Lu Chang & Zhang Mengjiao

LeetCode Solutions

First Edition

Preface

This project is aimed to accompany my girlfriend @MengjiaoZhang to learn git, programming skills and algorithms.

Here, I want to thank LeetCode providing these problems. It will be better if all problems can be accessed without subscribing. (ಡωಡ)hiahiahia

Contents

Chapter 1

Solutions for Algorithms

"*Perhaps the most important principle for the good algorithm designer is to refuse to be content.*"

— Alfred V. Aho

535. Encode and Decode TinyURL

Difficulty

Medium

Tags

Cryptology

Description

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 the `encode` and `decode` methods for the TinyURL service. 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.

Analysis

This is an open problem where numerous solutions can be applied. We can even keep the original url as encode and decode, although it is meaningless.

We should to pay attention to these limitations:

- Correctness: We must make sure that the decoded url is the same as

the original url.

- Uniqueness: Each url must have an unique encoded url, and an encoded url must be decoded to a single url.
- Simplicity: The aim to encode a url is to make it easy to share or write, so we need to make the encoded url as simple as possible.

We can use current popular encode/decode algorithms such as `AES` , `DES` , but in this problem, we just design a simpler algorithm to encode and decode a url.

```
function encode(url)
    return hex(current number of urls in hash table)
end

function decode(encoded_url)
    return (hash table).find(encoded_url)
end
```

Solution

C++

```cpp
typedef unordered_map<string, string> Urlmap;

class Solution {
public:

    Urlmap urlmap;

    // Encodes a URL to a shortened URL.
    string encode(string longUrl) {
        size_t size = urlmap.size();
        stringstream encoded;
        encoded << hex << size;
        string encoded_url = encoded.str();
        urlmap.insert(make_pair(encoded_url, longUrl));
        return encoded_url;
    }

    // Decodes a shortened URL to its original URL.
    string decode(string shortUrl) {
    Urlmap::iterator it = urlmap.find(shortUrl);
    if (it == urlmap.end()) return NULL;
        return it->second;
    }
};
```

Indexes of Solutions for Algorithms

Tags of Solutions for Algorithms

Cryptology

535. Encode and Decode TinyURL

Chapter 2

Solutions for Databases