Previous Next

# 271. Encode and Decode Strings

July 6, 2019 | 15.4K views

★★★★★ Average Rating: 4.78 (9 votes)

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.

**Note:**

- The string may contain any possible characters out of 256 valid ascii characters. Your algorithm should be generalized enough to work on any possible characters.
- Do not use class member/global/static variables to store states. Your encode and decode algorithms should be stateless.
- Do not rely on any library method such as `eval` or serialize methods. You should implement your own encode/decode algorithm.

## Solution

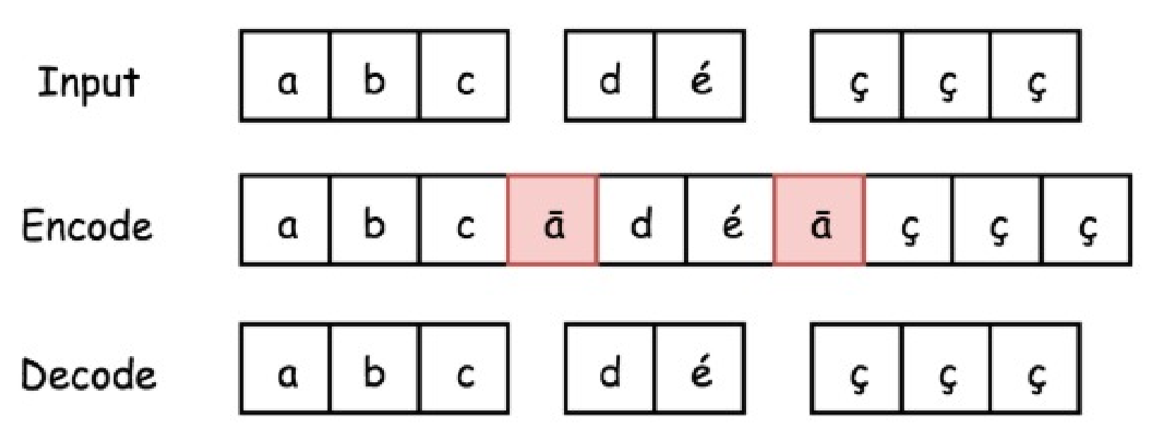
### Approach 1: Non-ASCII Delimiter

**Intuition**

Naive solution here is to join strings using delimiters.

> What to use as a delimiter? Each string may contain any possible characters out of 256 valid ascii characters.

Seems like one has to use non-ASCII unichar character, for example `unichr(257)` in Python and `Character.toString((char)257)` in Java (it's character `ā`).



Here it's convenient to use two different non-ASCII characters, to distinguish between situations of "empty array" and of "array of empty strings".

**Implementation**

Use `split` in Java with a second argument `-1` to make it work as `split` in Python.

Java Python Copy

```
public class Codec {
  // Encodes a list of strings to a single string.
  public String encode(List<String> strs) {
    if (strs.size() == 0) return Character.toString((char)258);

    String d = Character.toString((char)257);
    StringBuilder sb = new StringBuilder();
    for(String s: strs) {
      sb.append(s);
      sb.append(d);
    }
    sb.deleteCharAt(sb.length() - 1);
    return sb.toString();
  }

  // Decodes a single string to a list of strings.
  public List<String> decode(String s) {
    String d = Character.toString((char)258);
    if (s.equals(d)) return new ArrayList();

    d = Character.toString((char)257);
    return Arrays.asList(s.split(d, -1));
  }
}
```

**Complexity Analysis**

- Time complexity : $\mathcal{O}(N)$ both for encode and decode, where N is a number of strings in the input array.
- Space complexity : $\mathcal{O}(1)$ for encode to keep the output, since the output is one string. $\mathcal{O}(N)$ for decode keep the output, since the output is an array of strings.

