中山大學
SUN YAT-SEN UNIVERSITY

数组进阶与练习

中山大学计算机学院

主讲人：潘茂林

目录

CONTENTS

搜索插入位置

题目描述：给定一个严格升序数组和一个目标值，在数组中找到目标值，并返回其索引。如果目标值不存在于数组中，返回它将会被按顺序插入的位置。

示例1:
数组：1, 3, 5, 6
目标值：5
答案：2

示例2:
数组：1, 3, 5, 6
目标值：2
答案：1

示例3:
数组：1, 3, 5, 6
目标值：7
答案：4

题目链接：35. 搜索插入位置 - 力扣（Leetcode）

搜索插入位置

有序数组：像题目给定的数组一样，数组里的元素是按大小关系顺序排列的数组称为有序数组，从小到大排列的称为升序数组，当数组中相邻两个元素是严格小于关系时，称为严格升序数组；类似的可以定义降序数组和严格降序数组。

升序数组：1, 3, 3, 5, 5, 6
严格升序数组：1, 3, 5, 6
降序数组：6, 5, 3, 3, 3, 1
严格降序数组：6, 5, 3, 1

搜索插入位置

解题方法1:

由于数组是严格升序数组，可以从左到右依次扫描数组中的每一个元素，当扫描到的元素大于等于目标值时，该位置就是答案位置，后面的元素不需要再扫描；若扫描完整个数组都没找到满足条件的位置，则输出末尾位置。

代码:

```
int searchInsert_simple(int a[], int n, int target)
{
    for (int i=0; i<n; ++i)
        if (a[i]>=target) return i;
    return n;
}
```

搜索插入位置

解题方法2：

由于数组是严格升序数组，可以使用二分法查找答案位置。

二分法（对边界的不同处理会导致不同的二分写法，下面只介绍其中一种）：

我们可以使用两个指针（L, R）标记查询的数组范围，开始时为$L = -1, R = n$，表示范围$(-1, n]$。每次判断范围中的中间（$mid = (L + R)/2$）元素与目标值的大小关系，若中间元素比目标值小，则$L = mid$，否则$R = mid$。重复操作，直到L, R相邻（即$L + 1 == R$），最终答案为R。

能这样做的原因是数组是严格升序数组，当中间元素比目标值小时，左边的元素一定也比目标值小，因此左边的元素（包括 mid ）肯定不是答案，因此缩小范围到 $(mid, R]$； 同理，当中间元素大于等于目标值时，右边的元素一定大于目标值，因此右边的元素肯定不是答案，因此缩小范围到$(L, mid]$。

搜索插入位置

二分法例子，目标值是8

第一轮，$L=-1, R=8, mid=\frac{-1+8}{2}=3$。$a[mid]=6<8$，所以$L=mid=3$

下标	0	1	2	3	4	5	6	7
元素	1	2	3	6	8	9	10	11
	L			mid				R

第二轮，$L=3, R=8, mid=\frac{3+8}{2}=5$。$a[mid]=9\geq 8$，所以$R=mid=5$

下标	0	1	2	3	4	5	6	7
元素	1	2	3	6	8	9	10	11
				L		mid		R

搜索插入位置

第三轮，$L=3, R=5, mid=\frac{3+5}{2}=4$。$a[mid]=8\geq 8$，所以$R=mid=4$

下标	0	1	2	3	4	5	6	7
元素	1	2	3	6	8	9	10	11
				L	mid	R		

$L=3, R=4$。满足$L+1==R$，结束，答案为$R=4$

代码：

```
int searchInsert_binary_search(int a[], int n, int target){
    int L=-1, R=n, mid;
    while (L+1<R){
        mid=(L+R)>>1;
        if (a[mid]<target) L=mid;
        else R=mid;
    }
    return R;
}
```

搜索插入位置

两个解法的对比：

在最坏情况下，解法1要扫描完整个数组，即要进行n次比较；解法2每次可以缩小一半的范围，因此需要进行大约$\log_2 n$次比较。当 n 较大时解法2在时间上要优于解法1，但使用解法2的前提是有序数组，若是在非有序数组中查找元素，还是需要用解法1。

