

南京理工大学

2019 年硕士学位研究生入学考试试题

科目代码: 825 科目名称: 计算机专业基础 (B) 满分: 150 分

注意: ①认真阅读答题纸上的注意事项; ②所有答案必须写在答题纸上, 写在本试题纸或草稿纸上均无效; ③本试题纸须随答题纸一起装入试题袋中交回!

第一部分 数据结构 (共 75 分)

一、选择题 (每题 1 分, 共 15 分)

- 一个非连通的无向图, 共有 45 条边, 则该图至少有 () 个顶点。
A. 10 B. 11 C. 12 D. 13
- 分别采用堆排序, 快速排序, 直接插入排序和归并排序, 对初态为有序的表, 则最省时间的是 () 算法。
A. 堆排序 B. 快速排序 C. 直接插入排序 D. 归并排序
- 定义一个一维数组 A[50] 来表示一个 10 行 5 列的 2 维数组 M, 每个元素占据一个位置, 当采用行优先存储机制的时候, 数组元素 M(2,3) 在 A 中的位置是 ()。
A. 17 B. 13 C. 23 D. 32
- 已知关键字的集合 {63, 78, 99, 55, 40, 75, 43, 71, 30, 120, 110}, 构造二叉排序树, 该二叉排序树的树高为 ()。
A. 7 B. 6 C. 5 D. 4
- 对二叉排序树采用 () 遍历可以得到一个递增序列。
A. 先序 B. 中序 C. 后序 D. 层次
- 已知有向图 $G=(V,E)$, 其中 $V=\{1,2,3,4\}$, $E=\{(1,3),(3,4),(2,4)\}$, 添加弧 () 后, 则有唯一的拓扑序列。
A. (2,3) 或 (1,2) B. (2,1) 或 (3,2) C. (2,1) 或 (1,4) D. (1,2) 或 (1,4)
- 已知关键字的集合 {49, 38, 65, 97, 76, 13, 27, 48, 55, 4}, 从低位到高位采用基数排序, 则经过一趟基数排序后的结果为 ()。
A. 13, 4, 55, 65, 76, 27, 97, 38, 48, 49 B. 13, 4, 65, 55, 76, 97, 27, 38, 48, 49
C. 13, 4, 55, 65, 76, 97, 27, 38, 48, 49 D. 13, 4, 65, 55, 76, 97, 27, 48, 38, 49
- 以下序列为堆的是 ()。
A. {60, 70, 82, 91, 84, 100, 75, 98, 120} B. {76, 86, 92, 82, 88, 100, 160, 92, 200}
C. {120, 100, 92, 86, 78, 84, 60, 40, 76, 16} D. {200, 108, 90, 98, 180, 70, 60, 20, 30}
- 在二叉排序树上查找关键字为 52 的结点, 假设该结点存在, 则依次比较关键字有可能的是 ()。
A. 60, 72, 50, 42 B. 60, 50, 55, 52 C. 30, 20, 50, 52 D. 55, 35, 25, 52
- 一个无向图有 25 条边, 度为 5 的顶点有 1 个, 度为 4 的顶点有 3 个, 其余顶点的度数均小于等于 3, 则该图中至少有 () 个顶点。
A. 16 B. 14 C. 15 D. 17
- 折半查找的时间复杂度为 ()。
A. $O(\log n)$ B. $O(n)$ C. $O(n \log n)$ D. $o(n^2)$
- 4 阶 B-树中, 每个结点最少有 () 个关键字。

- A. 2 B. 3 C. 1 D. 4
13. 设 $\langle V_1, V_2 \rangle$ 是 AOE 网中的一条弧, 弧的权是 6, 若顶点 V_2 的最早发生时间是 22, 最迟发生时间是 32, 则活动 $\langle V_1, V_2 \rangle$ 的最迟发生时间是 ()。
A. 28 B. 32 C. 26 D. 16
14. 已知关键字的集合 {40, 12, 45, 8, 9, 18, 99, 1, 80, 11, 29}, 采用冒泡排序从小到大排序要经过 () 次比较。
A. 54 B. 55 C. 56 D. 57
15. 根据使用频率为 4 个字符设计的 Huffman 编码不可能的是 ()。
A. 111, 110, 10, 0 B. 000, 001, 01, 1 C. 01, 00, 10, 00 D. 100, 110, 10, 0

二、填空题 (每空 1 分, 共 10 分)

