9.

1）jal指令的立即数imm有21位，相较PC可以跳转的地址空间范围为$2^{21}bit, 2^{19}Byte$

2）条件分支指令的立即数imm有12位，相较PC可以跳转的地址空间范围为$2^{12}bit, 2^{10}Byte$

3）可以，lui指令将imm存到rd的高12位，jalr指令让PC=rs1+imm[20:0]，其中rs1就是lui指令的rd

10.

1）32位指令能被压缩为16位RVC指令的条件是：

- 立即数imm的长度有限，最长不能超过6位

2）RVC中指令只能访问x0-x15这16个通用寄存器的其中一个。

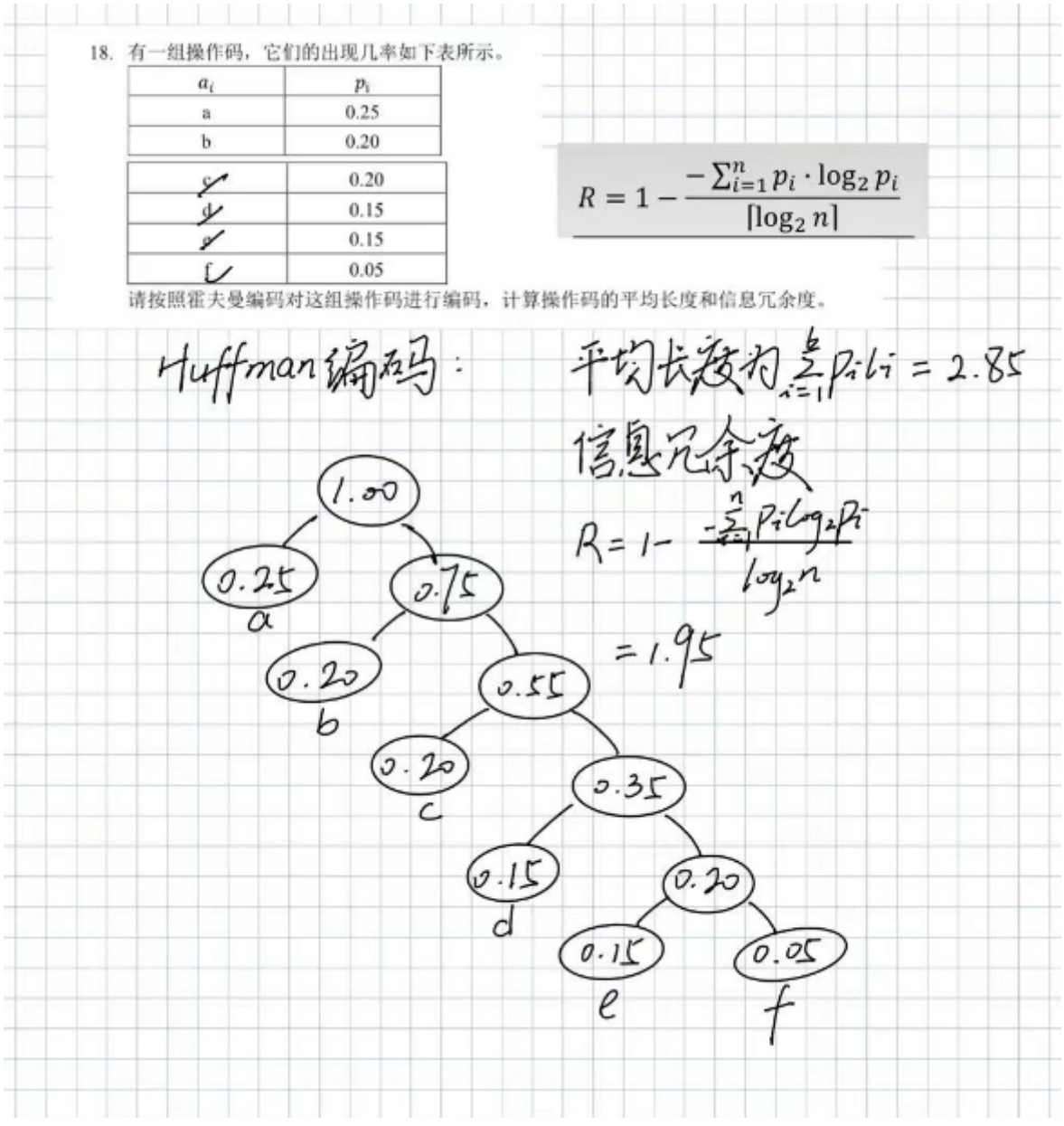
18.

18. 有一组操作码，它们的出现几率如下表所示。

a_i	p_i
a	0.25
b	0.20
c	0.20
d	0.15
e	0.15
f	0.05

$$R = 1 - \frac{-\sum_{i=1}^{n} p_i \cdot \log_2 p_i}{\lceil \log_2 n \rceil}$$

请按照霍夫曼编码对这组操作码进行编码，计算操作码的平均长度和信息冗余度。



19.

1）因为每次进行嵌套，都会额外申请栈空间，当嵌套调用层数过多，就会造成栈溢出

2）缓解或避免特定情况下的栈溢出问题：

- 增加栈的大小：可以增加线程或进程的栈大小，以便能够处理更多的数据。但是，这可能会导致内存分配和使用上的效率降低。
- 减少递归深度：在实现递归算法时，可以尝试减少递归深度以减少栈的使用量。可以考虑使用迭代算法或尾递归来代替递归。
- 使用非递归的数据结构：使用非递归的数据结构，如循环或迭代，可以减少递归深度，从而减少栈的使用量。
- 检查边界条件：在处理边界条件时，要确保不会导致栈溢出。例如，在使用递归算法时，要确保在调用递归函数之前检查基本情况。
- 使用动态内存分配：在某些情况下，使用动态内存分配可以避免栈溢出。这种方法的缺点是需要手动管理内存，容易出现内存泄漏和堆溢出等问题。
- 编写高效的代码：优化算法和数据结构，避免不必要的函数调用和重复计算，可以减少栈的使用量。
- 使用尾递归：尾递归是一种特殊的递归形式，可以在每次调用递归函数时重用相同的栈帧，从而减少栈的使用量。

20.

20. 假设有三个函数：F1、F2 和 F3。其中 F1 包含 1 个输入参数，计算过程使用寄存器 t0 和 s0；F2 包含 2 个输入参数，计算过程使用寄存器 t0-t1 及 s0-s1，返回一个 int 值；F1 执行过程中会调用 F2，F2 执行过程中会调用 F3。下表模拟了 F1 执行过程中栈的内容，其中第一行为 F1 函数被首次调用时 sp 寄存器指向的位置。请在表中填入当 F2 函数首次调用 F3 前栈内保存的可能内容，并在每行的括号内标注该值是被哪个函数所保存的。第一行的内容已经给出。（可根据需要增删行数）

ra (F1)
t0 (F1)
s0 (F1)
ra (F2)
t0 (F2)
t1 (F2)
s0 (F2)
s1 (F2)