内容提要

本书介绍数据结构和算法相关知识，引入部分 C++ 标准模板库（standard template library，STL）和程序设计竞赛知识，培养学生的计算思维、分析与解决具体问题的能力及创新能力。本书包括绪论、线性表、栈与队列、其他线性结构、树、图、查找、排序 8 章内容，重点介绍的内容主要包括线性表、栈、队列、二叉树和图等数据结构的概念、逻辑结构、存储结构及相应算法，二分查找、二叉排序树和哈希查找等查找算法，插入排序、快速排序、堆排序和二路归并排序等排序算法。

本书力求把数据结构的抽象理论知识和算法实现讲得通俗易懂，以 C++ 语言描述算法，注重问题求解，可作为高等院校计算机类、电子信息类、数据科学与大数据技术及应用统计学等相关专业的教材，也可作为程序设计类竞赛和信息学竞赛的参赛者、数据结构与算法的教师或自学者的参考用书。

图书在版编目（CIP）数据

数据结构与算法 / 黄龙军主编 . —上海：上海交通大学出版社，2022.7

ISBN 978-7-313-26988-1

Ⅰ. ①数… Ⅱ. ①黄… Ⅲ. ①数据结构—高等学校—教材 ②算法分析—高等学校—教材 ③ C++ 语言—数据结构—高等学校—教材 Ⅳ. ① TP311.12 ② TP301.6

中国版本图书馆 CIP 数据核字（2022）第 108164 号

数据结构与算法
SHUJU JIEGOU YU SUANFA

主　　编：黄龙军
出版发行：上海交通大学出版社
邮政编码：200030
印　　制：北京华创印务有限公司
开　　本：889 mm × 1194 mm　1/16
字　　数：393 千字
版　　次：2022 年 7 月第 1 版
书　　号：ISBN 978-7-313-26988-1
定　　价：56.00 元

地　　址：上海市番禺路 951 号
电　　话：6407 1208
经　　销：全国新华书店
印　　张：15.5
印　　次：2022 年 7 月第 1 次印刷

目录

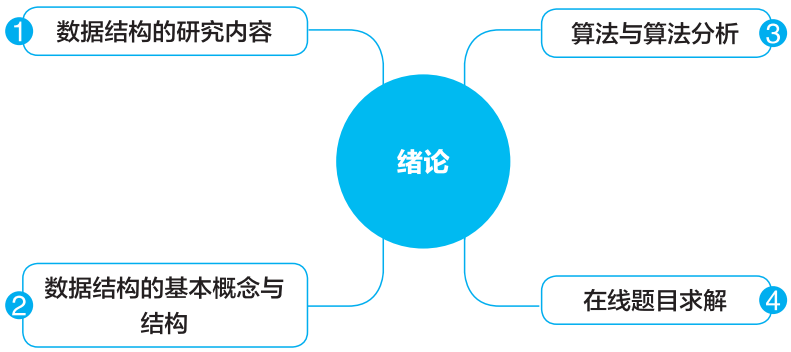
CONTENTS

第 1 章 绪 论

学习目标

1. 记忆和理解数据结构的基本概念和术语。
2. 记忆和理解算法的含义及其特性。
3. 初步学会算法分析的方法。
4. 理解如何更好地提高算法的时间、空间效率。
5. 树立学好专业知识、助力中国梦的信念。
6. 获取仰望星空、探索创新的精神动力。

知识导图



导言

数据结构是介于数学、计算机硬件和计算机软件三者之间的一门核心课程，是诸多高校计算机相关专业考研、考博的常考课程，是程序设计竞赛的必备课程，也是计算机类专业及其他相关专业的必修课程。学好数据结构对于这些专业的学生而言，至关重要。

思政链接：中国梦

通过学习数据结构课程，能够分析研究计算机加工对象的特性，获得其逻辑结构，根据需求选择合适的存储结构及其相应的算法；学习一些常用算法，如查找和排序；进行复杂程序设计的训练；初步掌握算法的时间、空间效率分析技术。