- 算法效率的度量方法常分为 (1) 和事前估算方法 2 种方法。
- 假定有 35 个关键字值互为同义词, 若采用线性探测再散列把这 35 个关键字值存入散列表中, 至少要进行 (2) 次探查。
- 由关键字的集合 {70, 40, 80, 55, 45, ...} 构造平衡二叉树, 当插入 45 时引起不平衡, 则其旋转类型为 (3)。
- 设有 40 个值, 用它们构造哈夫曼树, 则该哈夫曼树共有 (4) 结点。
- 长度为 60 的有序表采用折半查找, 共有 (5) 个元素的查找长度为 6。
- 一组数据为 {49, 38, 65, 97, 76, 13, 27, 49}, 现在用某种排序算法进行一趟后产生如下序列 {27, 38, 13, 49, 76, 97, 65, 49}, 则采用的是 (6) 排序算法。
- 散列表的地址区间为 0-17, 散列函数为 $H(K)=K \bmod 17$ 。采用线性探测法处理冲突, 并将关键字序列 26, 25, 72, 38, 8, 18, 42 依次存储到散列表中。元素 42 存放在散列表中的地址是 (7)。
- 对于双向链表, 插入一个结点需修改的指针共 (8) 个。
- 普里姆算法的时间复杂度为 (9), 它对 (10) 图较为适合。

三、问答题 (共 26 分)

- 已知二叉树的后序序列为 GDIHEBKJFONMPLCA, 中序序列为 GDBHIEAKFJCMONLP。
(1) (4 分) 写出该二叉树;
(2) (2 分) 写出先序序列;
(3) (4 分) 写出该二叉树对应的森林。
- 已有序输入序列 {48, 20, 35, 33, 50, 70, 31}
(1) (4 分) 生成一棵平衡二叉树;
(2) (3 分) 构造 3 阶 B-树, 并分别写出依次删除 33 和 50 后的 B-树;
(3) (3 分) 若初始的关键字序列为该序列, 写出 2-路归并排序的过程。
- (6 分) 某通讯系统使用 8 种字符 a、b、c、d、e、f、g、h, 其使用频率分别为 0.05, 0.27, 0.07, 0.08, 0.14, 0.23, 0.04, 0.12。构造以字符使用频率作为权值的哈夫曼树, 并给出每个字符对应的哈夫曼编码。

四、算法设计 (每题 8 分, 共 24 分)

1. 二叉树的定义如下:

```
typedef struct bitnode{
    int data;
    bitnode *lchild, *rchild;
}bitnode;
```

已知函数 $initstack(s)$ 表示初始化建立一空栈, $emptystack(s)$ 表示栈的判空函数, 若栈为空返回 1, 否则返回 0; $push(s,p)$ 表示压栈操作, $pop(s,p)$ 表示退栈操作。利用已知函数构造

- 一个计算二叉树叶子数的非递归算法。函数原型为 `int countleaf(bitnode *t)`。
- 已知二叉排序树的数据关系定义如题 1 所示，分别设计一个返回二叉排序树中最小值结点的地址的递归和非递归算法。函数原型为 `bitnode *findMin(bitnode *t)`。
 - 有带头结点的递增有序单链表 `heada` 和 `headb`，结点类型为 `(data,next)`，`heada` 和 `headb` 分别存储集合 A 和集合 B 中的元素，设计一个计算 $A \cap B$ 的算法，并将结果存放在 `heada` (注 `heada` 中不能有相同元素) 中，同时返回链表 `heada` 的元素个数。函数原型为 `int intersection(lnode *&heada, lnode *headb)`。

第二部分 操作系统 (共 75 分)

五、选择题 (每题 1 分, 共 10 分)

- 不属于操作系统特征的是 ()。
 - 并行性
 - 共享性
 - 虚拟性
 - 异步性
- 不一定会导致 CPU 进入内核态的操作是 ()。
 - 访寄存器
 - 访中断向量表
 - 读文件
 - 读键盘
- 进程进入就绪态不可能是因为 ()。
 - 该进程创建成功
 - 某进程抢占 CPU
 - 该进程执行 wait 操作
 - 某进程执行 signal 操作
- 系统中有 2 个不同的临界资源 R1 和 R2, 4 个进程 p1、p2、p3 和 p4, 其中 p1 和 p2 都申请 R1 和 R2, p3 申请 R1, p4 申请 R2。若系统出现死锁, 则处于死锁状态的进程数至少是 () 个。
 - 1
 - 2
 - 3
 - 4
- 进程调度算法的设计目标不包括 ()。
 - 提高 CPU 利用率
 - 提高系统吞吐量
 - 提高平均带权周转时间
 - 降低算法自身复杂度
- 在请求分页系统中, 页面分配策略与页面置换策略不能组合使用的是 ()。
 - 可变分配, 局部置换
 - 可变分配, 全局置换
 - 固定分配, 局部置换
 - 固定分配, 全局置换
- 使用 I/O 缓冲技术的先决条件是 ()。
 - 设备 I/O 速度与 CPU 速度匹配
 - 设备带宽高于 CPU 带宽
 - 设备带宽低于 CPU 带宽
 - 用户请求
- SPOOLING 系统中输入井和输入缓冲区分别位于 ()。
 - 内存和磁盘
 - 磁盘和内存
 - 寄存器和内存
 - 内存和寄存器
- 当应用程序试图请求从打印机输入数据, () 负责拒绝此请求。
 - 中断处理程序
 - 设备独立性软件
 - 应用程序
 - 设备驱动程序
- 某 500 个盘块的文件的目录项已在内存 (若为索引分配, 其索引块也在内存)。若需要在文件倒数第二块删除, 以下分配方式中操作磁盘 I/O 的次数最多的是 ()。
 - 连续分配
 - 隐式链接分配
 - 显示链接分配
 - 索引分配

