Problem 271: Encode and Decode Strings

Problem Information

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

Problem Description

Design an algorithm to encode

a list of strings

to

a string

. The encoded string is then sent over the network and is decoded back to the original list of strings.

Machine 1 (sender) has the function:

string encode(vector<string> strs) { // ... your code return encoded_string; }

Machine 2 (receiver) has the function:

vector<string> decode(string s) { //... your code return strs; }

So Machine 1 does:

string encoded_string = encode(strs);

and Machine 2 does:

vector<string> strs2 = decode(encoded_string);

strs2

in Machine 2 should be the same as

strs

in Machine 1.

Implement the

encode

and

decode

methods.

You are not allowed to solve the problem using any serialize methods (such as

eval

).

Example 1:

Input:

dummy_input = ["Hello","World"]

Output:

["Hello","World"]

Explanation:

Machine 1: Codec encoder = new Codec(); String msg = encoder.encode(strs); Machine 1 ---msg---> Machine 2

Machine 2: Codec decoder = new Codec(); String[] strs = decoder.decode(msg);

Example 2:

Input:

dummy_input = [""]

Output:

[""]

Constraints:

1 <= strs.length <= 200

0 <= strs[i].length <= 200

strs[i]

contains any possible characters out of

256

valid ASCII characters.

Follow up:

Could you write a generalized algorithm to work on any possible set of characters?

Code Snippets

C++:

```
class Codec {
public:

// Encodes a list of strings to a single string.
string encode(vector<string>& strs) {

}

// Decodes a single string to a list of strings.
vector<string> decode(string s) {

}
};

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

Java:

```
public class Codec {

// Encodes a list of strings to a single string.
public String encode(List<String> strs) {

}

// Decodes a single string to a list of strings.
public List<String> decode(String s) {

}
}

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

Python3:

```
class Codec:
def encode(self, strs: List[str]) -> str:
"""Encodes a list of strings to a single string.
"""



def decode(self, s: str) -> List[str]:
```

```
"""Decodes a single string to a list of strings.
"""



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

Python:

```
class Codec:

def encode(self, strs):
"""Encodes a list of strings to a single string.

:type strs: List[str]
:rtype: str
"""


def decode(self, s):
"""Decodes a single string to a list of strings.

:type s: str
:rtype: List[str]
"""


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

JavaScript:

```
/**
* Encodes a list of strings to a single string.
*
* @param {string[]} strs
* @return {string}
*/
var encode = function(strs) {
};

/**
* Decodes a single string to a list of strings.
*
* @param {string} s
* @return {string[]}
*/
var decode = function(s) {

};

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

TypeScript:

```
/**
* Encodes a list of strings to a single string.
*/
function encode(strs: string[]): string {

};

/**
* Decodes a single string to a list of strings.
*/
function decode(s: string): string[] {

};

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

C#:

```
public class Codec {

// Encodes a list of strings to a single string.
public string encode(IList<string> strs) {

}

// Decodes a single string to a list of strings.
public IList<string> decode(string s) {

}
}

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

C:

```
/** Encodes a list of strings to a single string */
char* encode(char** strs, int strsSize) {

}

/**
* Decodes a single string to a list of strings.
*
* Return an array of size *returnSize.
* Note: The returned array must be malloced, assume caller calls free().
*/
char** decode(char* s, int* returnSize) {

}

// Your functions will be called as such:
// char* s = encode(strs, strsSize);
// decode(s, &returnSize);
```

Go:

```
type Codec struct {

}
```

```
// Encodes a list of strings to a single string.
func (codec *Codec) Encode(strs []string) string {

}

// Decodes a single string to a list of strings.
func (codec *Codec) Decode(strs string) []string {

}

// Your Codec object will be instantiated and called as such:
// var codec Codec
// codec.Decode(codec.Encode(strs));
```

Kotlin:

```
class Codec {
// Encodes a list of strings to a single string.
fun encode(strs: List<String>): String {

}

// Decodes a single string to a list of strings.
fun decode(s: String): List<String> {

}
}

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

Swift:

```
class Codec {
func encode(_ strs: [String]) -> String {

}
func decode(_ s: String) -> [String] {

}
}

/**
* Your Codec object will be instantiated and called as such:
* let obj = Codec()
* val s = obj.encode(strs)
* 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 {

}

fn encode(&self, strs: Vec<String>) -> String {

}

fn decode(&self, s: String) -> Vec<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 list of strings to a single string.
#
# @param {string[]} strs
# @return {string}
def encode(strs)

end

# Decodes a single string to a list of strings.
#
# @param {string} s
# @return {string[]}
def decode(s)

end



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

PHP:

