# Lab 2

Gray code was proposed by Frank Gray of Bell Laboratories in 1940. It is used to prevent errors when transmitting signals in the PCM (Pulse Code Modulation) method. It obtained a US patent on March 17, 1953. The Gray code is a set of numbers. Only one bit changes between two adjacent numbers. It is an unweighted number, and the sequence of the Gray code is not unique.

The principle of Gray code avoiding signal transmission errors：

The traditional binary system, for example, the representation of the number 3 is 011. To switch to the adjacent number 4, which is 100, all three bits in the device have to be converted. Therefore, the device will experience a short period of time during the incomplete conversion process. Yes, 010, 001, 101, 110, 111 and other states, which represent 2, 1, 5, 6, 7. Therefore, this digital encoding method has a relatively large error range when adjacent digital conversions. The invention of Gray code is used to minimize the possibility of error. The coding method is defined as the difference of each adjacent number by only one bit. Therefore, it is also called the minimum difference code, which can make the device only Change the least number of bits to improve stability. The coding comparison of numbers 0～7 is as follows:

An **n-bit gray code sequence** is a sequence of 2n integers where:

* Every integer is in the **inclusive** range [0, 2n - 1],
* The first integer is 0,
* An integer appears **no more than once** in the sequence,
* The binary representation of every pair of **adjacent** integers differs by **exactly one bit**, and
* The binary representation of the **first** and **last** integers differs by **exactly one bit**.

Given an integer n, return *any valid****n-bit gray code sequence***.

First, I write a C++ library, which contains functions to generate Gray code sequences. The header files of this c++ library are as follows：

*//*

*// grayCode.h*

*//*

#ifndef GRAYCODE\_H

#define GRAYCODE\_H

class Solution {

public:

Solution();

long int getTime();

char\* grayCode(int *n*);

};

#endif

In order to detect the running time of the program, I also added a function to get the time. The functions of this c++ library are as follows

#include "grayCode.h"

#include <vector>

#include <iostream>

#include <sstream>

#include <string>

#include <ctime>

using namespace std;

Solution::Solution() {}

long int Solution::getTime() {

return clock();

}

char\* Solution::grayCode(int *n*) {

vector<int> result;

result.push\_back(0);

for (int i = 1; i <= *n*; i++) {

int previousSequenceLength = result.size();

int mask = 1 << (i - 1);

for (int j = previousSequenceLength - 1; j >= 0; j--) {

result.push\_back(mask + result[j]);

}

}

stringstream ss;

string str;

copy(result.begin(), result.end(), ostream\_iterator<int>(ss, ","));

str = ss.str();

char \*p=(char\*)str.c\_str();

cout << p << endl;

return p;

}

This C++ library contains two functions. The first function is grayCode, its input is an int type integer, and the output is a string. There is also a function getTime to get the current time, which returns the current CPU time, which can be divided by CLOCK\_PER\_SECOND to get the function running time.

Originally, I intended to let the function return a vector<int>, but when writing the WebIDL file, I did not find the corresponding Key name. So, I converted Vector<int> into a string meal, and found the corresponding Key name. The corresponding WebIDL file is as follows

interface Solution {

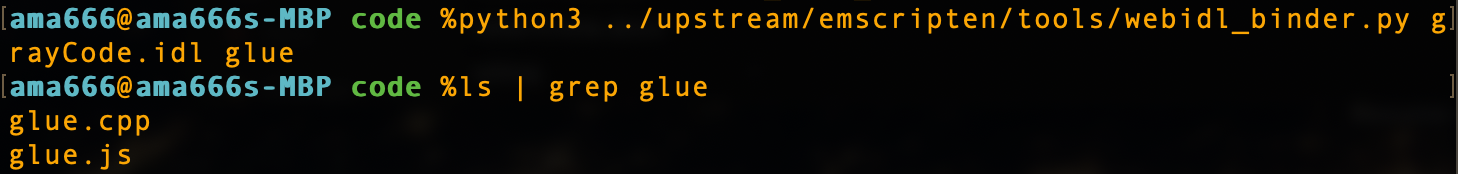
void Solution();

long getTime();

DOMString grayCode(long val);

};

After getting these three files, I used emscripten's own tool to generate glue code.



At this time, the two files obtained are glue.js and glue.cpp. Some changes need to be made here. The size\_t type is used in glue.cpp, and the stddef header file needs to be introduced so that there will be no problems in the next step of transcompiling.

