[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
H03M 7/30

[12] 发明专利说明书

[21] ZL 专利号 97197455.1

[45] 授权公告日 2004 年 4 月 7 日

[11] 授权公告号 CN 1145264C

[22] 申请日 1997.7.23 [21] 申请号 97197455.1

[30] 优先权

[32] 1996. 7.24 [33] US [31] 60/023,094

[32] 1996.12. 3 [33] US [31] 08/753,871

[86] 国际申请 PCT/US1997/012943 1997.7.23

[87] 国际公布 WO98/04045 英 1998.1.29

[85] 进入国家阶段日期 1999.2.24

[71] 专利权人 尤尼西斯公司

地址 美国宾夕法尼亚州

[72] 发明人 泰瑞·A·韦尔奇

阿尔伯特·B·库珀

审查员 谷 威

[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所

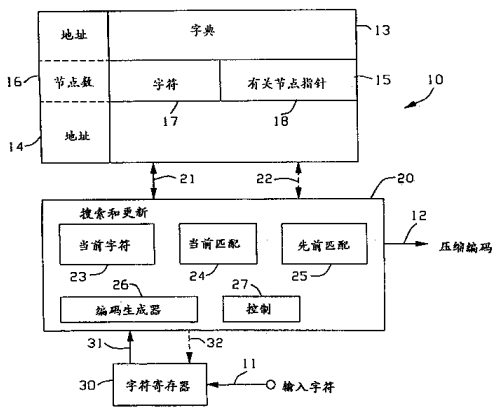
代理人 王以平

权利要求书 10 页 说明书 25 页 附图 16 页

[54] 发明名称 与串搜索交错进行即时字典更新的数据压缩和解压缩系统

[57] 摘要

基于字典的数据压缩和解压缩系统，在压缩装置(10)中，当部分串 W 与字符 C 在字典(13)中匹配时，将以 C 作为串 PW 的扩展字符的新串输入到字典中，其中 P 为与上次输出的压缩编码信号相对应的串。 将一个更新串输入(113)到所读取并且匹配的各输入字符的压缩字典中。 更新是即时的，并与当前串的逐字符匹配交错。 继续更新过程，直至在字典中找到最长匹配。 在串匹配循环中输出(106)最长匹配串的编码。 如果在字典中存在单一字符或多字符串“A”，就将串 AAA…A 编码为两个压缩编码信号，而不考虑串的长度。 以上编码处理将在解压缩装置产生一个未识别的编码信号。 解压缩装置(40)响应于未识别的编码信号，根据与先前接收的编码信号相对应的恢复串(161)，未识别的编码信号，解压缩装置的扩展编码以及先前恢复串中的字符数目(135)，将更新串输入到解压缩字典(43)中(图 8)。



ISSN1008-4274

1.用于将数据字符流(11)压缩为压缩编码流(12)的数据压缩装置(10),其特征在于:

用于存储数据字符串的存储装置(13),每个所述串具有一个与此有关的编码(16),

通过比较所述流与所述存储串搜索所述数据字符流的装置(103-105),以便在确定所述数据字符流与所述存储的串之间的最长匹配之前,进行逐个字符匹配,

用于输出与所述预定匹配有关的编码的装置(106),以便提供所述压缩编码流,

用于在所述逐个字匹配期间将扩展串输入到所述存储装置的装置(107,112-114),在所述逐个字符匹配期间,当匹配每个数据字符时,所述扩展串依次包括一个与利用各数据字符扩展的上次输出的编码相对应的先前最长匹配串,所述扩展串的输入与所述逐个字符匹配的数据字符匹配交错进行,以便当匹配每个数据字符时并且在匹配下一个数据字符之前,将一个扩展串输入到所述存储装置,为每个匹配数据字符输入一个扩展串,在确定最长匹配之前,所述扩展串的输入连续进行,以及

用于将有关编码指派给所述扩展串的装置(112).

