
def decode(shortUrl)

end


# Your functions will be called as such:
# decode(encode(url))
```

PHP Solution:

```
class Codec {
/**
* Encodes a URL to a shortened URL.
* @param String $longUrl
* @return String
*/
function encode($longUrl) {

}

/**
* Decodes a shortened URL to its original URL.
* @param String $shortUrl
* @return String
*/
function decode($shortUrl) {

}
}

/**
* Your Codec object will be instantiated and called as such:
* $obj = Codec();
* $s = $obj->encode($longUrl);
* $ans = $obj->decode($s);
*/
```

Scala Solution:

```
class Codec {
// Encodes a URL to a shortened URL.
def encode(longURL: String): String = {

}

// Decodes a shortened URL to its original URL.
def decode(shortURL: String): String = {

}
}

/**
* Your Codec object will be instantiated and called as such:
* var obj = new Codec()
* val s = obj.encode(longURL)
* val ans = obj.decode(s)
*/
```