---

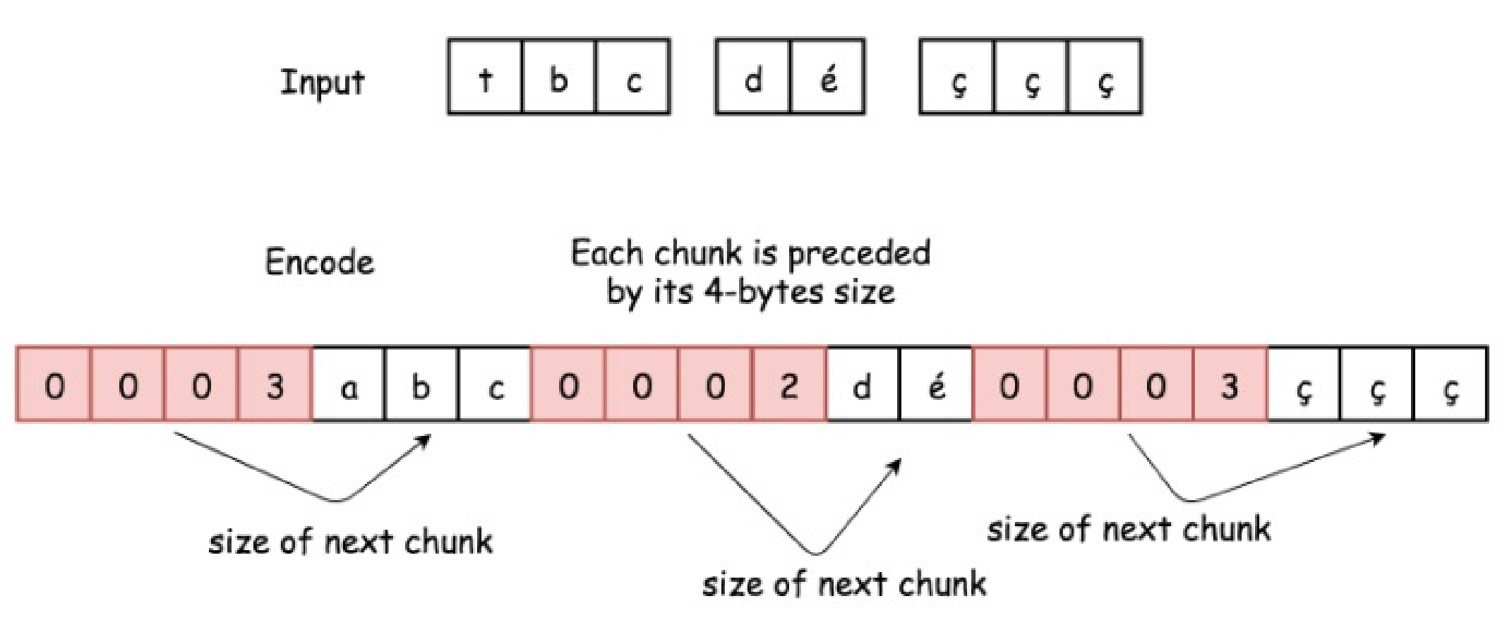
### Approach 2: Chunked Transfer Encoding

Pay attention to this approach because last year Google likes to ask that sort of low-level optimisation. Serialize and deserialize BST problem is a similar example.

This approach is based on the encoding used in HTTP v1.1. It doesn't depend on the set of input characters, and hence is more versatile and effective than Approach 1.