本章介绍数据结构的研究内容、数据结构的基本概念与结构、算法与算法分析等方面的内容，重点关注数据元素、数据项、数据结构、逻辑结构、存储结构、算法、时间复杂度、空间复杂度等概念与结构，算法的特性，算法的时间复杂度和空间复杂度分析方法。最后，通过在线题目的求解，表明有效提升算法性能的必要性。

1.1 数据结构的研究内容

思政链接：沃斯

数据结构主要研究非数值计算问题，重点考虑如何合理地组织数据，如何高效地处理数据。

著名的计算机科学家、图灵奖得主尼古拉斯·沃思（Niklaus Wirth）曾提出一个公式：

$$\text{算法} + \text{数据结构} = \text{程序}$$

其中，算法是处理问题的策略，数据结构是问题的数学模型，程序是为实现特定目标或解决特定问题而用程序设计语言编写的指令序列。

数值计算问题的数学模型是数学方程，而非数值计算问题则无法用数学方程建立数学模型。对于非数值计算程序设计问题，首先要考虑操作对象在计算机中的组织方式。下面举例说明。

例 1.1.1 学生信息管理问题

对于学生信息管理系统中的学生信息，可以采用二维表的形式组织，如表 1-1 所示。

表 1-1　学生信息表

学号	姓名	专业
21145141	张无忌	计算机科学与技术
21539141	令狐冲	网络工程
21145241	郭靖	计算机科学与技术
21536141	杨过	数据科学与大数据

在学生信息表中，除了第一个和最后一个学生外，其他每个学生的前一个位置和后一位置都有且仅有一个学生，是一种“一对一”的线性关系，可视为线性表。本例的数据模型为线性表。

例 1.1.2 对弈问题

在对弈问题中，前一阶段的一种局势可以演变为后一阶段的若干种局势。例如，图 1-1 所示的三子棋局势变化中，上一步的一种局势可能产生下一步的多种局势。可见，对弈问题的局势变化是一种

“一对多”的关系，可用树结构表达。在树结构中，一个父节点可能有多个子节点，而每个节点最多只有一个父节点。本例的数据模型为树结构。

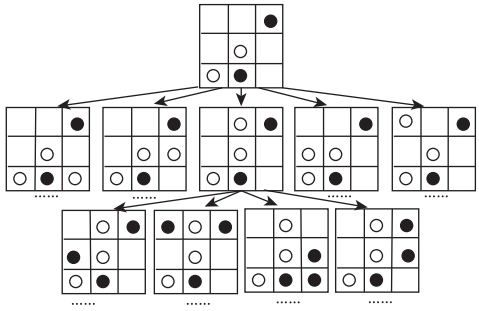


图 1-1　三子棋局势变化示意图

例 1.1.3 最短路径问题

在自驾旅游规划时，若希望从始发城市到目的城市的行车距离尽可能短，则该如何安排驾车路线呢？例如，图 1-2 所示把 5 个城市抽象为 5 个顶点 1、2、3、4、5，任意一个城市都可能与其他多个城市之间有直达道路，即任意两个顶点之间是一种“多对多”的关系，这是一种图结构。设始发城市为 1，目的城市为 2，则从 1 到 2 有 1 → 2、1 → 4 → 2、1 → 5 → 3 → 2、1 → 4 → 3 → 2 等多条路线可走，行车距离分别为 90、80、70、60，因此最佳路线是距离为 60 的 1 → 4 → 3 → 2，这是图结构中的一个典型应用问题——最短路径问题。本例的数据模型为图结构。

