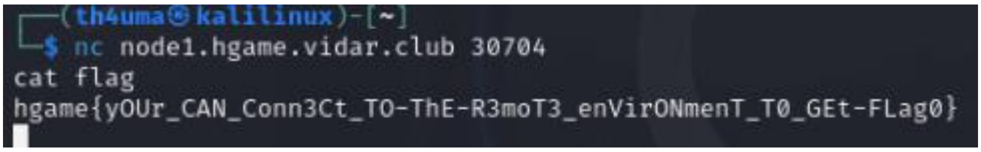
Th4uma #0000db write up

TEST NC

Kali 输入 nc node1.hgame.vidar.club 30704

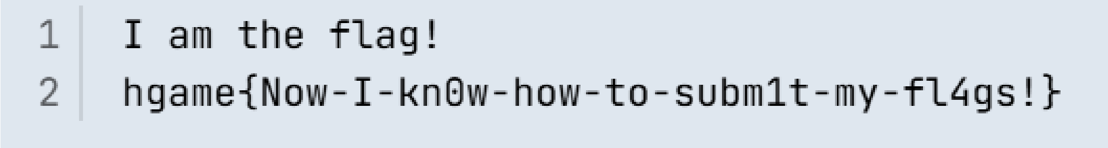
cat flag



```
┌──(th4uma㉿kalilinux)-[~]
└─$ nc node1.hgame.vidar.club 30704
cat flag
hgame{yOUr_CAN_Conn3Ct_TO-ThE-R3moT3_enVirONmenT_T0_GEt-FLag0}
```

获得 flag

从这里开始的序章。

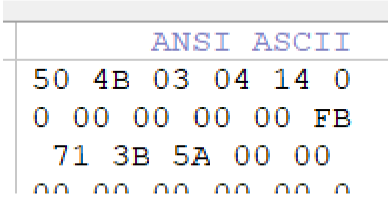


```
I am the flag!
hgame{Now-I-kn0w-how-to-subm1t-my-fl4gs!}
```

复制粘贴 flag

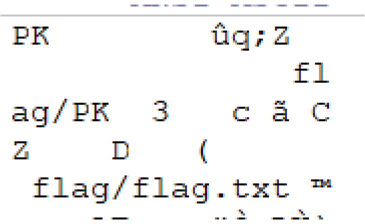
Hakuya Want A Girl Friend

hky.txt 放进 winhex，ascii 显示头文件是 zip 的头文件 16 进制

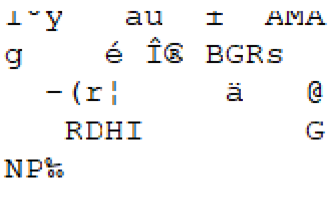


```
      ANSI ASCII
 50 4B 03 04 14 0
 0 00 00 00 00 FB
  71 3B 5A 00 00
```

把 ascii 导出文件再放入 winhex，发现里面有一个 zip 一个倒置 png



```
PK          ûq;Z
                fl
ag/PK  3    c ã C
Z    D    (
 flag/flag.txt ™
```



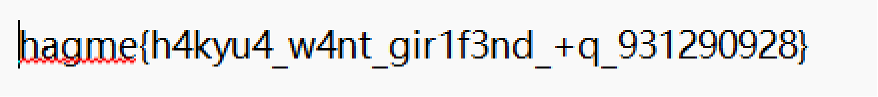
```
1˜y   au   ±   AMA
g    é Î® BGRs
  -(r¦      ä   @
   RDHI         G
NP‰
```

