哈尔滨工业大学（深圳）2023 年春《数据结构》
第一次作业 线性结构

学号	220110515	姓名	金正达	成绩	

说明：部分题目来自国外高校，保留其原文。

1. 简答题

1-1 Big-O

For each of the functions *f(N)* given below， indicate the tightest bound possible (in other words，giving $O(2^N)$ as the answer to every question is not likely to result in many points). Unless otherwise specified， all logs are base 2. You MUST choose your answer from the following (not given in any particular order)， each of which could be re-used (could be the answer for more than one of a) – h)):

$O(N^2)$ $O(N^{½})$ $O(N^3 \log N)$ $O(N \log N)$ $O(N)$ $O(N^2 \log N)$ $O(N^5)$
$O(2^N)$ $O(N^3)$ $O(\log N)$ $O(1)$ $O(N^4)$ $O(N^{12})$ $O(N^N)$
$O(N^6)$ $O(N^8)$ $O(N^9)$ $O(N^{10})$

You do not need to explain your answer.

Functions	T(n)
a) $f(N) = (1/2)\ (N \log N) + (\log N)^2$	$O(N \log N)$
b) $f(N) = N^2 \cdot (N + N \log N + 1000)$	$O(N^3 \log N)$
c) $f(N) = N^2 \log N + 2^N$	$O(2^N)$
d) $f(N) = (\ (1/2)\ (3N + 5 + N)\ )^4$	$O(N^4)$
e) $f(N) = (2N + 5 + N^4\ ) / N$	$O(N^3)$
f) $f(N) = \log_{10}(2^N)$	$O(N)$
g) $f(N) = N! + 2^N$	$O(N^N)$
h) $f(N) = (N \cdot N \cdot N \cdot N + 2N)^2$	$O(N^8)$

1.2 Big-O and Run Time Analysis

Describe the worst case running time of the following pseudocode functions in Big-O notation in terms of the variable n. *Showing your work is not required* (although showing work *may* allow some partial credit in the case your answer is wrong – don't spend a lot of time showing your work.). You MUST choose your answer from the following (not given in any particular order)， each of which could be re-used (could be the answer for more than one of I. – IV.):

$O(n^2)$ $O(n^3 \log n)$ $O(n \log n)$ $O(n)$ $O(n^2 \log n)$ $O(n^5)$
$O(2^n)$ $O(n^3)$ $O(\log n)$ $O(1)$ $O(n^4)$ $O(n \cdot n)$

	functions	T(n)

I	void silly(int n) {
	for (int i = 0; i < n; ++i) {
	j = n;
	while (j > 0) {
	System.out.println("j = " + j);
	j = j - 2;
	}
	}
	}
		$O(n^2)$

II	void silly(int n， int x， int y) {
	for (int k = n; k > 0; k--){
	if (x < y + n) {
	for (int i = 0; i < n; ++i){
	for (int j = 0; j < i; ++j){
	System.out.println("y = " + y);
	}
	}
	} else {
	System.out.println("x = " + x);
	}
	}
	}
		$O(n^3)$

III	void silly(int n) {
	for (int i = 0; i < n; ++i) {
	for (int j = 0; j < n; ++j){
	System.out.println("j = " + j);
	}
	for (int k = 0; k < i; ++k) {
	System.out.println("k = " + k)
	for (int m = 0; m < 100; ++m){
	System.out.println("m = " + m);
	}
	}
	}
	}
		$O(n^3)$

IV	int silly(int n， int m) {
	if (m < 2) return m;
	if (n < 1) return n;
	else if (n < 10)
	return silly(n/m， m);
	else
	return silly(n - 1， m);
	}
		O(n)

1.3 对链表设置头结点的作用是什么?简述线性链表头指针，头结点，首元结点(第一个结点) 三个概念的区别；

答：头节点作用：便于对空链表进行操作；在首项插入时与对其他节点的操作保持一致；

头指针：用于存放线性链表中的第一个节点的存储地址，是单链表的标识；

头节点：线性链表的第一个元素节点前面的一个附加节点；
首元节点：线性链表的第一个存储信息节点，地址储存在头指针中。

1.4 在什么情况下用顺序表比链表好，什么时候用链表比顺序表好？请给出你的分析和理解;