```
class Codec {
/**
* @param String[] $strs
* @return String
*/
function encode($strs) {

}

/**
* @param String $s
* @return String[]
*/
function decode($s) {

}

}

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

Scala:

```
class Codec {
// Encodes a list of strings to a single string.
def encode(strs: List[String]): String = {

}

// Decodes a single string to a list of strings.
def decode(s: String): List[String] = {

}
}

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

Solutions

C++ Solution:

```
/*
* Problem: Encode and Decode Strings
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Codec {
public:

// Encodes a list of strings to a single string.
string encode(vector<string>& strs) {

}

// Decodes a single string to a list of strings.
vector<string> decode(string s) {

}
};

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

Java Solution:

```
/**
* Problem: Encode and Decode Strings
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Codec {

// Encodes a list of strings to a single string.
public String encode(List<String> strs) {

}
// Decodes a single string to a list of strings.
public List<String> decode(String s) {

}
}

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

Python3 Solution:

```
"""
Problem: Encode and Decode Strings
Difficulty: Medium
Tags: array, string

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Codec:
def encode(self, strs: List[str]) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Codec:

def encode(self, strs):
"""Encodes a list of strings to a single string.

:type strs: List[str]
:rtype: str
"""



def decode(self, s):
"""Decodes a single string to a list of strings.
```

```


:type s: str
:rtype: List[str]
"""



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

JavaScript Solution:

```
/**
* Problem: Encode and Decode Strings
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Encodes a list of strings to a single string.
*
* @param {string[]} strs
* @return {string}
*/
var encode = function(strs) {

};

/**
* Decodes a single string to a list of strings.
*
* @param {string} s
* @return {string[]}
*/
var decode = function(s) {

};
```

```


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

TypeScript Solution:

```
/**
* Problem: Encode and Decode Strings
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Encodes a list of strings to a single string.
*/
function encode(strs: string[]): string {

};

/**
* Decodes a single string to a list of strings.
*/
function decode(s: string): string[] {

};

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

C# Solution:

```
/*
* Problem: Encode and Decode Strings
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Codec {

// Encodes a list of strings to a single string.
public string encode(IList<string> strs) {

}

// Decodes a single string to a list of strings.
public IList<string> decode(string s) {

}
}

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

C Solution:

```
/*
* Problem: Encode and Decode Strings
* Difficulty: Medium
* Tags: array, string
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/** Encodes a list of strings to a single string */
char* encode(char** strs, int strsSize) {

}
```

```
/**
* Decodes a single string to a list of strings.
*
* Return an array of size *returnSize.
* Note: The returned array must be malloced, assume caller calls free().
*/
char** decode(char* s, int* returnSize) {

}

// Your functions will be called as such:
// char* s = encode(strs, strsSize);
// decode(s, &returnSize);
```

Go Solution:

```
// Problem: Encode and Decode Strings
// Difficulty: Medium
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

type Codec struct {

}

// Encodes a list of strings to a single string.
func (codec *Codec) Encode(strs []string) string {

}

// Decodes a single string to a list of strings.
func (codec *Codec) Decode(strs string) []string {

}

// Your Codec object will be instantiated and called as such:
// var codec Codec
// codec.Decode(codec.Encode(strs));
```

Kotlin Solution:

```
class Codec {
// Encodes a list of strings to a single string.
fun encode(strs: List<String>): String {

}

// Decodes a single string to a list of strings.
fun decode(s: String): List<String> {

}
}

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

Swift Solution:

```
class Codec {
func encode(_ strs: [String]) -> String {

}

func decode(_ s: String) -> [String] {

}
}

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

Rust Solution:

```
// Problem: Encode and Decode Strings
// Difficulty: Medium
// Tags: array, string
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

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 {

}

fn encode(&self, strs: Vec<String>) -> String {

}

fn decode(&self, s: String) -> Vec<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 list of strings to a single string.
#
# @param {string[]} strs
# @return {string}
def encode(strs)

end

# Decodes a single string to a list of strings.
#
# @param {string} s
# @return {string[]}
def decode(s)

end


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

PHP Solution:

```
class Codec {
/**
* @param String[] $strs
* @return String
*/
function encode($strs) {

}

/**
* @param String $s
* @return String[]
*/
function decode($s) {

}
}

/**
* Your Codec object will be instantiated and called as such:
```

```
* $obj = Codec();
* $s = $obj->encode($strs);
* $ans = $obj->decode($s);
*/
```

Scala Solution:

```
class Codec {
// Encodes a list of strings to a single string.
def encode(strs: List[String]): String = {

}

// Decodes a single string to a list of strings.
def decode(s: String): List[String] = {

}
}

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