算法：解决某一问题时，按某种的操作步骤以得到问题答案的过程。二分法就是其中一种，在之后的学习中还会遇到很多算法，例如排序算法。

删除有序数组中的重复项

题目描述：给定一个升序数组，原地删除数组中重复出现的元素，使每个元素只出现一次，即去重后的数组为严格升序数组，并返回去重后数组的长度。

示例1：
数组：0, 0, 1, 1, 1, 2, 2, 3, 3, 4
去重后数组：0, 1, 2, 3, 4
长度：5

题目链接：26. 删除有序数组中的重复项 - 力扣（Leetcode）

删除有序数组中的重复项

解法：

一般想法是新建一个数组，从左到右扫描原数组的元素，如果当前元素与新数组的末尾元素不一样，则把当前元素添加到新数组中，否则不添加。重复操作，直到扫描完所有元素。

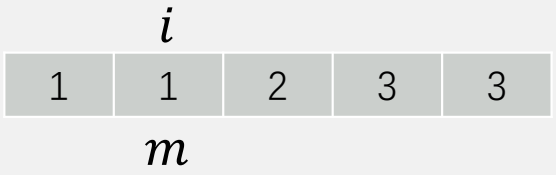
试着不新建数组，将去重后的数组放在原数组的前部。记原数组为a，用变量m表示当前新数组的长度，可知m也是下次添加位置下标，从左到右扫描原数组（下标记为i），如果$a[m-1]!=a[i]$，则$a[m]=a[i], m+=1$，否则不进行操作。由于$m \leq i$始终成立，所以不会出现未扫描的位置被覆盖的情况。

删除有序数组中的重复项

例子：

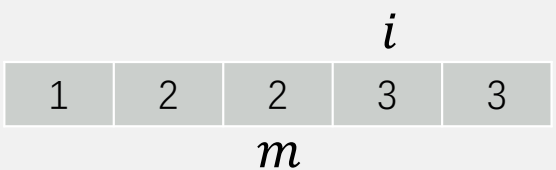
初始：$i=1, m=1$（非空数组第0位一定保留）

$a[m-1] == a[i]$，不保留$a[i]$

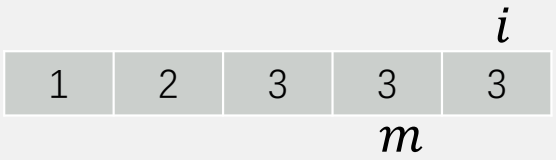


$i=2, m=1, a[m-1] != a[i]$，保留$a[i]$

$i=3, m=2, a[m-1] != a[i]$，保留$a[i]$



$i=4, m=3, a[m-1] == a[i]$，不保留$a[i]$



结束，$m=3$

代码：

```
int removeDuplicates(int a[], int n)
{
    if (n==0) return 0;
    int m=1;
    for (int i=1; i<n; ++i)
        if (a[i]!=a[m-1]) a[m++]=a[i];
    return m;
}
```

两数之和

题目描述：给定一个数组与一个目标值，找到数组中的两个不同位置，满足位置上的元素相加等于目标值。数据保证有唯一解。

示例1:
数组：2, 7, 11, 15
目标值：9
答案（下标）：0, 1

示例2:
数组：3, 2, 4
目标值：6
答案：1, 2

示例3:
数组：3, 3
目标值：6
答案：0, 1

题目链接：1. 两数之和 - 力扣（Leetcode）

两数之和

解法:

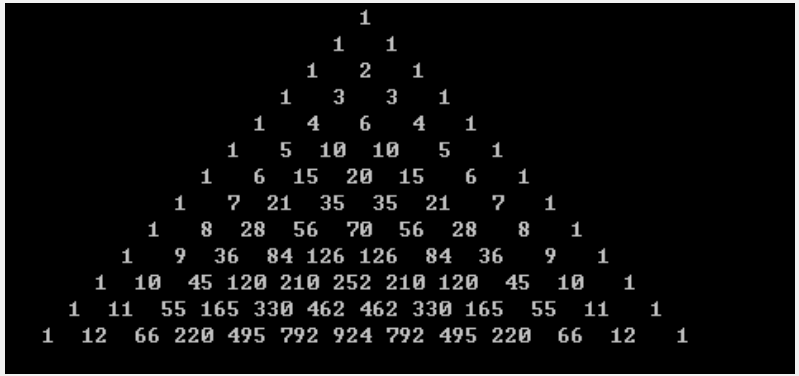
枚举所有的下标组合，判断哪一组对应元素相加等于目标值。

代码:

```
void twoSum(int a[], int n, int target, int idx[])
{
    for (int i=0; i<n-1; ++i)
        for (int j=i+1; j<n; ++j)
            if (a[i]+a[j]==target)
            {
                idx[0]=i;
                idx[1]=j;
                return;
            }
}
```