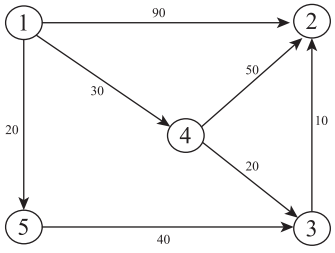


图 1-2　最短路径问题

从以上三个例子可见，非数值计算问题的数学模型是线性表、树结构和图结构等。因此，数据结构主要研究非数值计算程序设计问题中的操作对象以及它们之间的关系和操作。

1.2 数据结构的基本概念与结构

1. 基本概念

数据（data）是客观事物的符号表示，是所有能输入计算机中并能被计算机程序处理的符号的总称。数据可分为数值型数据（如整数、实数等）和非数值型数据（如字符串、图形、图像、声音和动画等）。

数据元素（data element）是数据的基本单位，也称元素或记录，在计算机中通常作为一个整体考虑。例如，表 1-1 中的每条学生记录都是一个数据元素。

数据项（data item）是组成数据元素的、有独立含义的、不可分割的最小单位。例如，表 1-1 中的学生学号、姓名和专业都是数据项。

数据对象（data object）是相同特性的数据元素的集合，是数据的一个子集。

例如，表 1-1 的学生信息表及英文字母子集 S={'A', 'B', 'C', 'D', 'E', 'F'} 都是数据对象。

数据结构（data structure）是相互之间存在一种或多种特定关系的数据元素的集合。或者说，数据结构是带“结构”的数据元素的集合，其中，“结构”是指数据元素之间存在的关系。

数据的逻辑结构、存储结构及运算是数据结构的三要素。

2. 逻辑结构

逻辑结构是指数据元素间抽象化的相互关系，与数据的存储无关，独立于计算机，是从具体问题抽象出来的数学模型。

对于逻辑结构而言，数据结构可归结为线性结构和非线性结构，其中，非线性结构主要包括树结构和图结构，如图 1-3 所示。

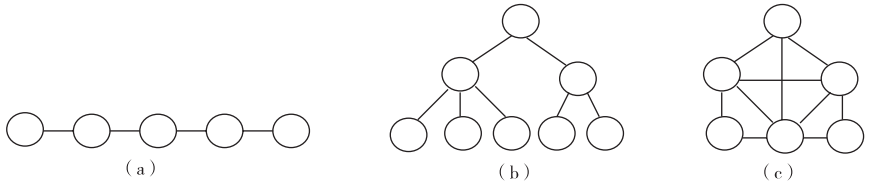


图 1-3　常见的数据结构

(a) 线性结构；(b) 树结构；(c) 图结构

3. 存储结构

存储结构也称物理结构，是数据元素及其关系在计算机存储器中的存储方式。存储结构可分为顺序存储结构和链式存储结构。

顺序存储结构借助元素在存储器中的相对位置来表示数据元素间的逻辑关系，一般采用数组表示，如图 1-4 所示。图中，元素 3 与元素 1 间隔了两个元素位置。

图 1-4　顺序存储结构

链式存储结构借助指示元素存储地址的指针来表示数据元素间的逻辑关系，一般采用链表表示。如图 1-5 所示是带头节点的单链表，头节点的地址设为 5000，存放在头指针 *head* 中；4 个元素相应节点的地址分别假设为 2000、3000、4000、1000。可见，后一个元素对应节点的地址存储在前一个节点的指针域，即前一个节点的指针域指向后一个节点。

图 1-5　链式存储结构

1.3　算法与算法分析

1.3.1 算法基础知识

1. 算法的定义及特性

算法是为解决某类问题而规定的一个有限长的操作序列（步骤）。一个算法必须满足有穷性、确定性、可行性、有输入、有输出 5 个特性。

（1）有穷性是指算法的步骤是有限的，且每个步骤都能在有限时间内完成。

（2）确定性是指算法的每个步骤都有确切规定，不会产生二义性。

（3）可行性是指算法中的每个步骤都是可行的，不会无法实现。

（4）有输入是指一个算法有 0 个或多个输入。因一个算法通常以一个函数来描述，故算法的输入往往以形参表示，在被调用时从主调函数获得输入值。0 个输入的算法通常包含一些初始化语句。