答：顺序表存储空间连续，存储密度高，但是存储容量固定；链表存储密度低，存储容量可变，故在对线性表的存储规模难以估计时，选用链表；
顺序表中访问元素的时间复杂度为 O(1)，链表为 O(n)。若经常按序号读取元素，选用顺序表；
顺序表插入删除操作较链表来说更加复杂，若频繁进行插入删除操作，选用链表。

1.5 若频繁地对一个线性表进行插入和删除操作，则该线性表宜采用何种存储结构，为什么?

答：采用链表。
顺序表在插入和删除时，平均要移动表中的一半元素，当数据量较大时，操作复杂且耗时，而链表的插入删除操作主要是比较，相比而言更加便捷。

1.6 队列可以用单循环链表来实现，故可以只设一个头指针或只设一个尾指针，请分析用哪种方案最合适?

答：采用尾指针。
设有尾指针的单循环链表可以方便地进行尾部插入元素，只需修改尾指针和新节点指针的指向即可，在头部插入时，由于该链表为单循环链表，所以可以通过尾指针快速找到链表的第一个元素，方便进行插入操作。

1.7 How to implement a queue using stack?

答：使用两个栈来实现，一个入队栈，一个出队栈，入队时将元素压入入队栈中，出队时，先将入队栈中的元素弹出并压入出队栈之中，全部压入出队栈之后再弹出。

1.8 设数组 A[50][80]，其基地址为 2000，每个元素占 2 个存储单元，以行序为主序顺序存储，回答下列问题：
(1) 该数据组有多少个元素？
(2) 该数组占用多少存储单元？
(3) 数组元素 A[30][30]的存储地址是多少?

答：（1）4000（2）8000（3）6860

2．算法设计

针对本部分的每一道题，要求：
(1) 给出算法的基本设计思想；
(2) 采用类 C 或类 C++语言描述算法，关键之处给出注释；
(3) 分析算法的时间复杂度和空间复杂度。

2.1 输入一个已经按升序排序过的数组和一个数字，在数组中查找两个数，使得它们的和正好是输入的那个数字。

答：设置首尾两个索引，从两边向中间靠拢，就索引位置数组元素和与目标和的

大小来增加 i 或者减少 j；

```c
//nums[]数组为已经升序排列过的数组，大小为N
//answers[]数组大小为2，用于存放结果
//target为目标和
//若未找到这两个数，则返回-1，否则返回0
int twoSum(int[] nums, int target, int answers[]) {
    int i = 0;
    int j = N -1;
    while(i < j) {
        int num = nums[i] + nums[j];
        if (num == target) {
            answers[0] = nums[i];
            answers[1] = nums[j];
            return 0;
        } else if (num > taget) {
            j -= 1;
        } else if (num < target) {
            i += 1;
        }
    }
    return -1
}
```

最坏的情况为 i 自增运算次数和 j 自减运算次数和为 n，故时间复杂度为 O(n);
空间复杂度为 O(1).

2.2 设计一个算法，利用栈(链栈)来实现带头结点的单链表 h 的逆序。
答：按链表顺序入栈后，越在前面的元素越接近栈底，从而出栈时顺序和原链表顺序相反；

```
void reverse(LinkList* H, Stack S) {
    //按单链表顺序将其中元素入栈
    for (int i = 0; i < ListLength(H); i ++) {
        push(&S, GetElem(H, i));
    }
    LinkList tempL;
    InitList(&temrL);
    //出栈并将元素加入新链表，即获得反向的链表
    for (int i = 0; i < ListLength(H); i ++) {
        InsertList(&tempL, pop(&S), i);
    }
    H = &temL;
}
```

时间复杂度 T(n) = 2n，故为 O(n)；
空间复杂度为 O(1)