2.根据权利要求1的装置,其中所述装置在连续串匹配循环中运行,在所述连续循环中确定有关的最长匹配串,当前循环跟随先前循环,

所述先前匹配串在所述先前循环中与在所述先前循环中提供的所述上次提供的编码匹配,

所述逐字符匹配出现在所述当前循环中,在所述当前循环中,利用所述每个数据字符扩展所述先前匹配串。

3.根据权利要求2的装置,其中所述搜索装置和所述输入装置用于在当部分串W与数据字符C匹配时,将一个所述扩展串输入到所述存储装置中,所述数据字符C作为串PW的扩展字符,其中P为所述先前匹配串,

W位于所述当前循环中的匹配处理过程。

4.根据权利要求2的装置，其中所述搜索装置能够在所述逐字符匹配期间，通过确定数据字符何时失配而确定何时到达所述最长匹配，

所述搜索装置包括利用所述失配数据字符开始下一个串匹配循环的装置(110)。

5.根据权利要求1的装置还包括：利用具有与其有关的特定编码的所有单一字符串来初始化所述存储装置的装置(110)。

6.根据权利要求1的装置还包括：

输出跟随于一标识的未压缩格式的第一次遇到的数据字符的装置，该标识表示正在输出以上数据字符，以及

以单一字符串的形式，将第一次遇到的所述数据字符输入到所述存储装置的装置。

7.根据权利要求6的装置，其中所述标识包括一个0编码。

8.根据权利要求2的装置，其中以链接树结构的方式将所述串存储到所述存储装置中，并且所述数据字符流包括一个重复字符串，重复字符串包括一个重复字符，

所述装置用于将所述重复字符串压缩为两个压缩编码，而不考虑其长度。

9.根据权利要求8的装置，其中

所述先前匹配串包括所述重复字符，

所述搜索装置用于经过所述树沿某条路径匹配所述重复字符串，

所述输入装置用于沿所述路径输入所述扩展串，

从而将所述重复字符串压缩为两个压缩编码，而不考虑其长度。

10.根据权利要求2的装置，其中以链接树结构的方式将所述串存储到所述存储装置中，并且所述数据字符流包括一个重复字符组串，重复字符组串包括一个重复字符组，

所述装置用于将所述重复字符组串压缩为两个压缩编码，而不考虑其长度。

11.根据权利要求10的装置，其中

所述先前匹配串包括所述重复字符组，

所述搜索装置用于经过所述树沿某条路径匹配所述重复字符组串，

所述输入装置用于沿所述路径输入所述扩展串，

从而将所述重复字符组串压缩为两个压缩编码，而不考虑其长度。

12.响应于数据字符流接收数据压缩装置提供的压缩编码流(41)的数据解压缩装置(40)，所述装置用于根据所述压缩编码流恢复所述数据字符流(42)，其特征在于:

用于存储数据字符串的存储装置(43)，每个所述串具有一个与此有关的编码(46)，

利用当前接收的压缩编码访问所述存储装置的装置(132-137，144，145)，用于从与所述当前接收的压缩编码相对应的所述存储装置中恢复串，从而恢复其数据字符，

用于输入到所述存储装置的装置(140-146)，在当前接收的压缩码是经识别的压缩码时工作，扩展串(142)包括与先前接收的压缩编码相对应的该先前恢复串，其中依次利用与所述当前接收的压缩编码相对应的所述恢复串的每个数据字符扩展先前接收的压缩编码，

将有关编码指派到所述扩展串的装置(141)，

用于输入到所述存储装置的其他装置(160-166)，在当前接收的压缩码是未被识别的压缩码时工作，其他扩展串(163)相应于一个未识别的压缩编码的接收，所述其他扩展串包括与所述先前接收的压缩编码相对应的所述先前恢复串，其中依次利用所述先前恢复串的每个数据字符，顺序地重复地扩展所述先前接收的压缩编码，直到输入一个与所述未识别的压缩编码相对应的串，

将有关编码指派到所述其他扩展串的装置(162)，以及

提供与所述未识别的压缩编码相对应的所述串的数据字符的装置(167)，以便恢复所述数据字符。

13.根据权利要求12的装置，其中

所述其他输入装置能够输入(170-175)到所述存储装置，附加扩展串(172)包括与所述未识别的压缩编码相对应的所述串，其中依次利用所述先前恢复串的数据字符，进一步扩展所述未识别的压缩编码，直到将所述附

