1. **代码实践**

**Leetcode题目4：二分查找**

#include <iostream>

#include <vector>

using namespace std;

/\*

class Solution {

public:

double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {

int m = nums1.size(), n = nums2.size();

int a = 0, b = 0;

double front = 0, current = 0, ans = 0;

for(int i = 0; i < (m + n) / 2 + 1; ++i){

front = current;

if((a >= m) || (b < n && nums1[a] > nums2[b])) //注意判断语句的顺序，防止数组访问越界

current = nums2[b++]; //指针指向较小的那个，保证两个数组后面的值都比当前current值大

else

current = nums1[a++];

}

if(1 == (m + n) % 2)

ans = current;

else

ans = (current + front) / 2.0;

return ans;

}

};\*/

//二分法

class Solution {

public:

double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {

int m = nums1.size(), n = nums2.size();

if(m > n){

vector<int> temp = nums1;

nums1 = nums2;

nums2 = temp;

int tmp = m;

m = n;

n = tmp;

}

int i = 0, j = 0; // i标记nums1的“分割点”，j标记nums2的“分割点”，i+j总等于half

int iMin = 0, iMax = m, half = (m + n + 1) / 2; //向下取整，故应为(m + n) + 1

while(iMin <= iMax){

i = (iMin + iMax) / 2;

j = half - i;

if(i < iMax && nums2[j - 1] > nums1[i]) iMin = i + 1; //i太小,需要增大左边界iMin

else if(i > iMin && nums1[i - 1] > nums2[j]) iMax = i - 1; //i太大,需要减小右边界iMax

else{ //找到合适的i了

double MaxLeft = 0;

if(i == 0) MaxLeft = nums2[j - 1];

else if(j == 0) MaxLeft = nums1[i - 1];

else MaxLeft = max(nums1[i - 1], nums2[j - 1]);

if((m + n) % 2) return MaxLeft; //奇数个元素，直接返回

double MinRight = 0;

if(i == m) MinRight = nums2[j];

else if(j == n) MinRight = nums1[i];

else MinRight = min(nums1[i], nums2[j]);

return (MaxLeft + MinRight) / 2.0; //偶数个元素，返回均值

}

}

return 0;

}

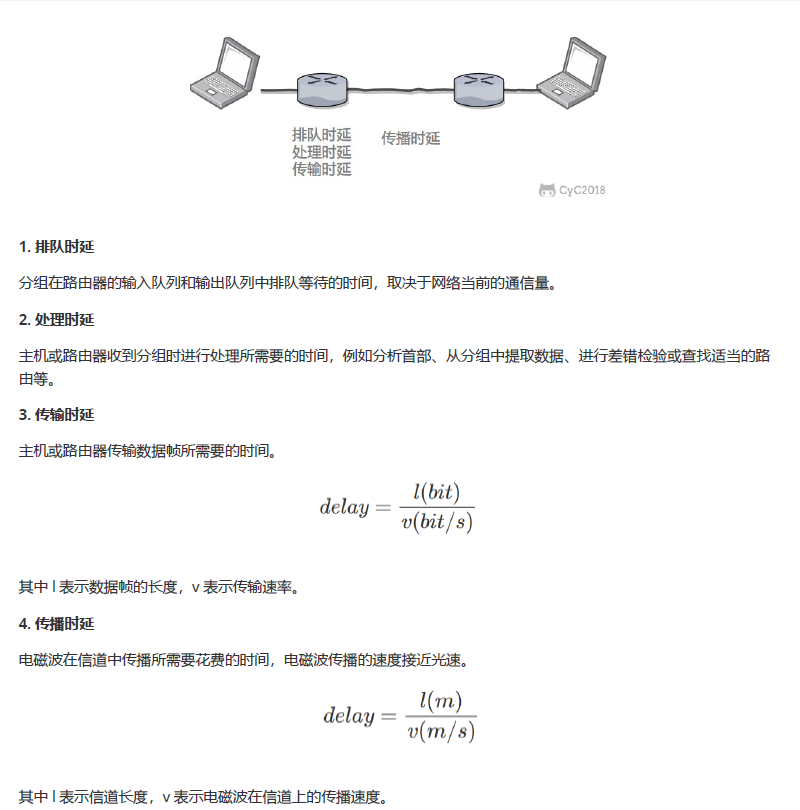
};

int main(){

return 0;

}

1. **计算机基础整理**



1. **开源特训营工作总结**
2. 补充小组成员至6个
3. 建立项目结构，每个人在git中建立自己的文件夹，用以记录自己的学习笔记；同时将已在本地跑通的疫情防控小程序的JAVA服务端源码和前端源码上传到小组的git仓库中，用于研究学习，后期尝试进行优化改进。