（5）有输出是指一个算法有 1 个或多个输出，表示算法进行信息加工后得到的结果。因一个算法通常以一个函数来描述，故算法的输出经常以函数的返回值或引用类型的形参表示。

2. 算法的评价

设计和评价算法时，通常应考虑达到正确性、可读性、健壮性、高效性等目标。

（1）算法的正确性是指算法能在有限运行时间内得到正确的结果。算法的正确性通常不易证明，可以通过精心设定测试数据的方法来表明算法的正确性。在程序设计竞赛中，经常使用这种方法。

（2）算法的可读性是指算法容易为人阅读和理解。可以通过添加适当的注释语句并合理缩排代码来提升算法的可读性。

（3）算法的健壮性是指算法在遇到非法输入时也能正常结束而不会崩溃。可以通过对非法输入进行特判来提高算法的健壮性。

（4）算法的高效性包括时间高效性和空间高效性，时间高效性是指算法执行效率高，可用时间复杂度衡量；空间高效性是指算法所占存储空间节省，可用空间复杂度衡量。

时间复杂度和空间复杂度是衡量算法优劣的两个主要指标。

1.3.2 算法的时间复杂度分析

1. 基础知识

算法的时间效率度量涉及两个相关术语，即问题规模和语句频度。

问题规模是指算法求解问题输入量的多少，是问题大小的本质表示，一般用整数 n 表示。当然，输入量也可能不仅仅与一个整数相关，这种情况下可用多个整数（如 n、m 等）表示问题规模。

语句频度是指一条语句的重复执行次数。一个算法的执行时间可用该算法中所有语句的频度之和来度量。

简单起见，可用基本语句频度衡量一个算法的时间复杂度。基本语句是指语句频度最大的语句。若计算基本语句的频度得到问题规模 n 的函数 $f(n)$，则算法的（渐近）时间复杂度 $T(n)$ 如式 (1-1) 所示。

$$T(n)=\mathrm{O}(f(n)) \tag{1-1}$$

式 (1-1) 表示随着问题规模 n 的增长，算法执行时间的增长率和函数 $f(n)$ 的增长率相同。

因此，分析一个算法的时间复杂度时，可先找出其中的基本语句，再取其频度函数 $f(n)$ 的数量级（仅需最高次幂项并忽略其系数）用符号 O 表示即可。

常见的时间复杂度依数量级递增（时间效率递减）有常数阶 $\mathrm{O}(1)$、对数阶 $\mathrm{O}(\log_2 n)$、线性阶 $\mathrm{O}(n)$、线性对数阶 $\mathrm{O}(n\log_2 n)$、平方阶 $\mathrm{O}(n^2)$、立方阶 $\mathrm{O}(n^3)$、k 次方阶 $\mathrm{O}(n^k)$、指数阶 $\mathrm{O}(2^n)$ 等。其中，$\mathrm{O}(1)$ 表示与问题规模无关。

2. 时间复杂度分析

例 1.3.1 分析两个 n 阶方阵相乘算法的时间复杂度

两个 n 阶方阵相乘的算法 mult 如下，请分析其时间复杂度。

```
// 以二维数组存储矩阵元素，c 为 a 和 b 的乘积，N、n 分别方阵的最大大小和实际大小
void mult(int a[ ][N], int b[][N], int c[ ][N], int n ) {
    for (int i=1; i<=n; i++){
        for (int j=1; j<=n; j++) {
            c[i][j]=0;
            for (int k=1; k<=n; k++) {
                c[i][j]+=a[i][k]*b[k][j];
            }
        }
    }
}
```

解析：

找出基本语句为“c[i][j] += a[i][k]*b[k][j];”，计算其频度为 n^3，因此算法 mult 的时间复杂度为 $O(n^3)$。

例 1.3.2 分析选择排序的时间复杂度

选择排序算法 select_sort 如下，请分析其时间复杂度。

```
//n 个数据存放在 a 数组中，按升序排序
void select_sort(int a[ ], int n) {
    for (int i=0; i<n-1; ++i) {
        j=i;
        for (int k=i+1; k<n; ++k) {
            if (a[k]<a[j]) j = k;
        }
        swap(a[j], a[i]);
    }
}
```

