# 2024-2025秋-数据结构与算法-试题回忆

| 编者 | Egopposer（line2345） |
| --- | --- |
| 日期 | 2025/1/3 |
| 学院 | 大数据与软件学院 |
| 课程代码 | SE21014 |

## 0、前言及试题概览

*斜体*为记忆模糊的题目，**黑体**为我认为需要关注的点。

**试题概览：**（主观评价，仅供娱乐）

| 题量 | 中上（80分钟左右） |
| --- | --- |
| 难易度（软院专业课中） | 5/10 |
| 送分题占比 | 送分比 $=\begin{cases}90\% & \text{if 你学了这部分}\\ 0\% & \text{if 你没学这部分}\end{cases}$ |
| 背诵记忆占比 | 0%（没有任何需要硬背的内容） |
| 21-23总均分 | 83.036 |
| 21-23平均满绩率 | 37% |

**注**:

1. 只有大题，无选择填空判断等小题，不考察手写代码（今年意外的还有代码阅读）。
2. **本卷出现的数字、字母序列及相关数据均为自行编撰**（谁记得住啊），可能导致构建出来的数据结构混乱，考查知识点不全面等情况，建议自行斟酌，全面复习相关内容。
3. *本卷未考察（高级）表，栈，队列。

## 一、简答题（48分）

1. 根据下面的程序片段，*写出其对应的时间**表达式***，用$\Theta$表示法表示。（10分）

(a):

```
for(j=3;j<n;n*=2){
    for(i=1;i<n;i++){
        a++;
    }
}
```

(b):

```
for(i=1;i<n;i*=2){
    func1(n); // func(n)的时间复杂度为n^2
    for(j=1;j<n;j++)
        func2(); // func2()的时间复杂度为常数倍
}
```

(c):

```
for(k=n;k>1;k=n/2){
    for(i=1;i<k;i++){
        a++;
    }
}
```

(d):

```
for(k=n;k>1;k=k/2){
    for(i=1;i<n;i*=2){
        a++;
    }
}
```

(e):

```
// 忘了
// 诶，我给你们出一个不就好了☝🤓

/*试分析Strassen矩阵乘法的时间复杂度*/
def strassen(A, B):
    n = len(A)
    if n == 1:
        return [[A[0][0] * B[0][0]]]

    mid = n // 2
    A11 = [row[:mid] for row in A[:mid]]
    A12 = [row[mid:] for row in A[:mid]]
    A21 = [row[:mid] for row in A[mid:]]
    A22 = [row[mid:] for row in A[mid:]]

    B11 = [row[:mid] for row in B[:mid]]
    B12 = [row[mid:] for row in B[:mid]]
    B21 = [row[:mid] for row in B[mid:]]
    B22 = [row[mid:] for row in B[mid:]]

    M1 = strassen(matrix_add(A11, A22), matrix_add(B11, B22))
    M2 = strassen(matrix_add(A21, A22), B11)
    M3 = strassen(A11, matrix_sub(B12, B22))
    M4 = strassen(A22, matrix_sub(B21, B11))
    M5 = strassen(matrix_add(A11, A12), B22)
    M6 = strassen(matrix_sub(A21, A11), matrix_add(B11, B12))
    M7 = strassen(matrix_sub(A12, A22), matrix_add(B21, B22))

    C11 = matrix_add(matrix_sub(matrix_add(M1, M4), M5), M7)
```

```
    C12 = matrix_add(M3, M5)
    C21 = matrix_add(M2, M4)
    C22 = matrix_add(matrix_sub(matrix_add(M1, M3), M2), M6)

    C = []
    for i in range(mid):
        C.append(C11[i] + C12[i])
    for i in range(mid):
        C.append(C21[i] + C22[i])
    return C

def matrix_add(A, B):
    n = len(A)
    C = [[0 for j in range(n)] for i in range(n)]
    for i in range(n):
        for j in range(n):
            C[i][j] = A[i][j] + B[i][j]
    return C

def matrix_sub(A, B):
    n = len(A)
    C = [[0 for j in range(n)] for i in range(n)]
    for i in range(n):
        for j in range(n):
            C[i][j] = A[i][j] - B[i][j]
    return C
// 此题需要使用主定理进行求解，使用主定理进行时间复杂度分析的方法将在下个学期开设计《算法设计与分析》[SE31036]中进行讲解。
// c++写出来声明太多太乱了，用python写的
// 答案见试卷末尾（只有此题有(´·ω·`)）
```