六、判断题 (每题 1 分, 共 10 分)

- 分时操作系统和实时操作系统均具有及时性的特点。
- 抢占式最短剩余时间优先调度算法适用于高级调度。
- 段页式存储管理中内碎片和外碎片都有可能产生。
- 进程执行原语时屏蔽中断。
- 利用整型信号量机制解决同步问题, 能够遵循所有四条同步准则。
- 当线程执行一个阻塞的系统调用时, 其所属进程内所有线程都会被阻塞。
- 由于死锁会带来多个进程的僵持状态, 因此操作系统一定要处理死锁。
- 无论系统采用何种程序装入方式, 装入时都是生成物理地址的时刻。

- 基于索引节点的文件共享方式中, 当文件所有者删除共享文件, 其它用户将无法访问该文件。
- 文件若没有被打开, 则没有该文件的内存索引节点。

七、填空题 (每空 1 分, 共 20 分。请在答题纸上标注出每个答案对应的空格编号。)

- 与单道程序相比, 多道程序系统的优点是资源利用率高和 (1)。
- 在单 CPU 系统中有 n 个进程, 则处于阻塞态的进程个数最多为 (2) 个。
- 某计算机按字节编址, 其动态分区内存管理采用最差适应算法。当前空闲分区信息如表 1 所示。经过回收起始地址为 70K、大小为 180KB 的分区后, 系统中空闲分区有 (3) 个, 最大的分区为 (4) KB; 再在响应了大小为 30KB 的分配要求后, 系统中最大空闲分区的起始地址为 (5) K。

表 1

分区起始地址	50K	1000K	200K	500K
分区大小	20KB	40KB	100KB	200KB

- 用于连接一个读进程和一个写进程以实现它们之间通信的共享文件被称为 (6)。
- 在设备管理中, I/O 软件中的 (7) 软件负责将逻辑设备名映射为物理设备名, 并将映射信息记录到 (8) 中。
- 若用文件保存一本英汉词典, 每个单词及其解释组成一个记录。为了使用户能够快速查找所需单词, 该文件的逻辑结构应为 (9)。
- 根据表 2 中的段表, 其中存在位 1 表示段在内存, 存取控制字段中 W 表示只写, R 表示只读, E 表示可执行。执行 STORE R1, [0, 100] 会发生 (10), 执行 LOAD R1, [1, 12] 会发生 (11), 执行 JMP [2, 12] 会发生 (12), 执行 JMP [2, 400] 会发生 (13)。

表 2

段号	存在位	基址	段长	存取控制
0	0		600	R
1	1	2300	14	W
2	1	90	100	E

- 文件系统用位示图法表示磁盘空间的分配情况, 位示图存于磁盘 30~120 号块中, 每个盘块占 1KB。该位示图所能表示的磁盘最大容量为 (14); 若要释放 34567 号盘块, 则需将 (15) 号盘块内的第 (16) 个字节的第 (17) 位改为 0 (所有编号均从 0 开始)。
- 系统采用页式虚拟存储管理, 采用请求调页、固定分配局部置换策略。某进程有 8 个逻辑页, 系统为其分配了 3 个物理页框。在页均未装入的情况下, 进程依次引用了页面 (括号内给出了引用方式): 7 (修改), 0 (访问), 1 (修改), 2 (访问), 0 (访问), 3 (修改), 0 (访问), 2 (访问)。若采用 LRU 算法, 缺页率为 (18); 若采用 Clock 置换算法, 缺页率为 (19); 若采用改进型 Clock 置换算法, 缺页率为 (20)。(设物理页框按页框号从小到大依次分配。Clock 和改进型 Clock 算法中也是按同样顺序链接成一个循环队列, 查询指针第一次检查时从最小号的页框开始。)

八、综合题 (35 分)

- (7 分) 在单 CPU 和两台独占型 I/O (I1, I2) 设备的多道程序环境下, 三个进程 P1、P2 和 P3 的执行轨迹如下:

P1: I2(10ms) CPU(6ms) I2(20ms)

P2: CPU(5ms) I1(20ms) CPU(10ms) I2(10ms)

P3: CPU(20ms) I1(5ms) CPU(5ms) I2(5ms)