解析：

找出基本语句为“if (a[k] < a[j]) j = k;”，其频度计算如式 (1-2) 所示。

$$f(n)=\sum_{i=0}^{n-2}(n-1-i)=\frac{1}{2}n^2-\frac{1}{2}n \tag{1-2}$$

取其最高次幂并忽略系数得到 n^2，因此算法 select_sort 的时间复杂度为 $O(n^2)$。

例 1.3.3 分析以下代码段的时间复杂度

有代码段如下，请分析其时间复杂度。

```
int i=1;
while (i<=n) {
    i*=2;
}
```

解析：

找出基本语句为“i*=2;”，先列出循环次数与 i 之间的对应关系，如图 1-6 所示。

循环次数	1	2	3	…	k
i 值	2	4	8	…	2^k

图 1-6　循环次数与 i 之间的对应关系

再令 $2^k=n$，有 $k=\log_2 n$，即基本语句的频度为 $\log_2 n$，因此该代码段的时间复杂度为 $O(\log_2 n)$。

1.3.3 算法的空间复杂度分析

1. 基础知识

算法的（渐近）空间复杂度 $S(n)$ 如式 (1-3) 所示。

$$S(n)=O(g(n)) \tag{1-3}$$

式 (1-3) 表示随着问题规模 n 的增大，算法运行所需存储量的增长率与某个函数 $g(n)$ 的增长率相同。

算法的存储量包括输入数据所占空间、程序本身所占空间和辅助存储空间，因前两者相对固定，算法的空间复杂度分析通常仅需考虑辅助存储空间即可。

若算法所需的辅助存储空间相对于输入数据量来说是常数，则称该算法为原地工作，相应的空间复杂度表示为 O(1)。

常见空间复杂度有 O(1)、$O(\log_2 n)$、$O(n)$ 等。

2. 空间复杂度分析

例 1.3.4 分析逆置数组元素的空间复杂度

有代码如下，分析算法 reverse1 和 reverse2 的空间复杂度。

```
// 算法 reverse1
void reverse1(int a[ ], int n) {
    int *b=new int [n];
    for (int i=n-1; i>=0; i--) {
        b[n-1-i]=a[i];
    }
    for (int i=0; i<n; i++) {
        a[i]=b[i];
    }
}
// 算法 reverse2
void reverse2(int a[ ], int n) {
    for (int i=0;  i<n/2; i++) {
        int t=a[i];
        a[i]=a[n-1-i];
        a[n-1-i]=t;
     }
}
```

解析：

算法 reverse1 使用一个长度为 n 的辅助数组 b，因此空间复杂度为 $O(n)$。

算法 reverse2 仅使用一个辅助变量 t，因此空间复杂度为 O(1)。

1.4 在线题目求解

例 1.4.1 最大子列和问题 ①

给定 K 个整数组成的序列 $\{N_1, N_2, \cdots, N_K\}$，“连续子列”被定义为 $\{N_i, N_{i+1}, \cdots, N_j\}$，其中 $1 \leqslant i \leqslant j \leqslant K$。“最大子列和”则被定义为所有连续子列元素的和中最大者。例如，给定序列 {–2, 11, –4, 13, –5, –2}，其连续子列 { 11, –4, 13 } 有最大的和 20。现要求编写程序，计算给定整数序列的最大子列和。

