CCF 编程能力等级认证
Grade Examination of Software Programming

Python　六级

2024 年 12 月

1　单选题（每题 2 分，共 30 分）

题号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
答案	B	A	B	A	C	C	B	D	A	A	B	B	A	D	D

第 1 题 关于哈夫曼树，下面说法正确的是（ ）。

- ☐ **A.** 不可能是满二叉树
- ☐ **B.** 哈夫曼树是一种用于数据压缩的二叉树
- ☐ **C.** 权值较大的结点离根较远
- ☐ **D.** 构建哈夫曼树的时间复杂度为$O(logn)$

第 2 题 给定一组权值{3, 4, 7, 14, 15, 20}，计算带其权路径长度（WPL）为（ ）。

- ☐ **A.** 147
- ☐ **B.** 146
- ☐ **C.** 142
- ☐ **D.** 145

第 3 题 二叉树T，已知其先根遍历是 1 2 4 3 5 7 6（数字为结点的编号，以下同），中根遍历是 2 4 1 5 7 3 6，则该二叉树的后根遍历是（ ）。

- ☐ **A.** 4 2 5 7 6 3 1
- ☐ **B.** 4 2 7 5 6 3 1
- ☐ **C.** 7 4 2 5 6 3 1
- ☐ **D.** 4 2 7 6 5 3 1

第 4 题 一棵二叉树的前序遍历序列是 ABCDEFG，后序遍历序列是 CBFEGDA，则根结点的左子树的结点个数可能是（ ）。

- ☐ **A.** 2
- ☐ **B.** 3
- ☐ **C.** 4
- ☐ **D.** 5

第 5 题 完全二叉树的顺序存储方案，是指将完全二叉树的结点从上至下、从左至右依次存放到一个顺序结构的数组中。假定根结点存放在数组的 1 号位置，则第 k 号结点的父结点如果存在的话，应当存放在数组的（ ）号位置。

- ☐ **A.** 2k
- ☐ **B.** 2k+1
- ☐ **C.** ⌊k/2⌋
- ☐ **D.** ⌊(k+1)/2⌋

第 6 题 如果根结点的深度记为 1，则一棵恰有 2011 个叶结点的二叉树的深度最少是（ ）。

- ☐ **A.** 10
- ☐ **B.** 11
- ☐ **C.** 12
- ☐ **D.** 13

第 7 题 广度优先搜索时，需要用到的数据结构是（ ）。

- ☐ **A.** 链表
- ☐ **B.** 队列
- ☐ **C.** 栈
- ☐ **D.** 散列表

第 8 题 如果一个栈初始时为空，且当前栈中的元素从栈底到栈顶依次为 a,b,c，另有元素 d 已经出栈，则可能的入栈顺序是（ ）。

- ☐ **A.** a,d,c,b
- ☐ **B.** b,a,c,d
- ☐ **C.** a,c,b,d
- ☐ **D.** d,a,b,c

第 9 题 在程序运行过程中，如果递归调用的层数过多，会因为（ ）引发错误。

- ☐ **A.** 系统分配的栈空间溢出
- ☐ **B.** 系统分配的堆空间溢出
- ☐ **C.** 系统分配的队列空间溢出
- ☐ **D.** 系统分配的链表空间溢出

第 10 题 一棵具有 5 层的满二叉树中结点数为( )。

- ☐ **A.** 31
- ☐ **B.** 32
- ☐ **C.** 33

- [ ] **D.** 16

第 11 题 今有一空栈 S，对下列待进栈的数据元素序列 a,b,c,d,e,f 依次进行进栈，进栈，出栈，进栈， 进栈，出栈的操作，则此操作完成后，栈 S 的栈顶元素为:

- [ ] **A.** f
- [ ] **B.** c
- [ ] **C.** a
- [ ] **D.** b

第 12 题 如果根的高度为 1，具有 61 个结点的完全二叉树的高度为（ ）

- [ ] **A.** 5
- [ ] **B.** 6
- [ ] **C.** 7
- [ ] **D.** 8

第 13 题 面向对象程序设计将对象作为程序的基本单元，将数据和程序封装在对象中，以提高软件的重用性、灵活性和扩展性。下面关于面向对象程序设计的说法中，不正确的是（ ）。