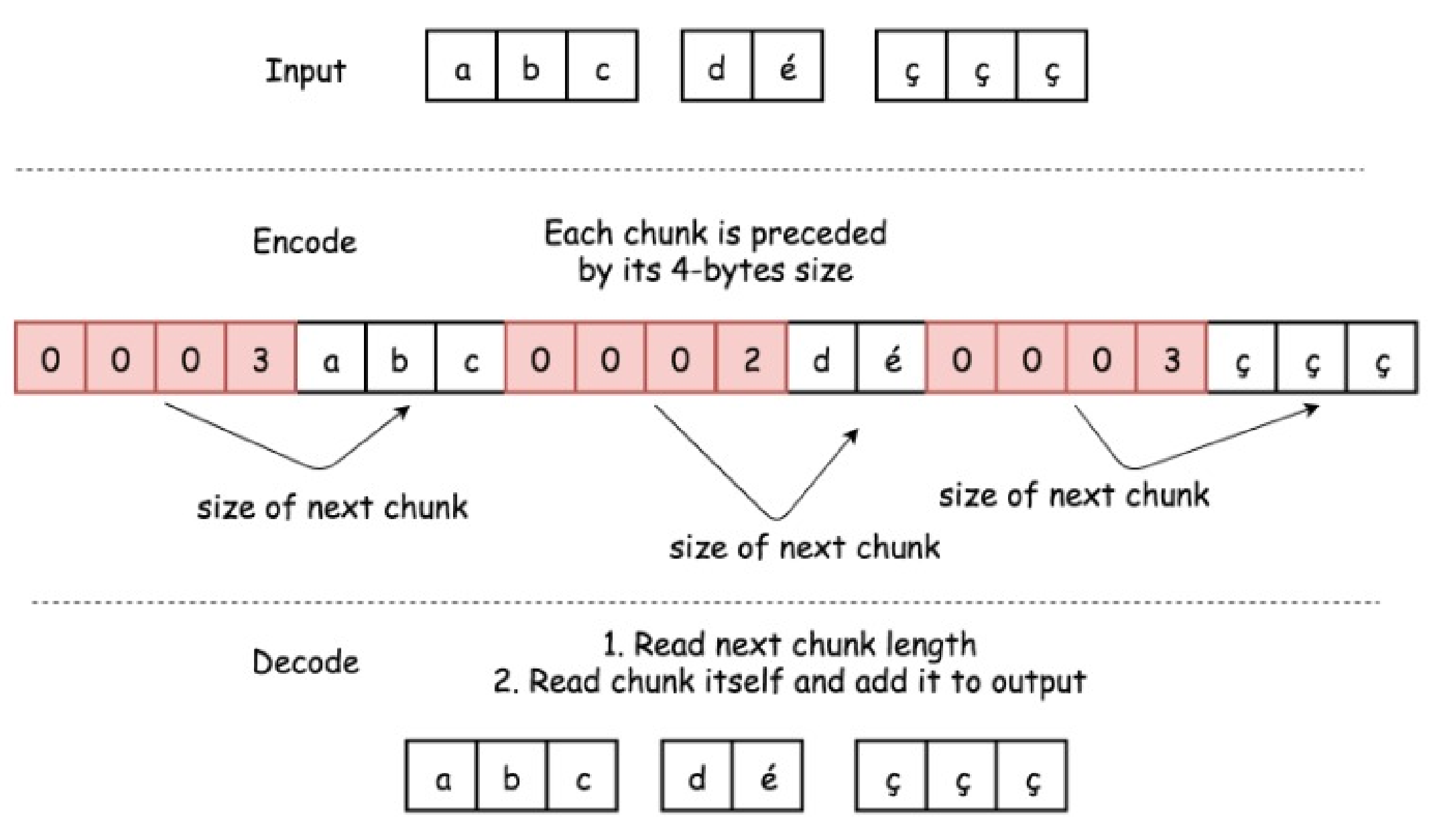
> Data stream is divided into chunks. Each chunk is preceded by its size in bytes.

**Encoding Algorithm**



- Iterate over the array of chunks, i.e. strings.
  - For each chunk compute its length, and convert that length into 4-bytes string.
  - Append to encoded string :
    - 4-bytes string with information about chunk size in bytes.
    - Chunk itself.
- Return encoded string.

**Decoding Algorithm**



- Iterate over the encoded string with a pointer `i` initiated as 0. While `i < n`:
  - Read 4 bytes `s[i: i + 4]`. It's chunk size in bytes. Convert this 4-bytes string to integer `length`.
  - Move the pointer by 4 bytes `i += 4`.
  - Append to the decoded array string `s[i: i + length]`.
  - Move the pointer by `length` bytes `i += length`.
- Return decoded array of strings.