分别导出文件，发现 zip 文件需要密码，修复 png 获得

010editer 修改高度，发现 zip 密码

To_f1nd_th3_QQ

解密后得 flag



hagme{h4kyu4_w4nt_gir1f3nd_+q_931290928}

Compress dot new

附件解压后获得代码，分析可知这是一个霍夫曼编码

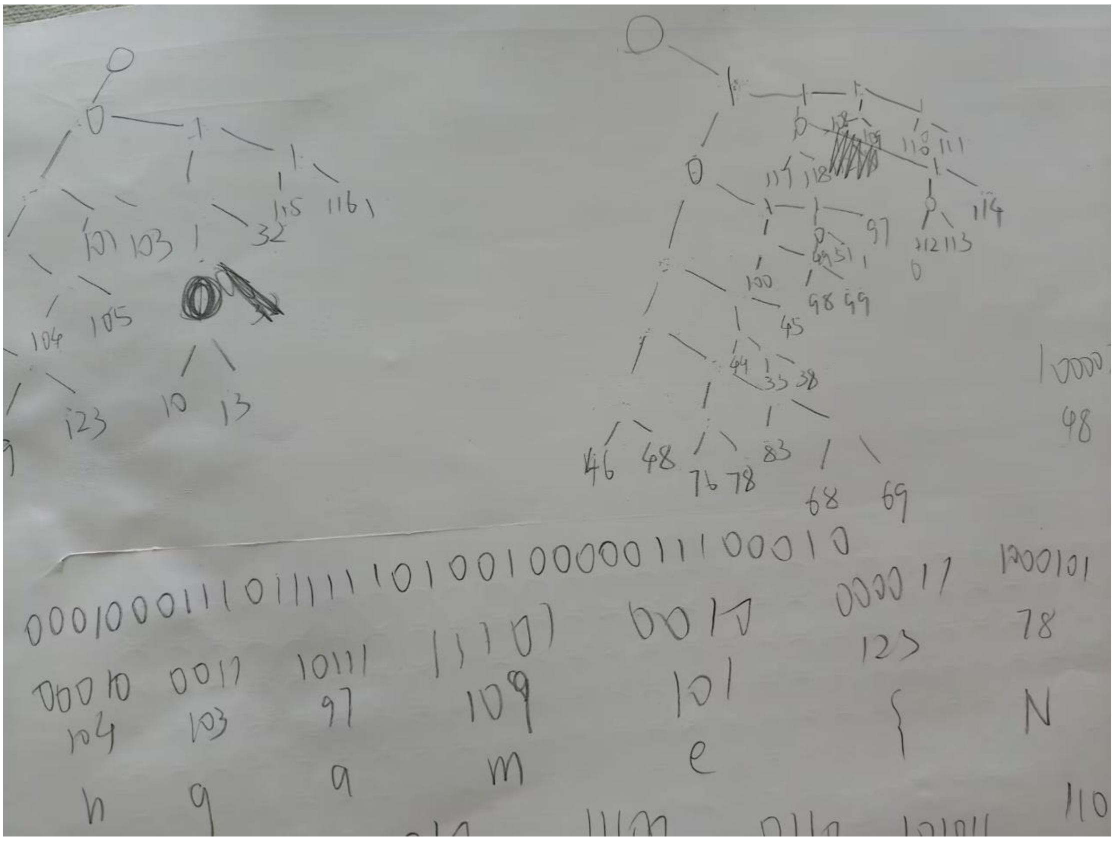
```
def "into b" [] {let arg = $in;0..(( $arg|length ) - 1)|each {|i|$arg|bytes at $i..$i|into int}};def gss [] {match $in {{s:
$s,w:$w} => [$s],{a:$a,b:$b,ss:$ss,w:$w} => $ss}};def gw [] {match $in {{s:$s,w:$w} => $w,{a:$a,b:$b,ss:$ss,w:$w}
=> $w}};def oi [v] {match $in {[] => [$v],[$h,..$t] => {if $v.w < $h.w {[$v,$h] ++ $t} else {[$h] ++ ($t|oi $v)}}}};def h
[] {match $in {[] => [],[$n] => $n,[$f,$sn,..$r] => {$r|oi {a:$f,b:$sn,ss:(($f|gss) ++ ($sn|gss)),w:(($f|gw) +
($sn|gw))}|h}}};def gc [] {def t [nd, pth, cd] {match $nd {{s:$s,w:$_} => ($cd|append {s:$s,c:$pth}),{a:$a,b:$b,ss:
$_,w:$_} => {t $b ($pth|append 1) (t $a ($pth|append 0) $cd)}}};t $in [] []|each {|e|{s:$e.s,cs:($e.c|each {|c|$c|into
string}|str join)}}};def sk [] {match $in {null => null,{s:$s,w:$_} => {s:$s},{a:$a,b:$b,ss:$_,w:$_} =>
{a:($a|sk),b:($b|sk)}}};def bf [] {$in|into b|reduce -f (0..255|reduce -f [] {|i,a|$a|append 0}) {|b,a|$a|update $b
(($a|get $b) + 1)}|enumerate|filter {|e|$e.item > 0}|each {|e|{s:$e.index,w:$e.item}}};def enc [cd] {$in|into b|each
{|b|$cd|filter {|e|$e.s == $b}|first|get "cs"}|str join};def compress []: binary -> string {let t = $in|bf|h;[($t|sk|to
json --raw), ($in|enc ($t|gc))]|str join "\n"}

# source compress.nu; open ./flag.txt --raw | into binary | compress | save enc.txt
```