输入：

输入第 1 行给出正整数 K ($\leqslant$ 100 000)；第 2 行给出 K 个整数，其间以空格分隔。

输出：

在一行中输出最大子列和。如果序列中所有整数皆为负数，则输出 0。

输入样例：

```
6
-2 11 -4 13 -5 -2
```

输出样例：

```
20
```

解析：

最大子列和也称为最大子段和或最大子序列和，暴力求解方法是穷举所有以 i 为起始位置，以 j 为终止位置的子序列，并求得其中的最大和值。

算法 maxSum1 采用三重循环求最大子段和，以 O(n) 的时间复杂度求一个子序列和，总的时间复杂度为 O(n^3)。具体代码如下：

```
int maxSum1(int a[], int n) {                    // 三重循环，时间复杂度为立方阶
    int max=0;                                   // 最大和
    for(int i=0; i<n; i++) {
        for(int j=i; j<n; j++) {
            int sum=0;                           // a[i]~a[j] 的和
            for(int k=i; k<=j; k++) {            // O(n) 时间求一个子序列和
                sum+=a[k];
            }
            if (sum>max) {
                max=sum;
            }
        }
    }
    return max;
}
```

若采用算法 maxSum1 求解本例，则在 PTA 提交代码将得到超时反馈。因此，需考虑改进算法，

① 资料来源：PTA (https://pintia.cn/)，作者：浙江大学 DS 课程组。

提高时间效率。

考虑到若已经求得下标范围为 $[i, j-1]$ 的和为 *sum*，则下标范围为 $[i, j]$ 的和等于 *sum* 加上下标为 *j* 的元素值，则可以 O(1) 的时间复杂度求一个子序列和，如此可用二重循环求最大子段和，如算法 maxSum2 所示，总的时间复杂度为 $O(n^2)$。具体代码如下：

```
int maxSum2(int a[], int n) {                    // 二重循环，时间复杂度为平方阶
    int max=0;                                   // 最大和
    for(int i=0; i<n; i++) {
        int sum=0;                               // a[i]~a[j]的和
        for(int j=i; j<n; j++) {
            sum+=a[j];                           // O(1)时间求一个子序列和
            if (sum>max) {
                max=sum;
            }
        }
    }
    return max;
}
```

实际上，最大子段和的时间复杂度可降为 $O(n)$，算法如 maxSum3 所示。可以这样考虑：若一个子序列的和为负，则该子序列不可能是最大连续子序列的一部分，因为可以通过不包含它来得到一个更大和的连续子序列，所以可在一重循环中求子段和，一旦当前和为负值，则令当前和为 0；否则比较当前和与当前最大和，使当前最大和为两者中的大者。具体代码如下：

```
int maxSum3(int a[], int n) {                    // 一重循环，时间复杂度为线性阶
    int sum=0, max=0;                            // sum是当前和，max是当前最大和
    for(int i=0; i<n; i++) {
        sum+=a[i];
        if (sum<0) {
            sum=0;
        }
        else if (sum>max) {                      // 当前和大于当前最大和
            max=sum;
        }
    }
    return max;
}
```

可以调用以上算法尝试求解本例。

```
#include<bits/stdc++.h>
using namespace std;

int maxSum1(int a[], int n);          // 立方阶的算法
int maxSum2(int a[], int n);          // 平方阶的算法
int maxSum3(int a[], int n);          // 线性阶的算法
void run() {
    int n;
```

```
    cin>>n;
    int *a=new int [n];                    // 申请动态数组
    for(int i=0; i<n; i++) {
        cin>>a[i];
    }
    int result=maxSum1(a, n);              // 调用 maxSum1 进行测试
    //int result=maxSum2(a, n);            // 调用 maxSum2 进行测试
    //int result=maxSum3(a, n);            // 调用 maxSum3 进行测试
    cout << result << endl;
    delete [] a;
}

int main() {
    run();
    return 0;
}
```

