

考试科目: 数据结构与算法 B

姓名: _____ 学号: _____

考试时间: 2024 年 4 月 9 日

任课教师: _____

注意: 1、请把解答写在答题纸上, 试卷上的解答无效! 2、背面可作稿纸, 试卷考试后收回。

一、(42 分) 单项选择题 (每小题 2 分)

1. 在河内塔问题中, 假设盘片的个数为 n , 则搬完盘片的时间复杂度为_____。
A: $O(n)$ B: $O(n^2)$ C: $O(2^n)$ D: $O(\sqrt{n})$
2. 设有三个元素 X, Y, Z 顺序进栈 (进的过程中允许出栈), 下列得不到的出栈排列是_____。
A: XYZ B: YZX C: ZXY D: ZYX

3. 判定一个无序表 Q (链表实现) 为空的条件是_____。
A: $Q.head == None$ B: $Q == None$ C: $Q.head == 0$ D: $Q.head != None$

点评: 使用 `head` 指向第一个结点; 判断“为空”。

4. `l=list(range(10**6))`, 下面那条语句执行时间最短?
A: `l.reverse()` B: `reversed(l)` C: `sorted(l)` D: `l.sort()`

点评: `reversed()` 只会返回一个 `iterator`, 没有进行实质运算。

5. `l=list(range(10**6))`, `l+=[10086]` 恰好比下面哪条语句慢?
A: `l=l+[10086]` B: `l.append(10086)` C: `l.insert(10086,0)` D: `l.insert(0,10086)`
6. 允许表达式内多种括号混合嵌套, 检查表达式中括号是否正确配对的算法, 通常选用_____。
A: 栈 B: 线性表 C: 队列 D: 带头结点的单链表

点评: 闭括号需要与前面最后出现的开括号匹配, 需要栈的“后进先出”特性。

7. 在稳定匹配算法中, 若改为由女生先邀请男生, 则下面说法正确的是:
A: 算法结果有利于女生群体
B: 女生对邀请的舞伴满意度是递增的
C: 匹配的结果稳定说明男女双方都找到了最满意的舞伴
D: 算法最多迭代 $2n$ 轮

8. 线性表的顺序存储结构是一种_____。
A: 随机存取的存储结构
B: 顺序存取的存储结构
C: 索引存取的存储结构
D: Hash 存取的存储结构

点评: ♣ 通过计算出来的地址, 可以对顺序存储在一片连续空间上的各个元素进行随机访问。

9. 若某线性结构采取顺序表存储, 则它的存储空间地址_____, 如果采取链表存储, 则它的存储空间地址_____。
- A: 一定不连续; 一定连续
B: 既可连续亦可不连续; 一定不连续
C: 一定连续; 既可连续亦可不连续
D: 一定连续; 与头结点存储地址保持连续

点评: 链表结点可以放在一次性分配的大块连续空间上。

10. 单链表中, 增加一个头结点的目的是_____。
- A: 使单链表至少有一个结点
B: 标识表结点中首结点的位置
C: 方便运算的实现

点评: ♣ 做一件事情客观上产生很多效果, 但并不都是我们的目的。

11. 栈和队列的主要区别是_____。
- A: 逻辑结构不同
B: 存储结构不同
C: 所包含的运算个数不同
D: 限定插入和删除的位置不同

点评: 都属于线性表, 栈只能在一端插入/删除; 而队列在一端插入, 另一端删除。

12. 为环形队列内部用一个大小为 11 的列表容纳队列元素。完整的数据操作 Z 需要先往队列中插入两个元素, 再取出一个元素。从空队列开始, 能够连续成功完成几次数据操作 Z?
- A: 9 B: 10 C: 11 D: 12