**Implementation**

Java Python Copy

```
public class Codec {
  // Encodes string length to bytes string
  public String intToString(String s) {
    int x = s.length();
    char[] bytes = new char[4];
    for(int i = 3; i > -1; --i) {
      bytes[3 - i] = (char) (x >> (i * 8) & 0xff);
    }
    return new String(bytes);
  }

  // Encodes a list of strings to a single string.
  public String encode(List<String> strs) {
    StringBuilder sb = new StringBuilder();
    for(String s: strs) {
      sb.append(intToString(s));
      sb.append(s);
    }
    return sb.toString();
  }

  // Decodes bytes string to integer
  public int stringToInt(String bytesStr) {
    int result = 0;
    for(char b : bytesStr.toCharArray())
      result = (result << 8) + (int)b;
    return result:
```

**Complexity Analysis**

- Time complexity : $\mathcal{O}(N)$ both for encode and decode, where N is a number of strings in the input array.
- Space complexity : $\mathcal{O}(1)$ for encode to keep the output, since the output is one string. $\mathcal{O}(N)$ for decode keep the output, since the output is an array of strings.

Analysis written by @liaison and @andvary

Rate this article: ★★★★★

Previous Next

Comments: 17 Sort By

Type comment here... (Markdown is supported)

Preview Post

san_py ★ 202 February 29, 2020 10:22 PM

# Understanding bytes = [chr(x >> (i * 8) & 0xff) for i in range(4)]

Read More

13 Share Reply

jrhero ★ 8 September 9, 2019 2:39 AM

what if length of string exceeds the Integer.MAX_VALUE (which means 4 butes cannot hold the length)?

8 Share Reply

SHOW 3 REPLIES

willye ★ 656 September 9, 2019 11:32 AM

Chunked encoding is too clever for me... I dont think I can solve this in an interview lol

10 Share Reply

SHOW 1 REPLY

xitrium ★ 3 November 19, 2019 12:29 AM

The space analysis is wrong here - just because a solution uses one string doesn't make it O(1) as strings have a variable size. Clearly the output of `encode` in both of these examples is linearly proportional to the size of the input.

3 Share Reply

cipone ★ 6 July 8, 2019 5:54 PM

HI, I am a little confused by this statement. **For each chunk compute its length, and convert that length into 4-bytes string.** It is actually a 8 bytes string since a char is 2 bytes (16 bits). And then the stored amount is sub-optimal, it uses 64 bits to store a 2**32 number.

4 Share Reply

SHOW 4 REPLIES

mehta_vijapur ★ 9 July 8, 2019 6:28 AM

what is "BE CodecDriver error? Python 3.* version do not have the issue atleast I do not see it. Also python 3.0 version uses `chr()` instead of `unichr`

2 Share Reply

SHOW 2 REPLIES

sin1080 ★ 40 July 6, 2019 9:49 PM

What about just implement the full HTTP Chunked Transfer Encoding and dump length as readable text representations. If you dump integer as bytes you can easily encounter endian issues between machines and things like strict pointer aliasing violation if you are using C/C++ and are not careful enough about dark corners of the language standards (E.g. if you used an int* to dereference data stored in a char* / string, you entered the land of undefined behaviour).

Read More

2 Share Reply

SHOW 4 REPLIES

lenchen1112 ★ 591 December 10, 2019 1:49 PM

Approach 1 by using escape character

```
class Codec:
    def encode(self, strs: [str]) -> str:
        return str(len(strs)) + '\t' + '\t'.join(strs)
```

Read More

1 Share Reply

m_2010 ★ 0 December 4, 2019 2:56 AM

Why no `Swift` implementation is possible?

0 Share Reply

gfmacode ★ 22 February 20, 2020 9:18 PM

@andvary can you explain this line `bytes[3 - i] = (char) (x >> (i * 8) & 0xff);` or recommend a resource if it is something I should know. Thank you.
`

0 Share Reply

SHOW 3 REPLIES

‹ 1 2 ›