加扩展串的数目输入到所述存储装置中，所述数目等于包括所述先前恢复串的数据字符的数目，

将有关编码指派到所述附加扩展串的装置(171)。

14.根据权利要求13的装置，其中所述装置在连续串恢复循环中运行，在所述连续循环中恢复有关串，当前循环跟随前一循环，

利用在所述前一循环中接收的所述先前接收的压缩编码，在所述前一循环中恢复所述先前恢复串，

在所述当前循环期间，接收所述当前接收的压缩编码，以及

所述输入装置用于在所述当前循环期间，将所述扩展串输入到所述存储装置中。

15.根据权利要求13的装置，其中在所述当前循环中接收所述未识别的压缩编码，

所述其他输入装置用于在所述当前循环期间，将所述其他扩展串和所述附加扩展串输入到所述存储装置中。

16.根据权利要求12的装置还包括：利用具有与其有关的相应编码的所有单一字符串来初始化所述存储装置的装置(130)。

17.根据权利要求12的装置，其中所述压缩编码流包括一个具有未压缩格式的数据字符，跟随于一个表示正在接收该数据字符的标识之后，

具有未压缩格式的所述数据字符，与所述数据压缩装置在所述数据字符流中第一次遇到的数据字符相对应，

所述装置还包括：

以单一字符串的形式，将以未压缩格式接收的所述数据字符输入到所述存储装置的装置，以及

输出以未压缩格式接收的所述数据字符的装置。

18.根据权利要求17的装置，其中所述标识包括一个0编码。

19.根据权利要求15的装置，其中

所述数据字符流包括一个重复字符串，重复字符串包括一个重复字符，所述数据压缩装置用于将所述重复字符串压缩为两个连续压缩编码，以及

以链接树结构的方式，在所述存储装置中存储所述串。

20.根据权利要求19的装置，其中

所述先前接收的压缩编码包括一个第一所述连续压缩编码，所述未识别的压缩编码包括一个第二所述连续压缩编码，

所述先前恢复串包括所述重复字符，并且与所述未识别的压缩编码相对应的所述串包括所述重复字符串，

经过所述树沿某条路径存储所述先前恢复串，以及

所述输入装置和所述其他输入装置用于沿所述路径输入所述其他扩展串和所述附加扩展串。

21.根据权利要求15的装置，其中

所述数据字符流包括一个重复字符组串，重复字符组串包括一个重复字符组，所述数据压缩装置用于将所述重复字符组串压缩为两个连续压缩编码，以及

以链接树结构的方式，在所述存储装置中存储所述串。

22.根据权利要求21的装置，其中

所述先前接收的压缩编码包括一个第一所述连续压缩编码，所述未识别的压缩编码包括一个第二所述连续压缩编码，

所述先前恢复串包括所述重复字符组，并且与所述未识别的压缩编码相对应的所述串包括所述重复字符组串，

经过所述树沿某条路径存储所述先前恢复串，以及

所述输入装置和所述其他输入装置能够沿所述路径输入所述其他扩展串和所述附加扩展串。

23.根据权利要求20的装置，其中所述其他输入装置用于根据所述先前恢复串生成所述其他扩展串，数据字符的数目包括所述先前恢复串，将所述未识别的压缩编码和所述编码指派到所述其他扩展串，一个所述其他扩展串为与所述未识别的压缩编码相对应的串。