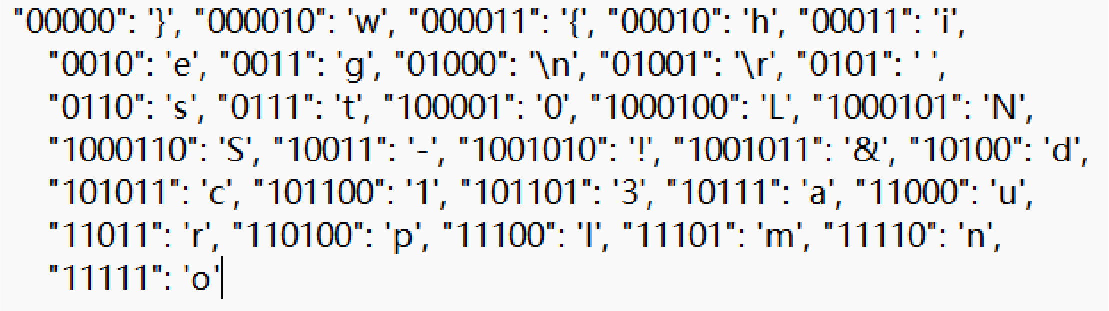
另一个文件里面有霍夫曼树的值，和二进制数值

{"a":{"a":{"a":{"a":{"a":{"s":125},"b":{"a":{"s":119},"b":{"s":123}}},"b":{"a":{"s":104},"b":{"s":105}}},"b":{"a":{"s":101},"
b":{"s":103}}},"b":{"a":{"a":{"a":{"s":10},"b":{"s":13}},"b":{"s":32}},"b":{"a":{"s":115},"b":{"s":116}}}},"b":{"a":{"a":{"a":
{"a":{"a":{"s":46},"b":{"s":48}},"b":{"a":{"a":{"s":76},"b":{"s":78}},"b":{"a":{"s":83},"b":{"a":{"s":68},"b":{"s":69}}}}},"b":
{"a":{"a":{"s":44},"b":{"a":{"s":33},"b":{"s":38}}},"b":{"s":45}}},"b":{"a":{"a":{"s":100},"b":{"a":{"s":98},"b":{"s":99}}},"b
":{"a":{"a":{"s":49},"b":{"s":51}},"b":{"s":97}}}},"b":{"a":{"a":{"a":{"s":117},"b":{"s":118}},"b":{"a":{"a":{"s":112},"b":{"s
":113}},"b":{"s":114}}},"b":{"a":{"a":{"s":108},"b":{"s":109}},"b":{"a":{"s":110},"b":{"s":111}}}}}}
0001000111011111101001000001110001011100010011100011000010001011100111001001101101010111011101
1001101000111011010011101111101110110110011101100111100111101101110111011010110011110110011100
0111001101111000011001100001011011101100011100101001110010111001111000011000101001010000000001001
0100010001001111111011001011101010100011110100011011000111010101101001111111100111111101101010101
10000110111010110111111010010011110010001011010111111111110011000101010110111001001111100011011
010110111101000001111010000011011010101100011111100011010100101110000011011110000000100101000100
01011100011100111001011101011111000101010110101111000001100111100011100101110101111100010110101
11000001010000001011000111101110001110111111010101001001110101110010001111001001011011110111011
10101111101100011110101011100100010111001001011100010110101000011101010001011110101001100011101
01011101100011011011000011010000000101100011101111111110001010101100000