点评: 环形队列的容量比它真实的空间数量少一, 浪费一个空间来区分满表与空表。

13. 设线性表有 $2n$ 个元素。以下操作中在单链表上实现比在顺序表上实现效率更高的是_____。
- A: 在最后一个元素后面插入一个新元素
B: 顺序输出前 k 个元素
C: 交换第 i 个元素和第 $2n-i-1$ 个元素的值 ($i=0, 1, \dots, n-1$)
D: 交换第 i 个元素和第 $n+i$ 个元素的值 ($i=0, 1, \dots, n-1$)
14. 在线性表中, 除开始元素外, 每个元素_____。
- A: 只有唯一的前驱元素
B: 只有唯一的后继元素
C: 有多个前驱元素
D: 有多个后继元素

点评: 除了开始元素, 都有唯一前驱。

15. 下面的递推公式, 符合哪道上机作业题的解题思路?

$$\begin{cases} Profit(i) = \max(Profit(i-1), Price(i) - lowest(i-1)) \\ lowest(i) = \min(lowest(i-1), Price(i)) \end{cases}$$

A: 机智的股民老张 B: 逐词倒放 C: 多项式加法 D: 放苹果 (盘子相同)

16. 若线性表最常用的操作是存取第 i 个元素, 及其前驱和后继元素的值, 为了提高效率, 应采用_____的存储方式。

A: 单链表 B: 双向链表 C: 单循环链表 D: 顺序表

点评: ♣ 通过计算出来的地址, 可以对顺序存储在一片连续空间上的各个元素进行随机访问。

17. 如果线性表中最常用的操作是在最后一个元素之后插入新元素和删除第一个元素, 则采用_____存储方式最节省时间。

A: 单链表
B: 仅有头指针的单循环链表
C: 双链表
D: 仅有尾指针的单循环链表

18. 已知字符串 S 为 “abaabaabacacaabaabcc”, 模式串 t 为 “abaabc”。采用 KMP 算法进行查找, 第一次出现不匹配 ($s[i] \neq t[j]$) 时, $i=j=5$, 则下次比对时, i 和 j 的值分别是_____。

A: $i=1, j=0$ B: $i=5, j=0$ C: $i=5, j=2$ D: $i=6, j=2$

点评: ♣ KMP 算法的一个重要特点就是字符串指针从不回滚。当出现不匹配时, 查出前面匹配部份 $\xi = \text{“abaab”}$ 的最长公共前后缀长度, 这里是 2, 赋值给模式串指针。所以这里的答案是 $i=5, j=2$ 。

提问: 如果 ξ 为空串, 对应什么情况, 该怎么办?

19. 都有 n 个不同元素的两个表, 一个递增有序, 一个无序。都采用顺序查找算法, 对有序表查找进行改进: 若发现当前元素大于待查元素, 提前返回。若查找任一元素的概率相同, 请问成功查找的平均时间_____。

A: 后者小 B: 相同 C: 前者小 D: 不好比较

点评: 题干中的陷阱 “成功查找”, 排除掉了提前返回的情况。

20. 有一个长度为 11 的有序表, 按二分查找法对该表进行查找, 在表内各元素等概率的情况下, 查找成功所需的平均元素比较次数为_____。

A: $35/11$ B: $34/11$ C: $33/11$ D: $32/11$

点评: ♣ 又是 “成功查找” 的坑。从 1 开始, 序号为 6 的元素, 只需要一次比较; 序号为 3 和 8 的元素, 需要两次比较; ...

21. 在以下四项计算中, 哪一项与其他三项结果不同?

A: 已知 n 个元素的人栈序列, 求可能出栈序列的数目。
B: n 对括号, 求合法的左右括号匹配的序列数目。
C: n 个结点的二叉树¹有多少种。
D: n 阶台阶, 小明每次可以选择走一级台阶或者走两级台阶, 求总共有多少种到达台阶顶端的走法。

点评: 卡特兰数:

$$Q_{n+1} = \sum_{i=0}^n Q_i Q_{n-i}, \text{ if } n \geq 0$$

$$Q_0 = 1$$

¹ 二叉树: 除根结点外每个结点有惟一父结点; 最多有两个子结点且区分左右。

斐波那契数:

I

$$Q_n = Q_{n-1} + Q_{n-2}$$

$$Q_0 = Q_1 = 1$$

二、(13 分) 判断题 (每小题 1 分; 对填写 “Y”, 错填写 “N”).