- [ ] **A.** 面向对象程序设计一般不采用自顶向下设计方法进行设计。
- [ ] **B.** 面向对象程序设计方法具有继承性、封装性和多态性等特点。
- [ ] **C.** 当前较为流行的面向对象的编程语言有 C++、JAVA、C# 等。
- [ ] **D.** 面向对象程序设计中对对象的成员属性的改变通常通过对象的成员函数实现。

第 14 题 前序遍历序列与中序遍历序列相同的二叉树为( )。

- [ ] **A.** 根结点无左子树
- [ ] **B.** 根结点无右子树
- [ ] **C.** 只有根结点的二叉树或非叶子结点只有左子树的二叉树
- [ ] **D.** 只有根结点的二叉树或非叶子结点只有右子树的二叉树

第 15 题 下面程序是一个二叉排序树的，横线处应该填入的是（ ）。

```
class BinarySortTree:
    def __init__(self):
        self.root = None

    def insert(self, key, value):
        node = TreeNode(key, value)
        if self.root is None:
            self.root = node
            return
        current = self.root
        while True:
            if key < current.key:
                ____________________________
```

```
                current = current.left
            else:
                if current.right is None:
                    current.right = node
                    return
                current = current.right

    def search(self, key):
        current = self.root
        while current:
            if current.key == key:
                return current.value
            elif current.key > key:
                current = current.left
            else:
                current = current.right
        return None

    def inorder_traversal(self, node):
        if node:
            self.inorder_traversal(node.left)
            print(node.key, node.value)
            self.inorder_traversal(node.right)
```

☐ **A.**

```
if current.left is None:
    current.right = node
    return
```

☐ **B.**

```
if current.right is None:
    current.right = node
    return
```

☐ **C.**

```
if current.right is None:
    current.left = node
    return
```

☐ **D.**

```
if current.left is None:
    current.left = node
    return
```

2　判断题（每题 2 分，共 20 分）

题号	1	2	3	4	5	6	7	8	9	10
答案	√	√	×	×	√	√	√	×	×	√

第 1 题　在哈夫曼树中，从树中一个结点到另一个结点之间的分支构成这两个结点间的路径。

第 2 题 满二叉树每一层的结点个数都达到了最大值。

第 3 题 如果一棵二叉树是满二叉树, 但是它不一定是完全二叉树。

第 4 题 栈中元素的插入和删除操作都在栈的顶端进行，所以方便用双向链表比单向链表更合适表实现。

第 5 题 格雷码是一种变权码，每一位码没有固定的大小。

第 6 题 格雷码的基本特点就是任意两个相邻的代码只有一位二进制数不同。

第 7 题 栈是一种只能在一端进行插入和删除操作的特殊线性表。

第 8 题 最后进入队列的元素才能最先从队列中删除。

第 9 题 当队列为满时，做出队运算产生的溢出现象,常用作程序控制转移的条件。

第 10 题 在循环队列的上下文中，rear指针通常用于指示队列尾部元素的下一个位置，而不是直接指示队列尾部的元素。因此，rear的计算通常与入队操作相关。

3 编程题（每题 25 分，共 50 分）

3.1 编程题 1

- **试题名称**：树上游走
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.1.1 题面描述

小杨有一棵包含无穷节点的二叉树（即每个节点都有左儿子节点和右儿子节点；除根节点外，每个节点都有父节点），其中根节点的编号为 1，对于节点 i，其左儿子的编号为 $2 \times i$，右儿子的编号为 $2 \times i + 1$。

小杨会从节点 s 开始在二叉树上移动，每次移动为以下三种移动方式的任意一种：

- **第1种移动方式**：如果当前节点存在父亲节点，向上移动到当前节点的父亲节点，否则不移动；
- **第2种移动方式**：移动到当前节点的左儿子；
- **第3种移动方式**：移动到当前节点的右儿子。

小杨想知道移动 n 次后自己所处的节点编号。**数据保证最后的所处的节点编号不超过** 10^{12}。

3.1.2 输入格式

第一行包含一个正整数 n, s，代表移动次数和初始节点编号。

第二行包含一个长度为 n 且仅包含大写字母 U, L, R 的字符串，代表每次移动的方式，其中 U 代表第1种移动方式，L 代表第2种移动方式，R 代表第3种移动方式。

