第一章 答案

一. 选择题

1. 从逻辑上可以把数据结构分为（ C ）两大类

A. 动态结构、静态结构　B. 顺序结构、链式结构

C. 线性结构、非线性结构 D. 初等结构、构造型结构

2. 下面关于算法的说法正确的是（ D ）

A. 算法的时间复杂度一般与算法的空间复杂度成正比

B. 解决某问题的算法可能有多种，但肯定采用相同的数据结构

C. 算法的可行性是指算法的指令不能有二义性

D. 同一个算法，一般情况下实现语言的级别越高，执行效率越低。

3. 在发生非法操作时，算法能够做出适当处理的特性称为（ B ）

A. 正确性　B. 健壮性　C. 可读性　D. 可移植性

二. 判断题

1. 数据的逻辑结构是指数据的各数据项之间的逻辑关系（ F ）
2. 顺序存储方式的优点是存储密度大，且插入、删除运算效率高（ F ）
3. 数据的逻辑结构说明数据元素之间的次序关系，它依赖于数据的存储结构（ F ）
4. 算法的优劣与描述算法的语言无关，但与所用的计算机性能有关（ F ）
5. 算法必须有输出，但可以没有输入（ T ）

三. 简答题

1. 分析程序段中带“#”语句的执行频度，并给出程序段的时间复杂度
   a. 第一小题： $n-1$，$O(n)$

C

```c
j=1; k=0;
while(j<=n-1){
    j++;
    k+=j; //#
}
```

b. 第二小题： $n^2/4, O(n^2)$

频次 = (n-2*1+1)+(n-2*2+1)+(n-2*3+1)+....+(n-2*n/2+1)

= n*n/2-(1+n/2)*n/2+n/2

C

```
1  设n是偶数
2  for(i=1, s=0; i<=n; i++){
3      for(j=2*i; j<=n; j++){
4          s++; //#
5      }
6  }
```

c. 第三小题 $n*(n+1)/2$， $O(n^2)$

频次 = n+(n-1)+(n-2)+....+1

C

```
1  k=0;
2  for(i=0; i<n; i++){
3      for(j=i; j<n; j++){
4          k++; //#
5      }
6  }
```

d. 第四小题 $n*log_2 n$， $O(n*log_2 n)$

C

```
1  k=0;
2  for(i=0; i<n; i++){
3      for(j=1; j<n; j<<=1){
4          k++; //#
5      }
6  }
```

2. 若 $f(n)=O(n^2)$ 且 $g(n)=O(n)$,则以下结论是否正确，并说明理由：
   a. $f(n)+g(n)=O(n^2)$ 正确
   b. $f(n)/g(n)=O(n)$ ~~作废~~
   c. $f(n)*g(n)=O(n^3)$ 正确
3. 有的情况下，算法中基本操作重复执行的次数还随问题的输入数据集不同而不同。假定对包含1、2、3、4、5共5个元素的序列（即n=5）做冒泡排序，请举例说明何时会出现以下情况：

a. 任何元素都无需移动；1、2、3、4、5

b. 某元素会一度朝着远离期最终位置的方向移动；5，4，1，2，3

c. 某元素的初始时已位于最终位置，确需要参与n-1次交换；4，5，3，1，2

d. 所有元素都需要参与n-1次交换。5，4，3，2，1

四. 算法题

《九章算术》记载的“中华更相减损术”可快速计算正整数a和b的最大公约数，其过程如下：

Plain Text

```
令p = 1
若a和b不都是偶数，则转到第5行
令p = px2，a = a/2，b = b/2
转到第2行
令t = |a - b|
若t = 0，则返回并输出 axp
若t为奇数，则转到第10行
令t = t/2
转到第7行
若a ⩾ b，则令a = t；否则，令 b=t
转到第5行
```

1. 按照上述流程，编写一个算法 int gcd(int a, int b)，计算a和b的最大公约数；

C

```c
int gcd(int a, int b){
    int r = 0; //a和b的2^r形式的公因子
    while(!((a&1)||(b&1))){ //若a和b都是偶数
        a >>= 1; b >>= 1; r++; //同时除2，并累加至r
    )
    //以下，a和b至多有一个为偶
    while(1){
        while(!(a&1)) a >> =1;若a偶(b奇)，则剔除a的所有因子2
        while(!(b&1)) b >> =1;若b偶(a奇)，则剔除b的所有因子2
        (a > b) ? a = a-b: b =b-a;
        if(0 == a) return b <<r;
        if(0 == b) return a <<r;
    }
}
```

2. 分析该算法的渐进时间复杂度

$O(log_2(a+b))$