9. 1）JAL 跳转的地址有+/- 1MB 的范围
2）Bne 指令目标地址由分支指令的地址加上符号位扩展的偏移量组成，范围是 2^13 = 8192 = 8 KB, 为 +/- 4 KB。
3）在 RISC-V 中可以使用一条 lui 指令和一条 jalr 指令的组合完成任意 32 位绝对地址的跳转操作。
RISC-V 中的 lui 指令也可以用来将一个立即数左移 20 位，然后存储到目标寄存器的高 20 位。再向寄存器中加上低 12 位，即得到一个 32 位的地址。
因此，要实现任意 32 位绝对地址的跳转，可以使用以下代码：

```
lui t0, %hi(addr)
addi t0, t0, %lo(addr)
jalr zero, t0, 0
```

其中，%hi(addr)和%lo(addr)分别返回该地址的高 20 位和低 12 位，可以通过特殊的汇编伪指令得到。这段代码会将%hi(addr)左移 20 位，然后与%lo(addr)进行加法运算，得到完整的 32 位地址。然后，它将跳转到 t0 寄存器中存储的地址，从而实现跳转操作。

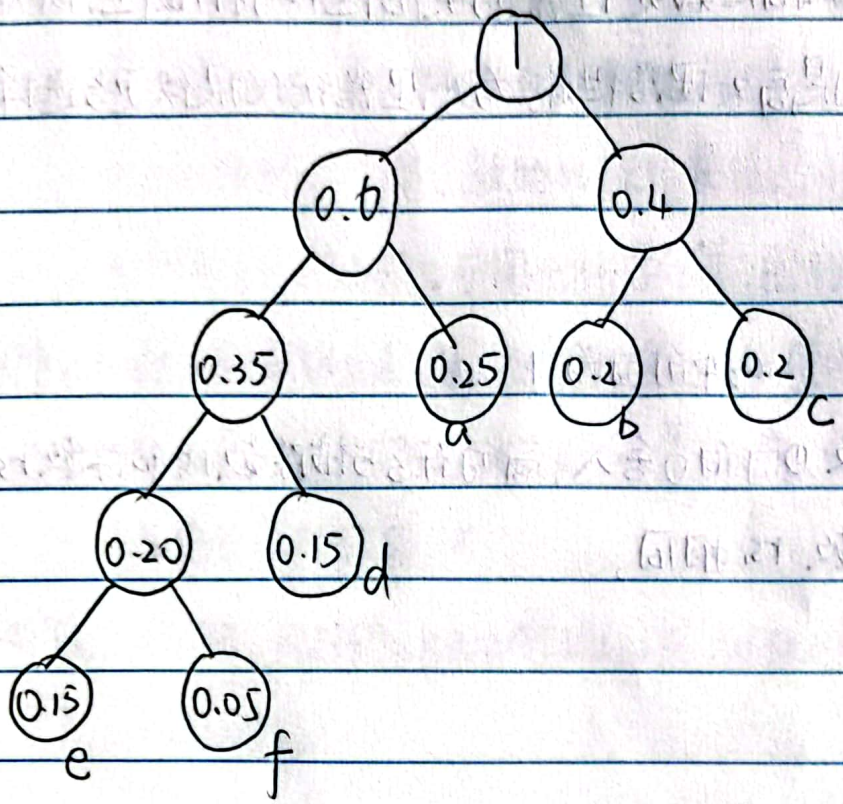
10.（1）常用指令：只有出现频率较高的指令才适合压缩，因为只有这些指令的压缩效果比较显著。
（2）操作数限制：被压缩的指令必须只涉及到一组寄存器或一个立即数，并且不包含 64 位整数操作。
（3）寻址方式限制：被压缩的指令不能使用复杂的寻址方式，如基址加变址寻址等。
（4）指令格式匹配：被压缩的指令必须符合 RVC 的指令格式，例如不能包含跳转指令。
RVC 指令集共有三种类型的指令格式，分别为 L 格式、S 格式和 B 格式，用于加载/存储、移位/逻辑运算和分支等操作。B 格式指令用于分支时只能使用其中的两个寄存器 x1 和 x5，而 L 格式和 S 格式的指令可以使用所有 32 个通用整型寄存器。

19. 1)在函数调用时，程序需要为每个被调用的函数分配一段内存空间来保存该函数内部的局部变量、参数和返回地址等信息。这些信息以栈的形式存储在内存中。

当函数嵌套调用层数过多时，就会产生大量的函数调用栈，这些栈会依次压入堆栈中，占用系统内存空间。如果函数递归调用层数过深，那么每次递归调用都会向调用栈中压入一层栈帧，最终可能导致栈溢出，即栈空间不足，无法继续向栈中压入新的栈帧。

当栈溢出发生时，可能会导致程序崩溃或异常终止，因为操作系统会检测到这种情况，并向应用程序发送一个 SIGSEGV 信号（segmentation violation），以提示程序已经访问了非法地址或越界访问了某个内存区域。
2)①增加栈的大小：可以通过修改操作系统中的默认栈大小来增加栈的大小，从而避免因为递归调用太深等原因导致的栈溢出问题。
②优化递归算法：对于递归调用比较深的算法，可以考虑改写为非递归算法，或者采用尾递归等方式来降低栈的深度，从而避免栈溢出。
③使用动态分配的内存：当需要在函数中创建大量的数组或其他数据结构，可以考虑使用堆内存（如 malloc，在 C 语言中）来替代栈内存，从而避免栈溢出。
④检查输入参数：应该对所有传入函数的参数进行检查，确保它们的类型、值域等都符合预期，避免造成无限递归或其他导致栈溢出的错误。
⑤合理设计程序架构：对于需要频繁使用的函数或变量，将其放在全局变量区域或静态数据区域，避免使用栈内存。

18、



a_i	P_i	l_i	$c(a_i)$
a	0.25	2	01
b	0.20	2	10
c	0.20	2	11
d	0.15	3	001
e	0.15	4	0000
f	0.05	4	0001

平均长度

$\sum_{i=1}^{7} P_i l_i = 2.55$

熵

$H = -\sum_{i=1}^{n} P_i \cdot \log_2 P_i = 2.4660$

信息冗余度

$R = 1 - \frac{H}{2.55} = 0.03296$

20. 第一行：ra (F1 保存)

第二行：s0 (F1 保存)

第三行：t0 (F1 保存)

第四行：t1 (F1 保存)

第五行：a0 (F2 保存)

第六行：(F2 的第 2 个参数) (F1 保存)

第七行：(F2 的第 1 个参数) (F1 保存)

第八行：ra (F1 保存)

第九行：s0 (F2 保存)

第十行：s1 (F2 保存)

第十一行：ra (F2 保存)