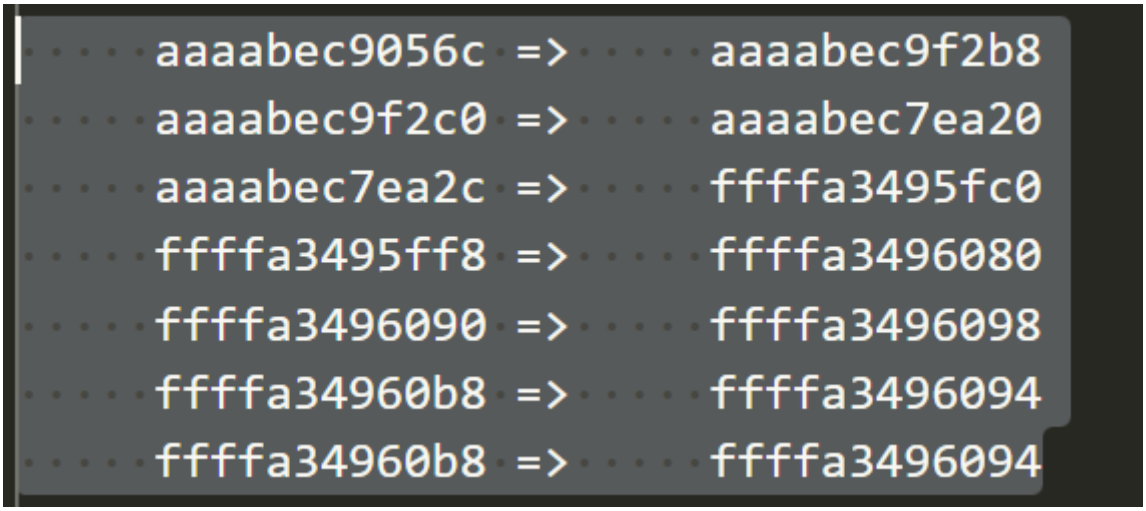

程序的输入将是以上这种形式的一对地址，左边为源地址，右边是跳转的目的地址。以上列出的是程序控制流中的跳转地址，我们需要据此完成整个指令流的恢复。

效果大概入下图

```
    aaaabec9056c
        jmp
    aaaabec9f2b8
    aaaabec9f2bc
    aaaabec9f2c0
        jmp
    aaaabec7ea20
    aaaabec7ea24
    aaaabec7ea28
    aaaabec7ea2c
        jmp
    ffffa3495fc0
    ffffa3495fc4
    ffffa3495fc8
    ffffa3495fcc
    ffffa3495fd0
    ffffa3495fd4
    ffffa3495fd8
    ffffa3495fdc
    ffffa3495fe0
    ffffa3495fe4
    ffffa3495fe8
    ffffa3495fec
    ffffa3495ff0
    ffffa3495ff4
    ffffa3495ff8
        jmp
    ffffa3496080
    ffffa3496084
    ffffa3496088
    ffffa349608c
    ffffa3496090
        jmp
    ffffa3496098
```

```
    ffffa349609c
    ffffa34960a0
    ffffa34960a4
    ffffa34960a8
    ffffa34960ac
    ffffa34960b0
    ffffa34960b4
    ffffa34960b8
        jmp
    ffffa3496094
    ffffa3496098
    ffffa349609c
    ffffa34960a0
    ffffa34960a4
    ffffa34960a8
    ffffa34960ac
    ffffa34960b0
    ffffa34960b4
    ffffa34960b8
        jmp
    ffffa3496094
```

`jmp`代表下行地址是从上行地址跳转过来的，只是作演示方便，在实际输出中**不需要写。**

注意：

- 输入的地址对已经提前存入了数组中，具体使用方法请询问`黄旋`。
- 输出的长度不定，可以采取链表来存储完整控制流，最好是在链表的每个node里分配一块连续地址（数组），以提高效率。