24.将数据字符流(103)压缩为压缩编码流(106)的数据压缩方法，其特征在于：

将数据字符串存储到存储装置(13)中，每个所述串具有一个与此有关的编码(16)，

通过比较(104)所述流与所述存储串来搜索(103-105)所述数据字符流，以便在确定所述数据字符流与所述存储的串之间的最长匹配之前，进行逐个字符匹配，

输出(106)与所述最长匹配有关的编码，以便提供所述压缩编码流，

在所述逐个字匹配期间将扩展串输入(107，112-114)到所述存储装置，在所述逐个字符匹配期间，当匹配每个数据字符时，所述扩展串依次包括一个先前最长匹配串，所述匹配串与利用各数据字符扩展的上次输出的编码相对应，所述扩展串的输入与所述逐个字符匹配的数据字符匹配交错进行，以便当匹配每个数据字符时，以及在匹配下一个数据字符之前，将一个扩展串输入到所述存储装置，为每个匹配数据字符输入一个扩展串，在确定最长匹配之前，所述扩展串的输入连续进行，以及

将相应编码指派(112)给所述扩展串。

25.根据权利要求24的方法，其中所述方法在连续串匹配循环中运行，在所述连续循环中确定有关的最长匹配串，当前循环跟随先前循环，

所述先前匹配串在所述先前循环中与在所述先前循环中提供的所述上次提供的编码匹配，

所述逐字符匹配出现在所述当前循环中，在所述当前循环中，利用所述每个数据字符扩展所述先前匹配串。

26.根据权利要求25的方法，其中执行所述搜索步骤和所述输入步骤，从而当部分串W与数据字符C匹配时，将一个所述扩展串输入到所述存储装置中，其中所述数据字符C作为串PW的扩展字符，P为所述先前匹配串， W位于所述当前循环中的匹配处理过程。

27.根据权利要求25的方法，其中所述搜索步骤包括所述逐字符匹配期间，通过确定数据字符何时失配来确定何时达到所述最长匹配，

所述搜索步骤包括利用所述失配数据字符开始(110)下一个串匹配循环。

28.根据权利要求24的方法还包括：利用具有与其有关的特定编码的所有单一字符串来初始化(100)所述存储装置。

29.根据权利要求24的方法还包括：

输出跟随于一标识之后的未压缩格式的第一次遇到的数据字符，该标识表示正在输出以上数据字符，以及

以单一字符串的形式，将第一次遇到的所述数据字符输入到所述存储装置中。

30.根据权利要求29的方法，其中所述标识包括一个0编码。

31.根据权利要求25的方法，其中所述存储步骤包括以链接树结构的方式将所述串存储到所述存储装置中，

所述数据字符流包括一个重复字符串，所述重复字符串包括一个重复字符，

所述方法将所述重复字符串压缩为两个压缩编码，而不考虑其长度。

32.根据权利要求31的方法，其中

所述先前匹配串包括所述重复字符，

所述搜索步骤包括经过所述树沿某条路径匹配所述重复字符串，

所述输入步骤包括沿所述路径输入所述扩展串，

从而将所述重复字符串压缩为两个压缩编码，而不考虑其长度。

33.根据权利要求25的装置，其中所述存储步骤包括以链接树结构的方式将所述串存储到所述存储装置中，

所述数据字符流包括一个重复字符组串，所述重复字符组串包括一个重复字符组，

所述方法将所述重复字符组串压缩为两个压缩编码，而不考虑其长度。

34.根据权利要求33的方法，其中

所述先前匹配串包括所述重复字符组，

所述搜索步骤包括经过所述树沿某条路径匹配所述重复字符组串，

所述输入步骤包括沿所述路径输入所述扩展串，

从而将所述重复字符组串压缩为两个压缩编码，而不考虑其长度。

35.响应于数据字符流，接收数据压缩装置提供的压缩编码流(132)的数据解压缩方法(图7)，所述方法用于根据所述压缩编码流，恢复所述数据字符流(137)，其特征在于

在存储装置(43)中存储数据字符串，每个所述串具有一个与此有关的编

码(46),

利用当前接收的压缩编码访问(132-137，144，145)所述存储装置，用于从与所述当前接收的压缩编码相对应的所述存储装置中恢复串，从而恢复其数据字符,

输入(140-146)到所述存储装置，扩展串包括与该先前接收的压缩编码相对应的该先前恢复串，其中依次利用与所述当前接收的压缩编码相对应的所述恢复串的每个数据字符扩展所述先前接收的压缩编码,