1. N python 中集合的元素和字典的键值一样, 只能是不可变类型。

点评: 定义了 `__hash__` 方法的自定义类型变量就可以做集合的元素和字典的键值。

2. Y 考虑一个长度为 n 的顺序表中各个位置插入新元素的概率是相同的, 则顺序表的插入算法平均时间复杂度为 $O(n)$ 。

点评: 无需查找, 只管插入。
顺序表插入新元素之后需要移动新位置后面所有元素。

平均移动次数为 $n/2$, 大 O 表示法为 $O(n)$ 。

3. N 线性表中每个元素都有一个直接前驱和一个直接后继。

点评: 大意的, 没有考虑到首尾元素。

4. N 顺序存储方式插入和删除时效率太低, 因此它不如链接存储方式好。

点评: 没有条件的 “好” 或者 “坏”, 一般都是错的。

5. Y 同时实现栈和队列, 单链表效率比顺序表高。

点评: 用顺序表实现队列, 出队和入队操作总有一个是 $O(N)$ 的。

6. N 由于 Python 的变量只是对象的引用, 在下面的语句之后可以通过 `a` 来修改 `l[0]` 的值。

```
l = [1, 2, 3, 4]; a = l[0]
```

点评: 这种情况下, 只有当 `a` 指向的是可变对象, 才会修改它和 `l[0]` 共同指向对象的值; 否则对 `a` 的赋值, 会生成新的对象, 把引用交给 `a`, 原来 `a` 指向的不可变对象不受影响。

7. Y 向栈顺序地输入一个整数序列 1, 2, 3, 4, 5, 6, 可以得到输出序列 3, 2, 5, 6, 4, 1。

点评: 在纸上画一下, 送分。

8. Y 抽象数据类型 ADT 只定义数据对象的行为和功能, 跟具体的程序设计语言无关。

点评: 概念, 送分。

9. Y 一个队列的入队顺序是 1、2、3、4, 则出队的输出顺序只能是 1、2、3、4。

点评: 队列和栈不一样, 出队序列是唯一的。

10. N 队列在程序调用时必不可少，因此递归离不开队列。

点评：记错了吧，程序调用时用到的是“栈”。

11. N 队列的入队和出队操作时间复杂度，总有一个为 $O(1)$ ，另一个为 $O(n)$ 。

点评：课上的一种实现符合这种情况：用一个顺序表，且队列容量不限。
但是还有其他实现：链表、环形队列、双栈等

12. Y 数据元素可以由类型互不相同的数据项构成。

点评：概念，送分。

13. Y 考察某个具体问题是否适合应用动态规划算法，必须判定它是否具有最优子结构性质。

点评：最优子结构就是能够把问题规模缩小的推导关系（推导式）。

三、（24 分） 填空（每题 2 分）

1. 计算机操作系统不能称之为算法，首先它不满足算法的基本特性 有穷性。

点评：^I 概念送分题。

2. 数据的逻辑结构可粗分为 线性结构 和 非线性结构。

点评：概念送分题。

3. 算法原地工作说的是 算法所需的辅助空间是常量或 $O(1)$ 。

点评：概念送分题。

4. 线性表的顺序存储与链式存储是两种常见存储形式；当表元素有序排序进行二分检索时，应采用 顺序表 存储形式。

点评：二分检索需要通过下标随机访问线性表元素，所以只能用顺序表。

5. 设环形队列的容量为 20（单元编号从 0 到 19），现经过一系列的入队和出队运算后，队头变量（第一个元素的位置） $front=18$ ，队尾变量（待插入元素的位置） $rear=11$ ，在这种情况下，环形队列中有 13 个元素。