不会编程，就手写了

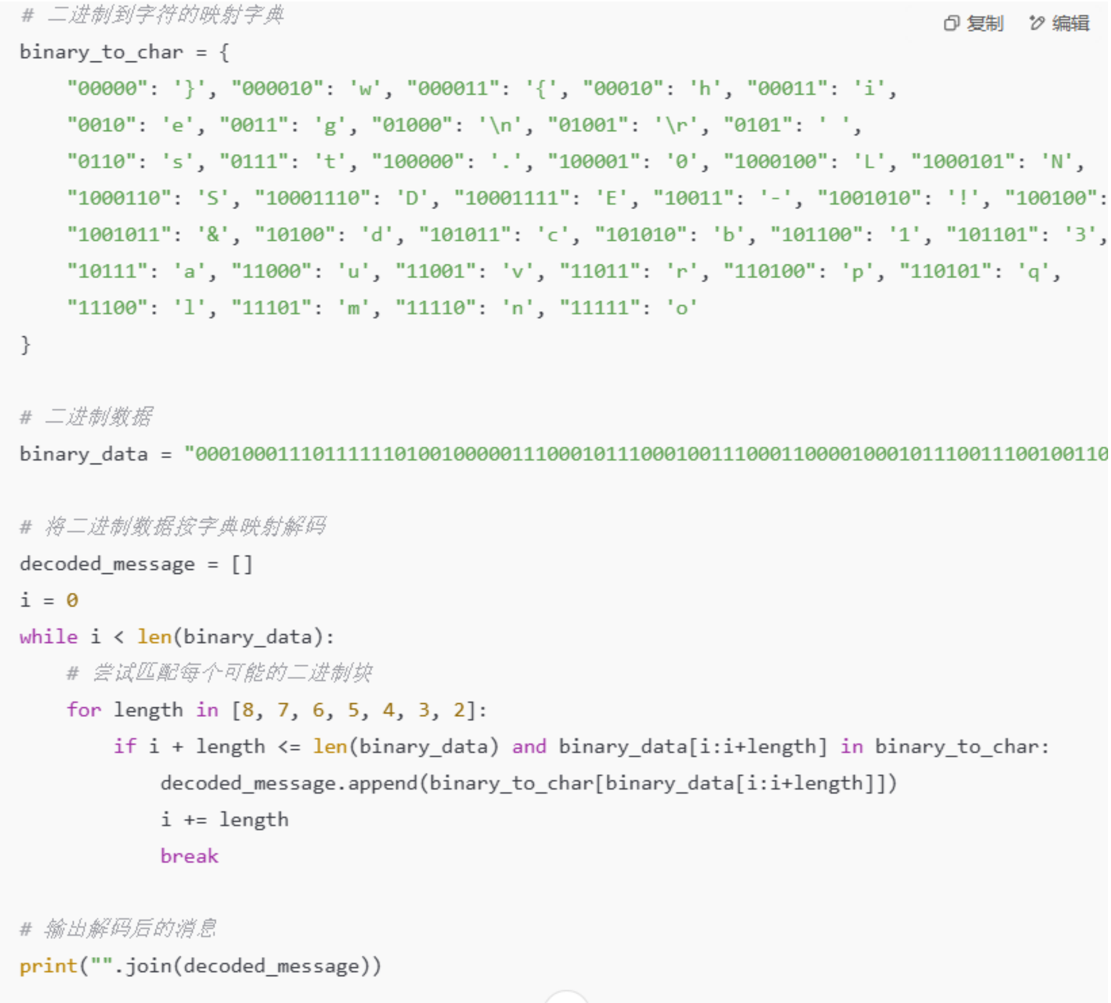


整理出字典



```
"00000": '}', "000010": 'w', "000011": '{', "00010": 'h', "00011": 'i',
   "0010": 'e', "0011": 'g', "01000": '\n', "01001": '\r', "0101": ' ',
   "0110": 's', "0111": 't', "100001": '0', "1000100": 'L', "1000101": 'N',
   "1000110": 'S', "10011": '-', "1001010": '!', "1001011": '&', "10100": 'd',
   "101011": 'c', "101100": '1', "101101": '3', "10111": 'a', "11000": 'u',
   "11011": 'r', "110100": 'p', "11100": 'l', "11101": 'm', "11110": 'n',
   "11111": 'o'
```