将有关编码指派(141)到所述扩展串,

进一步输入(160-166)到所述存储装置，其他扩展串(163)响应于一个未识别的压缩编码的接收，所述其他扩展串包括与所述先前接收的压缩编码相对应的所述先前恢复串，其中依次利用所述先前恢复串的每个数据字符，顺序重复扩展所述先前接收的压缩编码，直到输入一个与所述未识别的压缩编码相对应的串,

将有关编码指派(162)到所述其他扩展串，以及

提供(167)与所述未识别的压缩编码相对应的所述串的数据字符，以便恢复所述数据字符。

36.根据权利要求35的方法,其中所述其他输入步骤包括输入(170-175)到所述存储装置，附加扩展串(172)包括与所述未识别的压缩编码相对应的所述串，其中依次利用所述先前恢复串的数据字符进一步扩展所述未识别的压缩编码，直到将所述附加扩展串的数目输入到所述存储装置中，所述数目等于包括所述先前恢复串的数据字符的数目,

将有关编码指派(171)到所述附加扩展串。

37.根据权利要求36的方法，其中所述方法在连续串恢复循环中运行,在所述连续循环中恢复相应串，当前循环跟随前一循环,

利用在所述前一循环中接收的所述先前接收的压缩编码在所述前一循环中恢复所述先前恢复串,

在所述当前循环期间，接收所述当前接收的压缩编码，以及

所述输入步骤包括在所述当前循环期间，将所述扩展串输入到所述存储装置中。

38.根据权利要求37的方法，其中在所述当前循环中接收所述未识别的压缩编码，

所述其他输入步骤包括在所述当前循环期间将所述其他扩展串和所述附加扩展串输入到所述存储装置中。

39.根据权利要求35的方法还包括：利用具有与其有关的相应编码的所有单一字符串来初始化(130)所述存储装置。

40.根据权利要求35的方法，其中所述压缩编码流包括一个数据字符，所述数据字符具有未压缩格式，跟随于一个标识之后，所述标识表示正在接收该数据字符，

具有未压缩格式的所述数据字符与所述数据压缩装置在所述数据字符流中第一次遇到的数据字符相对应，

所述方法还包括：

以单一字符串的形式将以未压缩格式接收的所述数据字符输入到所述存储装置中，以及

输出以未压缩格式接收的所述数据字符。

41.根据权利要求40的方法，其中所述标识包括一个0编码。

42.根据权利要求38的方法，其中

所述数据字符流包括一个重复字符串，所述重复字符串包括一个重复字符，所述数据压缩装置用于将所述重复字符串压缩为两个连续压缩编码，以及

所述存储步骤包括以链接树结构的方式，在所述存储装置中存储所述串。

43.根据权利要求42的方法，其中

所述先前接收的压缩编码包括一个第一所述连续压缩编码，所述未识别的压缩编码包括一个第二所述连续压缩编码，

所述先前恢复串包括所述重复字符，并且与所述未识别的压缩编码相对应的所述串包括所述重复字符串，

经过所述树沿某条路径存储所述先前恢复串，以及

所述输入步骤和所述其他输入步骤包括沿所述路径输入所述其他扩展

串和所述附加扩展串。

44.根据权利要求38的方法，其中

所述数据字符流包括一个重复字符组串，所述重复字符组串包括一个重复字符组，所述数据压缩装置用于将所述重复字符组串压缩为两个连续压缩编码，以及

所述存储步骤包括以链接树结构的方式，在所述存储装置中存储所述串。

45.根据权利要求44的方法，其中

所述先前接收的压缩编码包括一个第一所述连续压缩编码，所述未识别的压缩编码包括一个第二所述连续压缩编码，

所述先前恢复串包括所述重复字符组，并且与所述未识别的压缩编码相对应的所述串包括所述重复字符组串，

经过所述树沿某条路径存储所述先前恢复串，以及

所述输入步骤和所述其他输入步骤包括沿所述路径输入所述其他扩展串和所述附加扩展串。

46.根据权利要求35的方法，其中所述其他输入步骤包括根据所述先前恢复串，生成所述其他扩展串，数据字符的数目包括所述先前恢复串，将所述未识别的压缩编码和所述编码指派到所述其他扩展串，一个所述其他扩展串为与所述述未识别的压缩编码相对应的串。