点评：

$$number = (rear + size - front) \bmod size$$

6. 在列表 $A=list(range(1,13))$ 中二分查找 11，比较的第一个元素下标为 5，后面依次比较的元素下标是 8,10。

点评：回忆一下二分查找的过程，从 1 到 12 共 12 个数，中位数可以是 6 或 7，这里是 6（下标为 5），可见计算中位数使用的是向下取整...，注意后面问的也都是下标。

7. 对长度为 3 的顺序表进行查找, 若查找第一个元素的概率为 $1/2$, 查找第二个元素的概率为 $1/4$, 查找第三个元素的概率为 $1/8$, 则执行任意查找需要比较元素的平均个数为 1.75 或 $7/4$ 。

点评: 注意 $1/2, 1/4, 1/8$ 加起来和 1 还差 $1/8$, 这是查找的数不在表中的概率。

8. 使用栈计算前缀表达式 (操作数均为一位数字) “ $- 1 + * + 2 3 4 5 3$ ”, 当读入第二个加号 (还未进行处理) 时, 栈的内容 (以栈底到栈顶从左往右的顺序书写) 为 3 5 20。

点评: 前缀表达式从后往前扫描。

9. 使用 KMP 算法查找模式串 “ababaaababaa”, 得到的模式值数组 next 是:
[0, 0, 1, 2, 3, 1, 1, 2, 3, 4, 5, 6]

点评: 模式值数组就是各个前缀的最长共同前后缀。

10. 后缀表达式中操作符紧跟在对应操作数的后面, 把 “ $[(c-b)+(b*b-4*a*c)]/(2*a)$ ” 转换为后缀形式:
c b - b b * 4 a * c * - + 2 a * /。

点评: 按照概念写, 送分题。

11. 设有一组记录的关键字为 {19, 14, 23, 1, 68, 20, 84, 27, 55, 11, 10, 79}, 用链地址法 (拉链法) 构造散列表, 散列函数为 $H(\text{key}) = \text{key} \bmod 13$, 散列地址为 1 的链中有 4 个记录。

点评: 有四个数: 1、14、27、79, 除以 13 的余数为 1。

12. 下面的程序段为什么不是递归: 没有减小问题规模或向基本结束条件演进。

```
1 f = lambda x: (f(x+1)+f(x-1))/2 if x>0 else 0
```

点评: $f(x)$ 的求解中出现了 $f(x+1)$, 没有减小问题规模。

2. (4 分) 字符串匹配算法从长度为 n 的文本串 S 中查找长度为 m 的模式串 P 的所有出现。字符串匹配的朴素算法使用暴力搜索, 大致过程如下: 对于 P 在 S 中可能出现的 $n-m+1$ 个位置, 比对此位置时 P 和 S 对应的子串是否相等。通常说的复杂度 $O((n-m+1)m)$, 只出现在它的最坏情况。
- 举例说明算法复杂度一种最坏情况。
 - 如果 P 和 S 都属于语言 $L \subset \Sigma^*$, 字典 Σ 的字符数目为 k , 在 L 中等概率随机出现。假设 $n \gg m \gg 0$, 求朴素算法在平均情况下所需的比对次数。

答: 1、例如 $P = "0^{m-1}1"$, $S = "0^{n-1}1"$, 执行朴素算法刚好需要 $(n-m+1)m$ 次比对。

2、对于 P 在 S 中可能出现的 $n-m+1$ 个位置, 如发现不匹配则比对结束, 比对一个字符结束的概率为 $\frac{k-1}{k}$, 比对两个字符结束的概率为 $\frac{1}{k} \cdot \frac{k-1}{k}$, ..., 每个位置需要的比对次数平均为:

$$\begin{aligned} & \frac{k-1}{k} \cdot 1 + \frac{1}{k} \cdot \frac{k-1}{k} \cdot 2 + \frac{1}{k^2} \cdot \frac{k-1}{k} \cdot 3 + \dots + \frac{1}{k^{m-1}} \cdot \frac{k-1}{k} \cdot m \\ &= (k-1) \left(\frac{1}{k} + \frac{2}{k^2} + \frac{3}{k^3} + \dots + \frac{m}{k^m} \right) \\ &\approx (k-1) \frac{\frac{1}{k}}{(1 - \frac{1}{k})^2} \\ &= \frac{k}{k-1} \end{aligned}$$

总的比对次数平均为: $(n-m-1) \cdot k / (k-1) = O((n-m-1) \cdot k / (k-1))$

3. (2 分) 什么是函数值缓存? 它在递归算法中有什么作用?