对于同一个题目，不同求解方法的效率可能差别很大。如何有效提高算法的时间效率和空间效率是学习数据结构需要重点关注的问题。

习题

一、选择题

1. 数据结构的三要素指的是（　　）。

A. 逻辑结构、存储结构及数据的运算　　B. 线性结构、树结构及图结构

C. 数据、数据元素及数据项　　D. 数据元素、数据关系和数据操作

2. 以下说法正确的是（　　）。

A. 数据元素是数据的最小单位　　B. 数据项是数据的基本单位

C. 数据结构是带有结构的数据元素的集合　　D. 数据对象是同类型数据项的集合

3. 以下数据结构中，（　　）是非线性结构。

A. 树　　B. 字符串　　C. 队列　　D. 栈

4. 以下不属于算法的特性的是（　　）。

A. 有 0 个或多个输入　　B. 至少有 1 个输出　　C. 正确性　　D. 有穷性

5. 下述程序段的时间复杂度为（　　）。

```
i=1;
while(i<=n){
    i=i*2;
}
```

A. $O(\log_2 n)$　　B. $O(n)$　　C. $O(\sqrt{n})$　　D. $O(n^2)$

6. 简单选择排序算法的时间复杂度是（　　）。

A. $O(n^3)$　　B. $O(n^2)$　　C. $O(n)$　　D. $O(1)$

7. 一维数组的逆置算法中，可以达到的最好空间复杂度是（　　）。

A. $O(1)$　　B. $O(\log_2 n)$　　C. $O(n)$　　D. $O(n^2)$

8. 以下空间复杂度中最好的是（　　）。

A. O(n^2)　　B. O(n)　　C. O($\sqrt{n}$)　　D. O(1)

9. 下述程序段的时间复杂度为（　　）。

```
for (i=0; i<m; i++)
    for (j=0; j<n; j++)
        a[i][j]=0;
```

A. O(n)　　B. O(mn)　　C. O(m^2)　　D. O(n^2)

10. 下述程序段的时间复杂度为（　　）。

```
int m=100, n=200;
while(n>0) {
    if(m>100) m-=10,n--;
    else m++;
}
```

A. O(mn)　　B. O(n)　　C. O(m)　　D. O(1)

11. 某算法按顺序执行程序段 1 和程序段 2，假设程序段 1 的执行次数为 $3n^2$，程序段 2 的执行次数为 $0.02n^3$，则该算法的时间复杂度为（　　）。

A. O(n)　　B. O(n^2)　　C. O(n^3)　　D. O(1)

12. 下述程序段的时间复杂度为（　　）。

```
a=n;
b=0;
while(a>=b*b) b++;
```

A. O($\log_2 n$)　　B. O(n)　　C. O($\sqrt{n}$)　　D. O(n^2)

13. 算法优劣分析的两个主要指标是（　　）。

A. 空间复杂度和时间复杂度　　B. 正确性和简单性

C. 可读性和可维护性　　D. 数据复杂性和程序复杂性

14. 执行下面程序段时，执行 S 语句的次数为（　　）。

```
for(int i=1; i<=n; i++)
    for(int j=1;j<=n; j++)
        S;
```

A. n^2　　B. $n^2/2$　　C. $n(n+1)$　　D. $n(n+1)/2$

15. 下述程序段的时间复杂度为（　　）。

```
for (i=0; i<n-1; i++)
    for (j=0; j<n-1-i; j++)
        t=a[j], a[j]=a[j+1], a[j+1]=t;
```

A. O(1)　　B. O(n)　　C. O(n^2)　　D. O(n^3)

16. 第 15 小题中的程序段的空间复杂度为（　　）。

A. O(1)　　B. O(n)　　C. O(n^2)　　D. O(n^3)

二、编程题

寻找第 k 小的数（HLOJ 9500）

给定若干整数，请设计一个高效的算法，确定第 k 小的数。

输入：

测试数据有多组，处理到文件尾。每组测试数据的第 1 行输入两个整数 n，k（$1 \leqslant k \leqslant n \leqslant 1\,000\,000$）。第 2 行输入 n 个整数，每个数据的取值范围为 0~1 000 000。

输出：

对于每组测试，输出第 k 小的数。

输入样例：

```
9 3
1 2 3 4 5 6 9 8 7
```

输出样例：

```
3
```