2-9.

(1) 范围为 $[-2^{19}, 2^{19}-1]$

(2) 范围为 $[-2^{12}, 2^{12}-1]$

(3) 可以. 先用 lui 指令将绝对地址的前20位按高位存入某一寄存器，然后在 jalr 指令中再使用后12位作为立即数. 最终 jalr 指令会将寄存器的值与立即数相加得到完整的绝对地址并跳转.

2-10. 条件：① 立即数或地址偏移量较小

② 指令使用的某个寄存器是 x0、x1、x2.

③ 目标寄存器与第一个源寄存器相同.

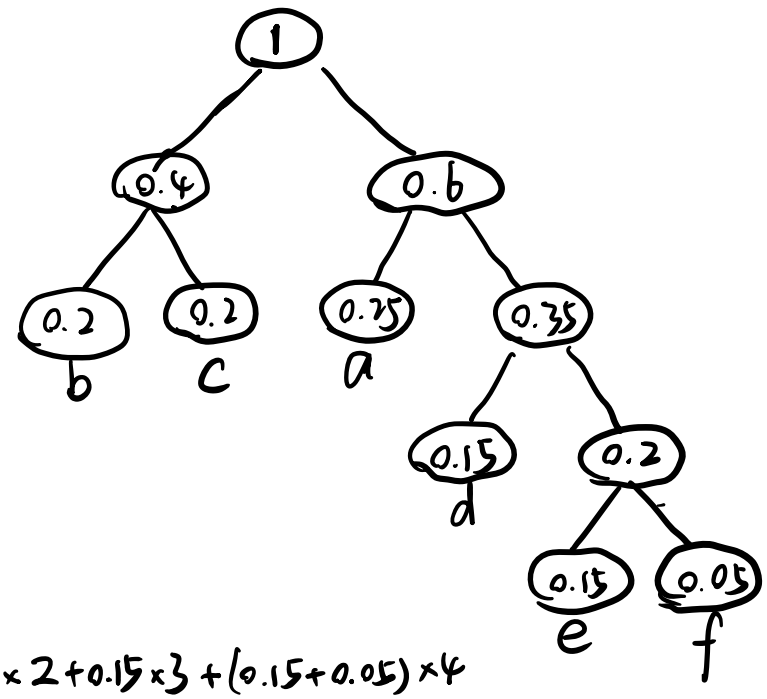
④ 使用的寄存器是8个最常用的：x8 - x15

} 满足其中一个条件即可.

不是所有RVC指令都可以使用32个完整通用寄存器. CR、CI、CSS 可以使用全部，CIW、CL、CS、CA、CB 只能使用 8个：x8 - x15.

2-18. 霍夫曼树如右图所示：

a_i	P_i	l_i	opcode
a	0.25	2	00
b	0.20	2	01
c	0.20	2	10
d	0.15	3	110
e	0.15	4	1110
f	0.05	4	1111



操作码平均长度

$$\bar{L} = \sum_i P_i \cdot l_i = (0.25+0.2+0.2)\times 2 + 0.15\times 3 + (0.15+0.05)\times 4$$

$$= 2.55$$

信息冗余度 $R = 1 - \frac{-\sum P_i \log_2 P_i}{2.55} = 0.033$

2-19. (1) 因为每次调用一个函数，都要为该函数分配相应的空间用作栈，以储存相关寄存器的值，使得它们在函数运行过程中不会丢失。如果函数嵌套调用层数过多，开辟栈所需要的空间也会很多，且它们在最后一层函数执行完之前都不会释放，因此可能会出现内存空间不足，导致栈溢出等情况。

(2) ①减少函数的嵌套调用层数，优化代码逻辑。

②尽量避免递归函数的使用。

2-20.

ra (F1)
$s0(F_1)$
$a0(F_1)$
$t0(F_1)$
$t1(F_1)$
$ra(F_2)$
$s0(F_2)$
$a0(F_2)$
$a1(F_2)$