1. 解压缩时的字符串和编码的映射放入字典中（key，value）。Key是字符串，value是对应的代码。仍然将单个字符的给予字典来初始化，如（0，a）（1，b）
2. 读入当前代码p，p只有在或不在字典中2种情况
3. 第一种：p在字典中，假设q是p之前的代码。能走到p处，说明前面我们都已经解压缩了，q是在字典中的。q、p都在字典中，我们想一下在编码时，当我们走到text（q）处，接下来要读入下一个字符时，发现text（p）的第一个字符是不在字典中才输出p的吧，如果还在编码中还要继续往下读的。然后在从text（p）的第一个字符开始读，读到text（p）的结尾，对吧。说明qp都有相应的数对，我们现在遇到了字典中没有的数对（解码时还没有的数对，不是编码），且新数对是（下一个代码，text（q）+text（p）的第一个字符），而且在编码过程中这个数对是肯定有的。
4. 第二种：p不在字典中。压缩文件是qp的顺序。q有相应的text（q），但是p没有相应的text（p）。压缩的时候，我读入text（q），要读text（p）的第一个字符时发现字典中没有，我先输出q，再从text（q）第一个字符开始读，那么text（q）和text（p）的第一个字符是不是要给他编码，没错吧，而且这个编码是以前都text（q）以前都没有出现过的吧。我现在从text（p）开始读，读到text（p）结尾，编码的时候是不是只有从以前在编码的时候已有的字典中找，text（p）的代码和以前的都不一样，是新出现的，那么是不是就是刚刚放入字典的数对，那么我text（p）是不是就是text（q）和text（q）的第一个字符。那么原来的文本是text（q）text（p）=text（q）text（q）first\_char\_of\_text（q）。最后，在解码字典中插入（p，text（q）first\_char\_of\_text（q））
5. 看例子：S编码是0214537

（其原来的字符串是S=aaabbbbbbaabaaba，我们现在还不知道，而且也不知道其字典）

1. 先把单个字符的分配代码，0的代码是a，1的代码是b。（实际编码的时候用整个ASCII码就行，0-255都用上，举例：65的代码是’A’）

7、下表是读取S=aaabbbbbbaabaaba的过程，每一次读取一个代码，直到读到空

|  |  |  |
| --- | --- | --- |
| 输入 | 输出（实际中输出的是相应的代码，我放在括号里） | 新加入字典中的数对 |
| 0 | text(0)->a |  |
| 2 | text（0）first\_char\_of\_text(0)->aa | （2，aa压入字典） |
| 1 | text(1)->b | （3，aab压入字典） |
| 4 | text(1)first\_char\_of\_text(1)->bb | （4，bb压入字典） |
| 5 | text(4)first\_char\_of\_text(4)->bbb | （5，bbb压入字典） |
| 3 | text(3)->aab | （6，bbba压入字典） |
| 7 | text(3)first\_char\_of\_text(3) ->aaba | （7，aaba压入字典） |

所以输出aaabbbbbbaabaaba

还有一点要注意的是，程序中的map的（key，value）是q的代码和当前代码p对应的text（p）的最后一个字符，我输出的时候先输出text（p）的时候，先输出text（p）最后一个字符，在追寻着q的代码找到前面一个字符，如此下去。