不会写代码，让 chatgpt 写的



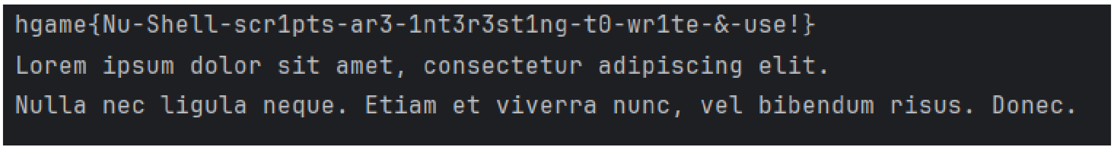
```python
# 二进制到字符的映射字典
binary_to_char = {
    "00000": '}', "000010": 'w', "000011": '{', "00010": 'h', "00011": 'i',
    "0010": 'e', "0011": 'g', "01000": '\n', "01001": '\r', "0101": ' ',
    "0110": 's', "0111": 't', "100000": '.', "100001": '0', "1000100": 'L', "1000101": 'N',
    "1000110": 'S', "10001110": 'D', "10001111": 'E', "10011": '-', "1001010": '!', "100100":
    "1001011": '&', "10100": 'd', "101011": 'c', "101010": 'b', "101100": '1', "101101": '3',
    "10111": 'a', "11000": 'u', "11001": 'v', "11011": 'r', "110100": 'p', "110101": 'q',
    "11100": 'l', "11101": 'm', "11110": 'n', "11111": 'o'
}

# 二进制数据
binary_data = "000100011101111110100100000111000101110001001110001100001000101110011100100110

# 将二进制数据按字典映射解码
decoded_message = []
i = 0
while i < len(binary_data):
    # 尝试匹配每个可能的二进制块
    for length in [8, 7, 6, 5, 4, 3, 2]:
        if i + length <= len(binary_data) and binary_data[i:i+length] in binary_to_char:
            decoded_message.append(binary_to_char[binary_data[i:i+length]])
            i += length
            break

# 输出解码后的消息
print("".join(decoded_message))
```

运行得 flag

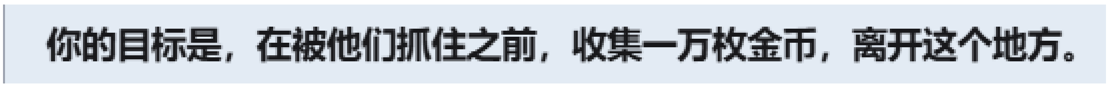


```
hgame{Nu-Shell-scr1pts-ar3-1nt3r3st1ng-t0-wr1te-&-use!}
Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Nulla nec ligula neque. Etiam et viverra nunc, vel bibendum risus. Donec.
```