2. 按照给出的序列*{1,17,23,2,10,45,55,56,70,14}*，使用buildheap()方法构建最大堆。（8分）
3. 按照序列*{40,17,23,2,10,45,55,56,70,14}*的插入顺序，构建一棵二叉搜索树，并给出这棵树的前序、中序、后序遍历的结果。（10分）
4. 在{堆排序，希尔排序，归并排序，快速排序}中，选择一个**稳定**的排序算法，对下列序列{}进行排序。（10分）
5. 将数据{26,32,69,68}分别插入到如下AVL树中，每次插入后的结果分别是什么？若发生旋转，指明旋转的类型。（10分）

```
        50
       /  \
     30    70
    /      / \
  20      60  80
```

（原题用四次插入将四种情况RR,RL,LL,LR全部考察了一遍，当然我编不出来）

# 二、应用题（52分）

1. 现在希望使用散列表来存储数据，储存的地址空间为0-9，使用除余法和线性探测法，使用的hash函数为$h(k) = k \bmod 10$。按照{27、33、77、57、12、7}的顺序依次插入到散列表当中。(12分)
    1. 给出填充完毕的散列表。
    2. 给出下一次填充时，每一个**空槽**被填充的概率。
2. 给出如下字符及其相应的出现频率，完成以下题目：（14分）

| A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|
| 0.21 | 0.03 | 0.05 | 0.16 | 0.15 | 0.1 | 0.17 | 0.13 |

    1. 构建相应的Huffman树。
    2. 对每个字符进行编码。
    3. 假如对以上字符进行等长编码，需要的长度（bit）是多少？使用Huffman编码的平均长度又是多少？
3. 给出下列**字母序列** *{A,R,S,B,H,I,P,L,H,I,U,Q,G}*，**构建一棵4阶B+树，假设每个叶节点最多存储3个数据（原描述）。**（12分）
4. 按照下列给出的G的邻接表，完成以下题目：(14分)

```
1: [(3, 3), (2, 2)]
2: [(1, 2), (4, 2)]
3: [(1, 3), (4, 1)]
4: [(2, 2), (3, 1), (5, 2), (6, 3)]
5: [(4, 2), (6, 1), (7, 2)]
6: [(4, 3), (5, 1), (8, 4)]
7: [(5, 2), (8, 3)]
8: [(6, 4), (7, 3)]
每一行的格式为：节点: [(邻接节点, 权重), ...]。
```

    1. 写出G的邻接矩阵
    2. 写出从点1开始的深度优先遍历和广度优先遍历的遍历序列
    3. 使用prim算法构建图G的最小生成树

# 三、附录

## 时间复杂度分析

1. 设算法的时间复杂度为 $T(n)$，根据主定理分析：
    - 每次将 $n \times n$ 的问题分解为 7 个 $\frac{n}{2} \times \frac{n}{2}$ 的子问题，即 $a = 7$
    - 每次把问题规模缩小一半，即 $b = 2$
    - 矩阵加减法的额外操作时间为 $O(n^2)$，即 $f(n) = O(n^2)$
2. 可以列出递归式：
   $T(n) = 7T\left(\frac{n}{2}\right) + O(n^2)$
3. 应用主定理：
    - $\log_2 7 \approx 2.807 > 2$
    - 因此 $T(n) = O(n^{\log_2 7}) \approx O(n^{2.807})$

孩子不懂写着玩的，没事就不要看了，也不会考。