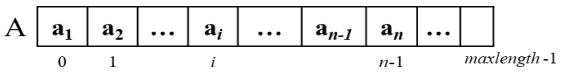
2.3 分别用顺序表、单项链表作为线性表的存储结构，元素类型为整型（int）。设

计算法，将线性表中所有的负数放在正数之前，亦可理解为线性表的左端元素值均小于 0，而右端元素值均大于或等于 0。

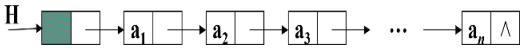
要求算法的空间复杂度 S(n)=O(1);

已知数据类型定义如下：

```
# define maxlength 100
Typedef struct {
        int data[maxlength];
        int last;
} SLIST;
SLIST A;
```




```
Typedef struct NodeType {
            int data ;
    NodeType *next ;
 }LLIST ;
LLIST H;
```



答：顺序表中使用首尾两个索引，自两边向中间靠拢，异号就进行交换；

单向链表中设置两个暂时的头节点，遍历单向链表，将负数插入一个头节点之后，将非负数插入另一个之后，最后将一个的尾节点指向另一个的首节点；

```c
//顺序表实现
void SortSlist(SLIST* A) {
    int i = 0;
    int j = maxlength - 1;
    while (i < j) {
        if (A.data[i] < 0) {
            i += 1;
        }
        if (A.data[j] > 0) {
            j += 1;
        }
        if (A.data[i] > 0 && A.data[j] < 0) {
            int temp = A.data[i];
            A.data[i] = A.data[j];
            A.data[j] = temp;
        }
    }
}

//单项链表实现
void SortLlist(LLIST* H) {
    LLIST A;
    LLIST B;
    for (LLIST* p = H; p->next != null; p = p->next) {
        if (p->data < 0) {
            A.next = p;
        } else {
            B.next = p;
        }
    }
    LLIST* last = H->next;
    while(last->next != null) {
        last = last->next;
    }
    last->next = B.next;
    H = &A;
}
```

顺序表实现中最坏的情况为 i 自增运算次数和 j 自减运算次数和为 n，故时间复杂度为 O(n);

单项链表实现中，T(n) = 2n，故时间复杂度为 O(n);

2.4 已知一个带有表头结点的单链表，结点结构为（data，next），假设该链表只给出了头指针 list。在不改变链表的前提下，请设计一个尽可能高效的算法，查找链表中倒数第 *k* 个位置上的结点（*k* 为正整数）。若查找成功，算法输出该结点的 data 域的值，并返回 1；否则，只返回 0。

答：先计算出单链表的长度，再将倒数转化为正数，然后从单链表开头开始遍历即可。

```c
int search(LinkList L, int k) {
    int length = 0;
    for (LinkList* p = &L; p->next != null; p = p-> next) {
        length += 1;
    }
    int count = 0;
    int n = length + 1 - k;
    //若k越界，则返回0
    if (n < 0) {
        return 0;
    }
    for (LinkList* p = &L; p->next != null; p = p-> next) {
        if (count == n) {
            printf("%d", p->data);
            return 1;
        }
        count += 1;
    }
}
```

T(n) = 2n，故时间复杂度为 O(n)，空间复杂度为 O(1);

2.5 设二维数组 A[1..m][1..n]含有 mxn 个整数，设计一个算法，判断 A 中所有元素是否互不相同，输出相关信息（yes/no）。

答：使用一个数组来记录 A 中每个元素是否出现，若第一次出现，则将对应值置为 1，若再次出现，则返回结果。

```c
//存在相同元素返回1，否则返回0
int IsDuplicate(int A[m][n], int m, int n) {
    int map[100000] = {0};
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            int key = A[i][j];
            if (map[key] == 1) {
                return 1;
            }else if (map[key] == 0) {
                map[key] = 1;
            }
        }
    }
    return 0;
}
```

```

只需遍历一次该二维数组，故时间复杂度为 O(m*n)，空间复杂度为 O(1).
```