P1 和 P2 在 0ms 时创建, P3 在 3ms 时创建。如果 CPU、I1、I2 都能并行工作, I1 和 I2 采用 FCFS 调度策略, 而 CPU 采用 RR 调度算法 (时间片为 5ms)。不考虑调度和切换的时间, 计

算：

- 1) 每个进程的结束时间。(3分)
 - 2) 平均周转时间。(1分)
 - 3) I1、I2和CPU的利用率。(3分)
2. (7分) 假设磁盘系统采用CSCAN调度策略。设某单面磁盘转速为每分钟6000转，每个磁道有100个扇区，相邻磁道间的平均移动时间为1ms。若在某时刻，磁头位于100号磁道处，并沿着磁道号增大的方向移动。最初，磁道号请求队列为50、90、30、120，磁头运行40ms后又到来了新的磁道号请求队列为110、60。请回答：
- 1) 磁道号访问序列是？(3分)
 - 2) 若对请求队列中的每个磁道需读取1个随机分布的扇区，则读完6个扇区总共需要多少时间？(4分)
3. (7分) 某文件系统的最大容量为4TB，以磁盘块为基本分配单位，盘块大小为1KB，采用索引分配方法。
- 1) 每个盘块号至少需要用多少个字节来描述？(1分)
 - 2) 若某文件大小为4MB，则该文件系统至少需要设置几级间接索引？为什么？至少需占用多少个盘块(包括索引块)？(3分)
 - 3) 若按上面需要相应级间接索引，那么该文件字节偏移量2000对应的物理盘块号如何查找？给出具体步骤。块内位移量为多少？(3分)
4. (8分) 某系统主存按字节编址，内存采用具有快表的分页管理机制，页框大小为8B，每个页表项占2B。快表中最多只能存放2个页表单元，且采用最近最久未使用的调出策略。CPU访问快表时，会根据虚地址的高n位 ($n = \text{虚地址总位数} - \text{页内地址位数}$) 在快表中查找，直接得到对应的物理页框号。CPU访存一次时间为50ns，访快表一次时间为10ns。设快表初始为空，快表未命中时，忽略快表的更新时间。进程P的逻辑地址空间为128B，其外层页表寄存器中保存的地址(物理页框号)为4。图1给出了进程P各级页表的内容(除页框号外的其他信息都省略了)。进程P有逻辑地址访问序列为20、40、18、15、42。本题所有数据均为十进制。
- 1) 画出进程P的逻辑地址结构。(2分)
 - 2) 访问序列中逻辑地址20对应的物理地址是多少(十进制)？本次访问需多少时间？(2分)
 - 3) 访问序列中逻辑地址18对应的物理地址是多少(十进制)？本次访问需多少时间？(2分)
 - 4) 访问序列中逻辑地址42对应的物理地址是多少(十进制)？本次访问需多少时间？(2分)

10	6	13	34	23
20	18	8	29	17
30	21	24	5	39
40	33	22	15	9

4号页框内容 10号页框内容 20号页框内容 30号页框内容 40号页框内容

图1

5. (6分) 某蛋糕房制作纸杯蛋糕，有m个原料桶、n个蛋糕模具、1个烤箱和1个搅拌机，具体流程和规定如下：
- 1) 原料搅拌中需要使用1个原料桶和搅拌机，1个搅拌机同时只能搅拌一桶原料；
 - 2) 原料搅拌好后，就能进行装模操作：将原料桶中的材料倒入蛋糕模具中，一个桶中的原料正好用于一个模具；空的原料桶可以继续为原料搅拌所使用；
 - 3) 在烤箱空闲时才能放入装好料的蛋糕模具，数量达到3个时，烤箱才可进行烘烤；

- 4) 烘烤完成，3个蛋糕模可以出烤箱，取出蛋糕后，可继续装模使用，空烤箱也可继续装入模具；
- 5) 在资源允许的情况下，原料搅拌、装模和烘烤均可并行进行；资源不足时等待。试用P、V操作填空，描述以下三个进程的工作过程。

semaphore blender=1, bucket=m, mould=n, room=3, s1=0, s2=0, mutex=1;
int count=0;

```

搅拌进程:
while(1){
    (1);
    装原料;
    (2);
    放桶搅拌;
    signal(blender);
    取桶;
    (3);
}

```

```

装模、装箱进程:
while(1){
    wait(s1);
    wait(mould);
    装模;
    (4);
    (5);
    模具放入烤箱;
    wait(mutex);
    count++;
    if(count==3)
        (6);
    signal(mutex);
}

```

```

烘烤、取蛋糕进程:
while(1){
    wait(s2);
    烘烤;
    取模具;
    wait(mutex);
    count=0;
    signal(mutex);
    signal(room);
    signal(room);
    signal(room);
    取蛋糕;
    signal(mould);
    signal(mould);
    signal(mould);
}

```