1. 两数之和 java

class Solution {

    public int[] twoSum(int[] nums, int target) {

        int i,j;

        int[] out = new int[2];

        for(i=0;i<nums.length-1;i++)

        {

            for(j=i+1;j<nums.length;j++)

            {

                if(nums[i]+nums[j]==target){

                    out[0] = i;out[1] = j;

                    return out;

                }

            }

        }

        return null;

    }

}

1. 两数之和 C

int\* twoSum(int\* nums, int numsSize, int target, int\* returnSize){

    int i,j;

    static int k[2];

    \*returnSize = 2;

    for(i=0;i<numsSize-1;i++)

    {

        for(j=i+1;j<numsSize;j++)

        {

            if(nums[i]+nums[j]==target)

            {

                k[0]=i;

                k[1]=j;

                return k;

            }

        }

    }

    return 0;

}

1. 两数相加 C

struct ListNode\* addTwoNumbers(struct ListNode\* l1, struct ListNode\* l2){

    struct ListNode \*a,\*out;

    int sum = 0,up = 0;

    a = (struct ListNode\*)malloc(sizeof(struct ListNode));

    out = a;

    while(l1!=NULL&&l2!=NULL)

    {

        sum = l1->val + l2->val + up;

        if(sum>=10) {up = 1; sum-=10;} else {up = 0;}

        a->val= sum;

        if(l1->next!=NULL||l2->next!=NULL){

            a->next = (struct ListNode\*)malloc(sizeof(struct ListNode));

            a = a->next;}

        a->next = NULL;

        l1 = l1->next;

        l2 = l2->next;

    }

    if(l1==NULL&&l2!=NULL)

    {

        while(l2!=NULL)

        {

            sum = l2->val + up;

            if(sum>=10) {up = 1; sum-=10;} else {up = 0;}

            a->val= sum;

            if(l2->next!=NULL){

            a->next = (struct ListNode\*)malloc(sizeof(struct ListNode));

            a = a->next;}

            a->next = NULL;

            l2 = l2->next;

        }

    }

    if(l1!=NULL&&l2==NULL)

    {

        while(l1!=NULL)

        {

            sum = l1->val+ up;

            if(sum>=10) {up = 1; sum-=10;} else {up = 0;}

            a->val= sum;

            if(l1->next!=NULL){

            a->next = (struct ListNode\*)malloc(sizeof(struct ListNode));

            a = a->next;}

            a->next = NULL;

            l1 = l1->next;

        }

    }

    if(up==1)

    {

        a->next = (struct ListNode\*)malloc(sizeof(struct ListNode));

        a = a->next;

        a->val = up;

        a->next = NULL;

    }

    return out;

}

1. 无重复字符串 C

int lengthOfLongestSubstring(char \* s){

    int i = 0,j = 0,k = 0,f = 0,g = 0,h = 1,i1 = 1;

    char t;

    while(s[0]!='\0')

    {

        j = 1;

        t = s[0];

        for(i=1;s[i]!='\0';i++)

        {

            if(s[i]!=t)

            {

                j++;

            }

            else

            {

                break;

            }

        }

                h = 1;

                for(f=0;f<j;f++)

                {

                    for(g=f+1;g<j;g++)

                    {

                        if(s[f]==s[g]) h = 0;

                    }

                }

                while(h==0&&j>=0){

                    j--;

                    i1 = 1;

                    for(f=0;f<j;f++)

                {

                    for(g=f+1;g<j;g++)

                    {

                        if(s[f]==s[g]) i1 = 0;

                    }

                }

                    if(i1==1) h = 1;

                }

                if(j>k&&h==1) k = j;

        s++;

    }

    return k;

}

1. 寻找两个正序数组的中位数

double findMedianSortedArrays(int\* nums1, int nums1Size, int\* nums2, int nums2Size){

int newSize = nums1Size + nums2Size;

int i=0,j=0,k=0;

int u[5000];

while(i<nums1Size&&j<nums2Size)

{

if(nums1[i]>=nums2[j])

{

u[k] = nums2[j];

k++;

j++;

}

else

{

u[k] = nums1[i];

k++;

i++;

}

if(k>=newSize/2+1) break;

}

if(i==nums1Size)

{

while(k<newSize/2+1)

{

u[k] = nums2[j];

k++;

j++;

}

}

if(j==nums2Size)

{

while(k<newSize/2+1)

{

u[k] = nums1[i];

k++;

i++;

}

}

if(newSize%2==1)

{

return u[newSize/2];

}

else

{

return (u[newSize/2]+u[newSize/2-1])/2.0;

}

}

1. 最长回文子串

char \* longestPalindrome(char \* s){

int end = strlen(s) - 1;

int position = 0,zhenchangdu = -1,changdu;

int i,j;

int jige,weizhi;

for(position=0;position<=end;position++)

{

i = position - 1;

j = position + 1;

changdu = 0;

while(i>=0&&j<=end)

{

if(s[i]==s[j]) changdu++;

else break;

i--;j++;

}

if(changdu>zhenchangdu)

{

zhenchangdu = changdu;

weizhi = position;

jige = 1;

}

if(position<=end-1&&s[position]==s[position+1])

{

i = position - 1;

j = position + 2;

changdu = 0;

while(i>=0&&j<=end)

{

if(s[i]==s[j]) changdu++;

else break;

i--;j++;

}

if(changdu>=zhenchangdu)

{

zhenchangdu = changdu;

weizhi = position;

jige = 2;

}

}

}

if(jige==1)

{

s[weizhi+zhenchangdu+1] = '\0';

}

else if(jige==2)

{

s[weizhi+zhenchangdu+2] = '\0';

}

return s+weizhi-zhenchangdu;

}

1. Z字形变换

char \* convert(char \* s, int numRows){

char map[1000][1000];

int i = 0,j = 0,k = 0,f = 0;

int i1,j1;

int lie;

char\* out = s;

if(strlen(s)==1||numRows==1) return s;

for(i1=0;i1<numRows;i1++)

{

for(j1=0;j1<1000;j1++)

{

map[i1][j1] = '0';

}

}

i=j=0;

while(\*s!='\0')

{

if(j%(numRows-1)==0&&(i<numRows))

{

if(i==0&&f==1) i++;

map[i][j] = \*(s++);

i++;

}

else

{

if(i==numRows) {i--;lie = j+numRows-1;}

i--;j++;

map[i][j] = \*(s++);

if(f==0) f = 1;

}

}

for(i1=0;i1<numRows;i1++)

{

for(j1=0;j1<=lie;j1++)

{

if(map[i1][j1]!='0') {out[k]=map[i1][j1];k++;}

}

}

return out;

}