3.1.3 输出格式

输出一个正整数，代表最后所处的节点编号。

3.1.4 样例

```
3 2
URR
```

```
7
```

3.1.5 样例解释

小杨的移动路线为 2-1-3-7。

子任务编号	数据点占比	n	s
1	20%	≤ 10	≤ 2
2	20%	≤ 50	≤ 10
3	60%	$\leq 10^6$	$\leq 10^{12}$

对于全部数据，保证有 $1 \leq n \leq 10^6, 1 \leq s \leq 10^{12}$。

3.1.6 参考程序

```python
INF = 1e12
n, s = map(int, input().split())
c = input()
st = []

for i in range(n):
    if c[i] == 'U':
        if s == 1:
            continue
        if st:
            st.pop()
            continue
        s >>= 1
    elif c[i] == 'L':
        if (s << 1) > INF:
            st.append('L')
            continue
        s = s << 1
    else:
        if (s << 1 | 1) > INF:
            st.append('R')
            continue
        s = s << 1 | 1
print(int(s))
```

3.2 编程题 2

- **试题名称**：运送物资
- **时间限制**：2.0 s
- **内存限制**：512.0 MB

3.2.1 题面描述

小杨管理着 m 辆货车，每辆货车每天需要向 A 市和 B 市运送若干次物资。小杨同时拥有 n 个运输站点，这些站点位于 A 市和 B 市之间。

每次运送物资时，货车从初始运输站点出发，前往 A 市或 B 市，之后返回初始运输站点。A 市、B 市和运输站点的位置可以视作数轴上的三个点，其中 A 市的坐标为 0，B 市的坐标为 x，运输站点的坐标为 p 且有 $0 < p < x$，货车每次去 A 市运送物资的总行驶路程为 $2p$，去 B 市运送物资的总行驶路程为 $2(x - p)$。

对于第 i 个运输站点，其位置为 p_i 且至多作为 c_i 辆车的初始运输站点。小杨想知道，在最优分配每辆货车的初始运输站点的情况下，所有货车每天的最短总行驶路程是多少。

3.2.2 输入格式

第一行包含三个正整数 n, m, x，代表运输站点数量，货车数量和两市距离。

之后 n 行，每行包含两个正整数 p_i, c_i，代表第 i 个运输站点的位置和最多容纳车辆数。

之后 m 行，每行包含两个正整数 a_i, b_i，代表第 i 辆货车每天需要向 A 市运送 a_i 次物资，向 B 市运送 b_i 次物资。

3.2.3 输出格式

输出一个正整数，代表所有货车每天的最短总行驶路程。

3.2.4 样例

```
3 4 10
1 1
2 1
8 3
5 3
7 2
9 0
1 10000
```

```
40186
```

3.2.5 样例解释

第 1 辆车的初始运输站点为站点 3，第 2 辆车的初始运输站点为站点 2。第 3 辆车的初始运输站点为站点 1，第 4 辆车的初始运输站点为站点 3。此时总行驶路程最短，为 40186。

子任务编号	数据点占比	n	m	c_i
1	20%	2	2	1
2	20%	$\leq 10^5$	$\leq 10^5$	1
3	60%	$\leq 10^5$	$\leq 10^5$	$\leq 10^5$

对于全部数据，保证有 $1 \leq n, m \leq 10^5, 2 \leq x \leq 10^8, 0 < p_i < x, 1 \leq c_i \leq 10^5, 0 \leq a_i, b_i \leq 10^5$。数据保证 $\sum c_i \geq m$。

3.2.6 参考程序

```python
n, m, x = map(int, input().split())
st = []
for _ in range(n):
    p, c = map(int, input().split())
```

```
    st.append((p, c))

st.sort()

a = [0] * m
b = [0] * m
for i in range(m):
    a[i], b[i] = map(int, input().split())

neg = []
pos = []
res = 0

for i in range(m):
    if a[i] >= b[i]:
        pos.append((a[i] - b[i], i))
    else:
        neg.append((a[i] - b[i], i))
    res += b[i] * x

neg.sort()
pos.sort(reverse=True)

l, r = 0, n - 1
for i in neg:
    while r >= 0 and st[r][1] == 0:
        r -= 1
    res += (a[i[1]] - b[i[1]]) * st[r][0]
    st[r] = (st[r][0], st[r][1] - 1)

for i in pos:
    while l < n and st[l][1] == 0:
        l += 1
    res += (a[i[1]] - b[i[1]]) * st[l][0]
    st[l] = (st[l][0], st[l][1] - 1)

print(res * 2)
```