与串搜索交错进行即时字典更新的数据压缩和解压缩系统

技术领域

本发明涉及基于字典的数据压缩和解压缩，具体地涉及更新压缩和解压缩字典所用的方式。

背景技术

称为 LZ2 的 Lempel—Ziv（LZ）算法为用途广泛的基于字典的数据压缩和解压缩系统提供了理论基础。在题目为“通过变速率编码进行单个序列的压缩”一文中描述了 LZ2，作者为 Jacob Ziv 和 Abraham Lempel，出版于 1978 年 9 月的 IEEE 信息理论汇刊第 IT-24 卷第 5 期第 530-536 页。称为 LZW 的一种普遍运用的数据压缩和解压缩系统，被采用为 V.42 双工调制解调器压缩和解压缩的标准，在 Welch 的美国专利 4,558,302 中叙述，发布于 1985 年 12 月 10 日。LZW 还被采纳为 GIF 和 TIFF 图象通信标准中所用的压缩和解压缩方法。LZ2 的一种变化的方式，在 Storer 的美国专利 4,876,541 中叙述，发布于 1989 年 10 月 24 日。叙述基于 LZ 字典的压缩和解压缩系统的更进一步的例子有： Eastman 等的美国专利 4,464,650,发布于 1984 年 8 月 7 日； Miller 等的美国专利 4,817,746，发布于 1989 年 3 月 21 日; Clark 的美国专利 5,153,591,发布于 1992 年 10 月 6 日; 以及 Lempel 等的欧洲专利申请公开，公开号 0 573 208A1，刊登于 1993 年 12 月 8 日。

在上述系统中，输入字符数据流逐个字符地与存储在字典中的字符串相比较，进行匹配。典型地，不断进行逐个字符的比较，直到确定最长的匹配。根据匹配，输出一个压缩码并且利用一个或多个附加字符串更新字典。在 Stover 的专利(541 号)中，通过连接当前最长匹配字符串中的所有非 0 前缀与先前的最长匹配字符串来更新字典。这样，如果在当前最长匹配中有 N 个字符，则在确定当前最长匹配之后，有 N 个字符串被加到字典中。在 Stover 的专利中，它被称为全前缀(AP)更新技术。

另一种数据压缩和解压缩方法被表述为行程长度编码(RLE)。RLE 算法通过提供一个表明字符或字符组的压缩码以及其行程长度来压缩一个重复的字符或字符组行程。RLE 在对相同字符或字符组的长行程进行编码时很有效。例如，RLE 在压缩一个可能包含于数据文件开头的由空格形成的长序列时很有效。在图象压缩中，如果图象中包含一连续的、具有相同值的象素点长行程时，例如在一个陆地—天空图象中的天空部分，RLE 也很有效。

上面讨论的基于 LZ 字典的压缩解压缩算法在压缩重复字符或字符组的长行程时不是特别有效。即使使用 AP 更新技术，也需要大量的压缩码输出值来压缩一个长行程。

传统上，通过把数据应用在一个行程长度编码器，并且把这个行程长度编码数据应用在基于 LZ 字典的系统来克服这种基于字典系统的缺陷。在以上结构中，在基于字典的压缩装置前端运用一个行程长度编码器，并且在基于字典的解压缩装置的输出端运用一个行程长度长度解码器。该系统具有以下缺陷：增加设备、费用、控制开销和处理时间。

英国专利申请 GB 2 277 179 A 公开了处理重复字符行程的另一种方案，公开日期为 1994 年 10 月 19 日。采用传统 LZ 方法压缩输入数据字符流并将压缩编码流解压缩。监视压缩算法所得的输出编码，以确定当前生成的压缩编码是否是由一个行程产生的。如果是这样的话，则禁止压缩装置输出，直到监视表示该行程结束。随后，重新允许输出压缩编码。根据接收的编码，解压缩装置“插入”缺少的编码。