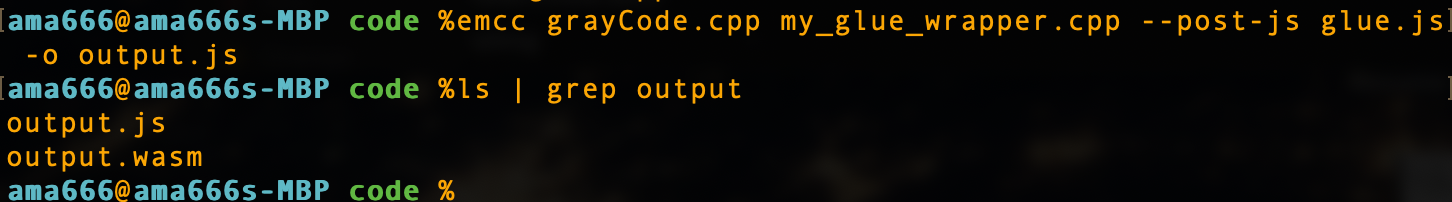


Next, you need to write a wrapper file, and write the files that need to be glued in it. The my\_glue\_wrapper.cpp file I wrote is as follows

#include "grayCode.h"

#include "glue.cpp"

Then use emcc to convert the c++ file into a javascript file.



Next, write an html file to call the “output” file output in this step. Here I use the Data class in javascript to get the time.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Get the Value of Text Input Field in JavaScript</title>

</head>

<body>

<p>An n-bit gray code sequence is a sequence of 2n integers where:</p>

<p>Every integer is in the inclusive range [0, 2n - 1],<br>The first integer is 0,<br>An integer appears no more than once in the sequence,<br>The binary representation of every pair of adjacent integers differs by exactly one bit, and<br>The binary representation of the first and last integers differs by exactly one bit.</p>

<p id="instruction">Type a integer, return any valid n-bit gray code sequence.</p>

<input type="text" placeholder="Type something..." id="myInput">

<button type="button" onclick="getInputValue();">Get Value</button>

<p id="demo">The results will be shown here</p>

<script type="text/JavaScript" src="output.js"></script>

<script>

function insertAfter( *newElement*, *targetElement* ){

var parent = *targetElement*.parentNode;

if( parent.lastChild == *targetElement* ){

parent.appendChild( *newElement*, *targetElement* );

}else{

parent.insertBefore( *newElement*, *targetElement*.nextSibling );

}

}

function getInputValue(){

*// Selecting the input element and get its value*

var inputVal = document.getElementById("myInput").value;

var sol = new Module.Solution();

var lastObj = document.getElementById("demo");

for (var i=1; i<=10; i++)

{

let tStart = new Date().valueOf();

console.log(tStart);

var ret = sol.grayCode(inputVal);

let tEnd = new Date().valueOf();

console.log(tEnd);

let t = tEnd - tStart;

var p = document.createElement("p");

p.innerHTML = "Round" + i + ": " + ret + " " + "Time: " + t + "ms";

insertAfter(p, lastObj);

lastObj = p;