Level 24 Pacman

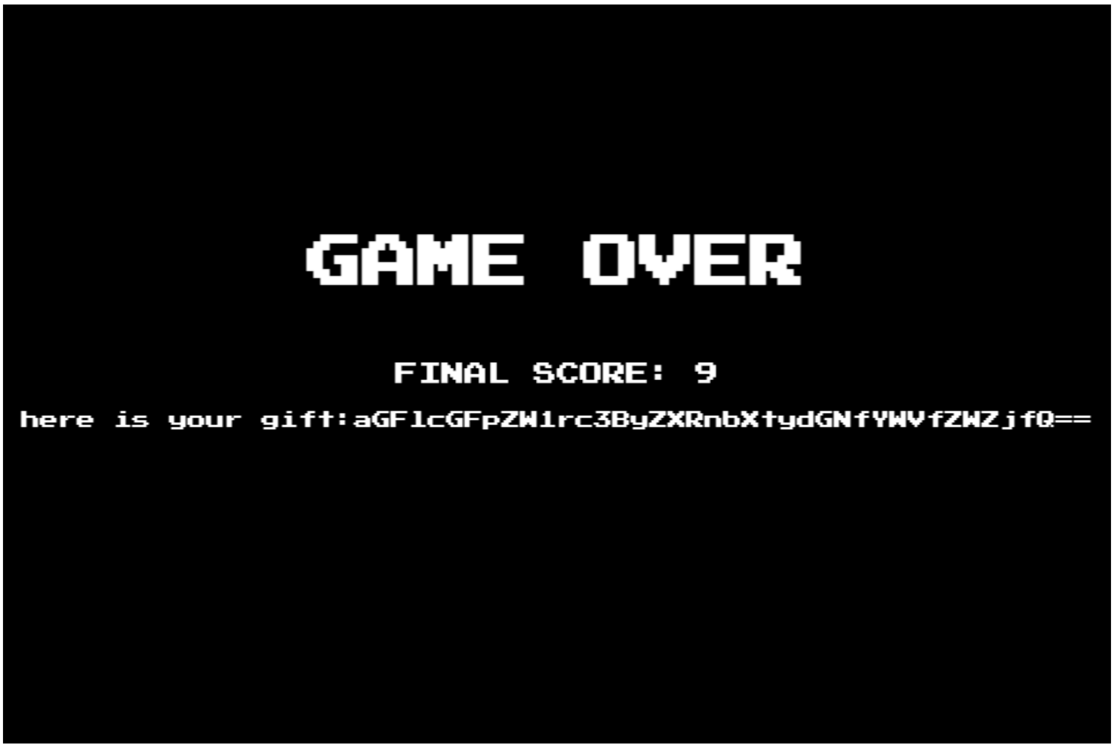
访问 node1.hgame.vidar.club:32306，发现吃豆人小游戏

题目说明分数要达到 10000



> **你的目标是，在被他们抓住之前，收集一万枚金币，离开这个地方。**

先尝试正常玩，死了获得一串代码



GAME OVER

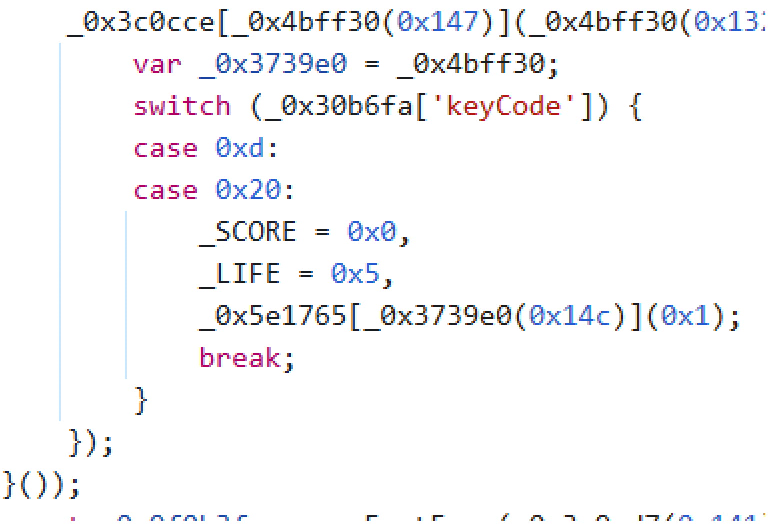
FINAL SCORE: 9

here is your gift:aGFlcGFpZW1rc3ByZXRnbXtydGNfYWVfZWZjfQ==

一眼 base64，解码得 haepaiemkspretgm{rtc_ae_efc}，栅栏密码，解码 hgame{pratice_makes_perfect}

提交显示错误

F12 分析网站源码，发现这一段是判断分数和生命的



```
    _0x3c0cce[_0x4bff30(0x147)](_0x4bff30(0x13
        var _0x3739e0 = _0x4bff30;
        switch (_0x30b6fa['keyCode']) {
        case 0xd:
        case 0x20:
            _SCORE = 0x0,
            _LIFE = 0x5,
            _0x5e1765[_0x3739e0(0x14c)](0x1);
            break;
        }
    });
}());
```

开始游戏后输入

_SCORE = 10000;

获得 flag hgame{u_4re_pacman_m4ster}