Fifth Study Report

Shuo Xu

June 24, 2018

Introduction

This week,I learned very little knowledge of MLP and finished an algorithm-related leetcode hard problem.In this report I only explained how I solved this problem.

Leetcode

Median of Two Sorted Arrays

Description

There are two sorted arrays nums1 and nums2 of size m and n respectively. Find the median of the two sorted arrays. The overall run time complexity should be O(log (m+n)).
Example 1:
nums1 = [1, 3]
nums2 = [2]
The median is 2.0
Example 2:
nums1 = [1, 2]
nums2 = [3, 4]
The median is (2 + 3)/2 = 2.5

Solution

The first idea is to merge two arrays and output corresponding elements based on the parity of the length of the array.Unfortunately, this solution does not meet the requirements of the problem.
Later I discovered that this problem has a certain relationship with the dichotomy search algorithm.So this is my second solution:
We already know that the arrays are already in order. So this problem is equivalent to finding the kth number in two arrays.First, assuming that the number of arrays is greater than k/2, we compare A[k/2-1] and B[k/2-1]. There are three results.If A[k/2-1] is less than B[k/2-1], this means that the elements of A[0] to A[k/2-1] are all in the first k elements after A and B are merged.So we can lose A.If A[k/2-1] bigger less than B[k/2-1], there will be similar conclusions. If they are equal, we have already found the kth number. Summarizing the above ideas, we can solve this problem with recursive methods. Of course, we must pay attention to special circumstances

code:

```
double findKth(vector<int>& nums1, int m, int start1, vector<int> &nusm2,
int n, int start2, int k){
        if (m >n)
                return findKth(b, n, start2, a, m, start1, k);
                //make m is equal or smaller than n
        if (m == 0)
                return nums2[k - 1];
        if (k == 1)
                return min(nums1[0], nums2[0]);


        int pa = min(k/2, m), pb = k - pa;
        if (nums1[star1 + pa - 1] < nums2[start2 + pb - 1])
                return findKth(nums1, m-pa, start1+pa, nums2, n,
                start2, k - pa);
        else if (nums1[start1 + pa - 1] > nums2[start2 + pb - 1])
                return findKth(nums1, m, start1, nums2, n-pb,
                start2+pb, k - pb);
        else
                return a[start1 + pa - 1];
}

class Solution
{
public:
        double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2){
                int total = nums1.size() + nums2.size();
                if (total & 0x1)  //Judging parity
                        return findKth(nums1, nums1.size(), nums2,
                        nums2.size(), total / 2 + 1);
                else
                        return (findKth(nums1, nums1.size(), nums2,
                        nums2.size(), total / 2)
                                        + findKth(nums1, nums1.size(), nums2,
                                        nums2.size(),total / 2 + 1)) / 2;
        }
};
```

another one:

```
  class Solution {
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
        vector<int> a;
        a.insert(a.end(), nums1.begin(), nums1.end());
        a.insert(a.end(), nums2.begin(), nums2.end());
        sort(a.begin(), a.end(), greater<int>());
        int len=a.size();
        double sum=0;
        if(len % 2){
            return a[len/2];
        }
        else{
            if(len ==2 )
                sum = (a[0] + a[1])/2.0;
            else
                sum=(a[len/2] + a[len/2-1])/2.0;
        }
        return sum;
    }
};
```

Summary And Thinking

Last week, I said that I would go to achieve MLP. I and Zhang Zhenchao found some code about MLP, but neither of us has understood them yet. Perhaps we need your guidance again. The inadequacy of my own time schedule led to a very low efficiency this week. I hope you can forgive me.