}

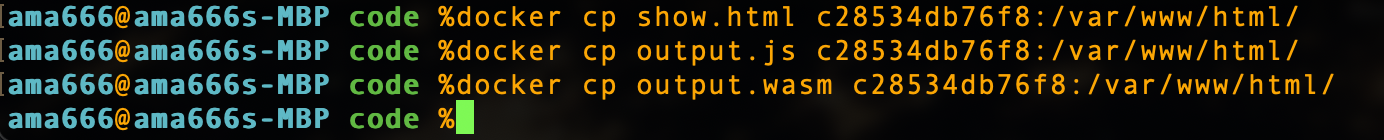
}

</script>

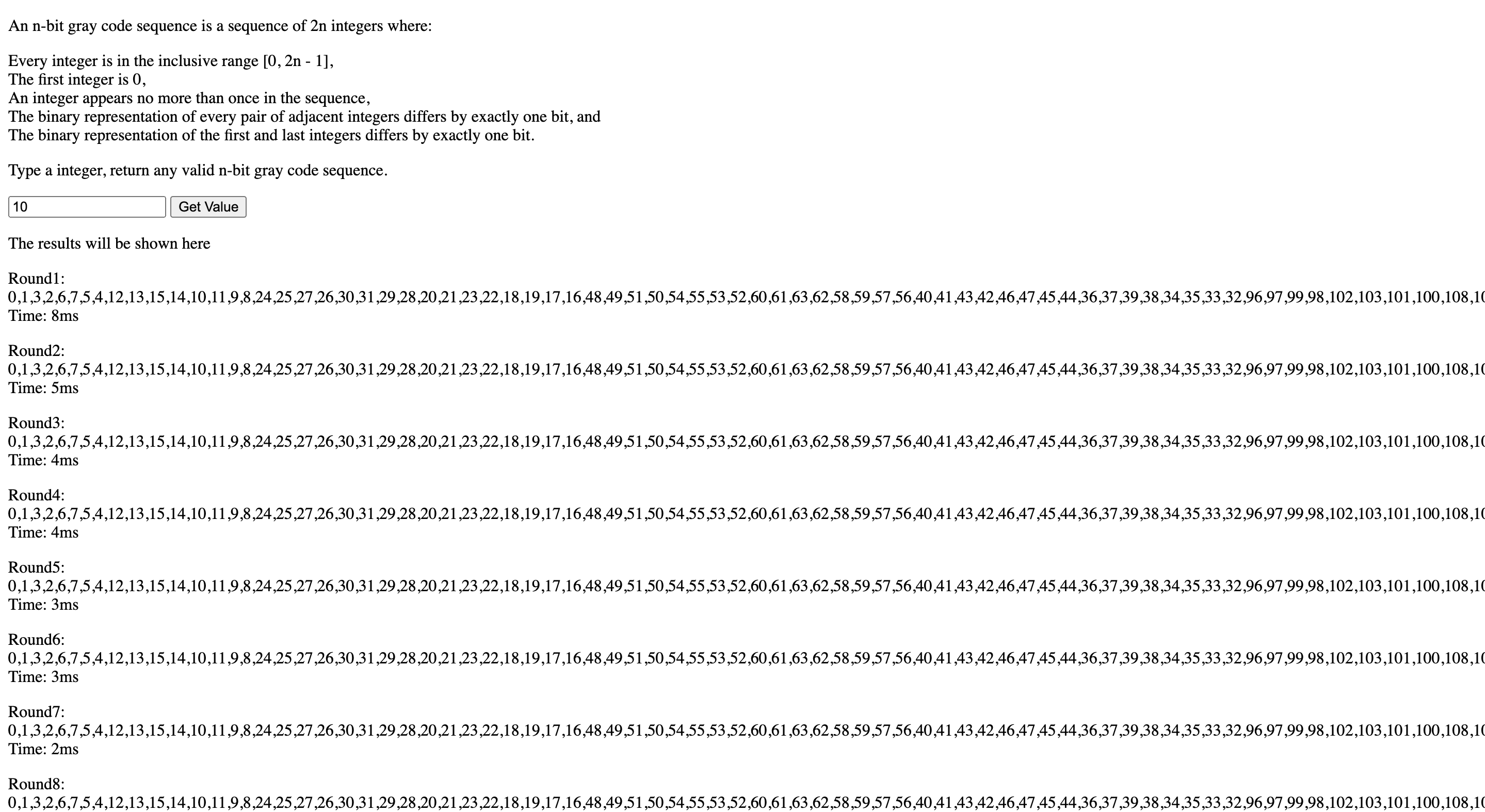
</body>

</html>

Copy the three files to the docker apache server. The setup process of this server can be found in lab1.



Visit show.html and enter 10 as the input of the grayCode function, click the button to get the gray code sequence, and execute it 10 times, you can see the time required for each execution.



Write a C++ program, reference the grayCode library written before, execute it in a loop ten times, and record the execution time of each time.

#include "grayCode.h"

#include <string>

#include <iostream>

#include <ctime>

using namespace std;

int main() {

Solution sol;

for (int i = 1; i <= 10; i++) {

long int startTime, endTime;

startTime = sol.getTime();

string str = sol.grayCode(10);

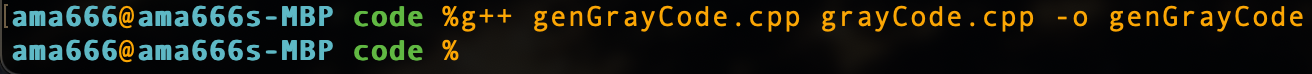
endTime = sol.getTime();

std::cout << str << std::endl;

cout<< "Round" << i << ": " << (float) (endTime-startTime)\*1000/CLOCKS\_PER\_SEC << "ms" << endl;

}

}



The running results are as follows, because the output is too long, I only screenshot the running time of each round.





















Comparing the running time of the two codes, the average running time of C++ is 0.5ms, while the average running time of Javascript is 3.8ms, which is significantly slower than C++

The standard deviation of the c++ execution result is 0.076, and the 95% confidence interval is 0.5+/-0.149 = [0.351, 0.649]

The standard deviation of the javascript execution program is 1.6, and the 95% confidence interval is 3.8+/-1.6 = [2.2, 5.4]

The running time of the first round of the two sets of data is not in the confidence interval, and the data of the other rounds are all in. Throwing away the results of the first round, the results obtained are statistically significant.