答: 1、函数值缓存是将某个函数在给定输入的结果进行存储, 下次调用输入相同时直接使用该结果。
2、它在递归算法中可以避免重复的递归调用, 从而提高算法效率。

五、（12分）算法填空。

1. （4分）给定一个已排序的链表的头 head，删除原始链表中所有重复数字的结点，只留下不同的数字。返回已排序的链表。例如：

1. 输入为 1->1->2->5->6->6->8，输出链表应该为 2->5->8
2. 输入为 1->2->4->4->4->5->5，输出链表应该为 1->2

请分别在下面的 A、B、C、D 位置补全所需的代码：

```
1 class ListNode:
2     def __init__(self, val=0, next=None):
3         self.val = val
4         self.next = next
5
6 def deleteDuplicates(head: ListNode) -> ListNode:
7     if not head:
8         return head
9
10    p = ListNode(0, head)
11
12    cur = p
13    while cur.next and cur.next.next:
14        if A) cur.next.val == cur.next.next.val:
15            x = cur.next.val
16            while cur.next and cur.next.val == x:
```

第 7 页（共 9 页）

I

北京大学

数据结构与算法 B

5 班期中考试

```
17         B) cur.next = cur.next.next
18     else:
19         C) cur = cur.next
20
21     D) return p.next
```

点评：注意题目的要求是去除所有重复了的数字，第一个也不保留。p 用来保存“无用”的头结点；由 cur 扫描链表，执行结点的删除。

点；由 cur 扫描链表，执行结点的删除。

2. （4分）我们实现的栈只提供了下面的四个操作，

- stack(): 创建一个空栈
- push(item): 将 item 入栈
- pop(): 弹出并返回栈顶元素
- isEmpty(): 判断栈是否为空。

现在需要你仅仅用两个栈 s1 和 s2 来实现一个队列 que，请分别在下面的 A、B、C 位置补全它的 enqueue、deque 和 isEmpty 操作。可以写 python 代码也可以用伪代码描述，要求他们的平均复杂度为 O(1)。

```

1 class que:
2     def __init__(self):
3         self.s1 = stack()
4         self.s2 = stack()
5     def enqueue(self, item):
6         A) self.s1.push(item)
7     def deque(self):
8         B) if not self.s2.empty(): return self.s2.pop()
9         while not self.s1.empty(): self.s2.push(self.s1.pop())
10        if not self.s2.empty(): return self.s2.pop()
11    def isEmpty(self):
12        C) return self.s1.empty() and self.s2.empty()

```

点评：题目强调了他们的平均复杂度为 $O(1)$ 。假设一共有 m 个元素入队并出队，每个元素都会经历同样的过程：{入 $s1$ ；出 $s1$ ；入 $s2$ ；出 $s2$ }。那么总的入栈/出栈操作数量为 $4m$ ，平均下来就是 $4=O(1)$ 。

3. (4 分) 一个只有头指针的单链表，由于操作失误导致它中间可能出现了环，单链表的形状变成了阿拉伯数字“6”这样的。请分别在下面的 A、B、C 位置补全算法代码，来判断这种情况。假设单链表原来的长度为 N ，要求算法的复杂度为 $O(N)$ 。

```

1 class ListNode:
2     def __init__(self, val=0, next=None):
3         self.val = val
4         self.next = next
5
6 def findRing(head: ListNode) -> bool:
7     if head is None or head.next is None:
8         return False
9     i, j = head, head.next
10    while A) i is not j:
11        if B) j is None or j.next is None:
12            return False

```

```

13         C) i, j = i.next, j.next.next
14     else:
15         return True

```