打印杨辉三角形

- 杨辉三角型如图



  - 输入 n >= 1 && n <=13
  - 解法一，要求使用一个最多 n 个元素的数组
  - 解法二，可以使用二维数组

参考题目链接：119. 杨辉三角 II - 力扣（Leetcode）

C语言数组（Array）与分类

数组是由**连续**无空隙分配的，拥有**特定元素类型的对象**构成的。这些对象的数目数量（**数组大小**）在数组生存期间**决不改变**。

有几种数组类型变体：

- 已知常量大小的数组，例如：double a[3][3]
- **变长度数组（VLA）C99，**例如：double a[n]
- 未知大小数组，例如：double a[]

VLA和未知大小数组只能声明于**块作用域**或**函数原型**作用域中。
函数声明中，数组的第一个维度都被替换为未知大小。例如，int a[3][3] 等价于 int a[][3]。

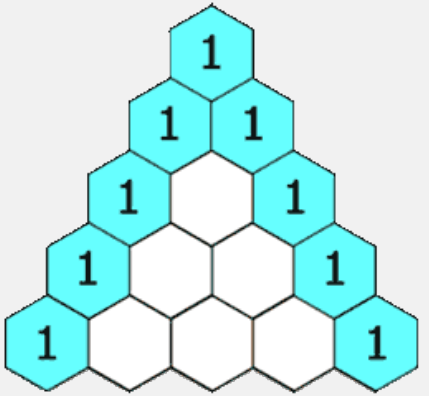
打印杨辉三角形

解法:

先看官方的动画，发现**当前行计算仅依赖上一行数值**。首先考虑从第1行迭代到第n行

代码:

```
void print_yanghui(int n){
    int line[n];
    for (int i=1;i<=n;i++) {
        nextline(line,i);
        for (int j=0;j<i;j++) {
            printf("%-4d",line[j]);
        }
        printf("\n");
    }
}
```



打印杨辉三角形

解法:

首先，第1，2行是基本情况。

其次，算下一行至少有两种方法，第一：申明一个数组计算，然后复制到 prev_line。第二，**利用计算 i 位置值依赖 i 和 i-1 位置数值这个特点**，可以在原数组上计算。

代码:

```
void nextline(int prev_line[], int line_no) {
    if (line_no == 1) {
        prev_line[0] = 1;
    }
    else if (line_no == 2) {
        prev_line[0] = prev_line[1] = 1;
    }
    else {
        int a=prev_line[0];
        int b=prev_line[1];
        for (int i=1;i<line_no-1;i++) {
           prev_line[i] = a + b;
           a = b;
           b = (i==line_no-2)? 0: prev_line[i+1];
        }
        prev_line[line_no-1] = a + b;
    }
}
```

打印杨辉三角形

练习:

首先，完善程序，修改左对齐的三角形，打印出要求的三角形。

其次，改计算下一行的方法为先申请一个数组，再计算下一行的方法。

代码:

案例与知识点总结

- 搜索插入位置
  - P4，**有序数组** - 严格有序数组 - 升序排列 - 降序排列
  - P5，单指针方法。使用 i 顺序扫描
  - P6，二分查找方法。它是双指针方法一种，每次根据中间m位置的值，将$[L,R]$区间，在 $[L,m-1],[m,R]$ 区间中取其一。该方法仅适用有序数组。
- 删除有序数组中的重复项
  - P11，双指针方法。它运用前（$Front$）、后（$Back$）指针特点，F在前面扫描，在B位置添加数据，实现在一个数组上完成重复项删除。
- 两数之和
  - P14，暴力法方法。通过枚举所有组合，虽然简单粗暴，也很优美。**先做到再做好**，做事的基本方法之一。
- 打印杨辉三角形
  - P16 **可变长度数组（VLA）**
  - P18，P9 前后数据依赖关系分析。例如：后一行依赖前一行等

中山大學
SUN YAT-SEN UNIVERSITY

中山大學
SUN YAT-SEN UNIVERSITY

谢 谢

中山大学计算机学院

编制人：课题组