发明内容

本发明提供了一种用于将数据字符流压缩为压缩编码流的数据压缩装置，其特征在于：用于存储数据字符串的存储装置，每个所述串具有一个与此有关的编码，通过比较所述流与所述存储串搜索所述数据字符流的装置，以便在确定所述数据字符流与所述存储的串之间的最长匹配之前，进行逐个字符匹配，用于输出与所述预定匹配有关的编码的装置，以便提供所述压缩编码流，用于在所述逐个字匹配期间将扩展串输入到所述存储装置的装置，在所述逐个字符匹配期间，当匹配每个数据字符时，所述扩展

串依次包括一个与利用各数据字符扩展的上次输出的编码相对应的先前最长匹配串，所述扩展串的输入与所述逐个字符匹配的数据字符匹配交错进行，以便当匹配每个数据字符时并且在匹配下一个数据字符之前，将一个扩展串输入到所述存储装置，为每个匹配数据字符输入一个扩展串，在确定最长匹配之前，所述扩展串的输入连续进行，以及用于将有关编码指派给所述扩展串的装置。

本发明体现于一个克服了上述缺陷的数据压缩和解压缩系统。如果在一个字典中存在一个串 A，于是将串 AAA...A 编码为两个压缩编码符号，而不考虑它的长度。这样，重复字符构成的串，如空格和 0，或者例如具有相同数值的连续象素的字符组，可以在第一次遇到时非常有效地编码。

在本发明的压缩算法中，当每个输入字符被读入和匹配时，一个串被输入到压缩字典中。传统地，当完成最长匹配时，一个更新的字符串或多个字符串被输入到字典中，并确定输出压缩编码符号。算法的操作同下面所述。每一次，一部分串 W 和字符 C 在字典中被找到，一个新的串被输入到字典中，通过 C 作为串 PW 中的一个扩展字符，其中 P 是在上次传送的压缩编码符号中传送的串。这样，当串 W 被匹配之后，当它们在串搜索过程中被匹配时，字符串 P 被字符 W 扩展。这也许会被称作“实时在线”字典更新， 这里，字典更新是即时的并与串搜索过程逐字符交替进行。输入与存储串 W 的逐字符匹配完成以后，每一个匹配字符被附加到不断增长的串 PW 之后。当输入数据字符与字典中的最长串 W 相匹配时，更新过程结束。

当被匹配的串 W 与先前匹配的串 P 相一致时，就实现了上面叙述的行程长度编码的优点。在这种情况下，压缩装置传送一个压缩编码符号，解压缩装置不能认出该符号。解压缩装置运用一个不被认出的编码过程来保持与压缩字典的同步，这个过程基于当前指派的解压缩编码，未认出的编码，先前解码的串和先前解码的串中的字符数。

附图说明

图 1 是用于体现本发明的数据压缩子系统的概要框图。

图 2 是为恢复图 1 中压缩装置的压缩编码输出的一个数据解压缩子系

统的概要框图。

图 3a 是表示图 1 和图 2 中字典搜索树中节点的一个有代表性的数据结构图。

图 3b 是表示图 1 和图 2 中字典搜索树中节点的实际数据结构的示意图。

图 4 是一个节点概要图，按照图 3a 中数据结构来表示图 1 和图 2 中字典搜索树中的一个节点。

图 4a 概要说明搜索树的一部分，表示运用图 4 中节点进行数据存储。

图 5 是一个节点概要图，按照图 3b 的数据结构来表示图 1 和图 2 中字典搜索树中的一个节点。

图 5a 概要说明搜索树的一部分，该搜索树运用图 5 中的节点并存储与图 4a 相同的串。

图 6 是一个控制流程图，表示由图 1 中的压缩子系统按本发明进行压缩操作。图 6 中的流程图基于用全部单一字符串来初始化压缩字典。

图 7 是一个控制流程图，表示用图 2 中的解压缩子系统来执行的操作，目的是解压缩按照图 6 产生的压缩编码。图 7 中的流程图基于用全部单一字符串来初始化解压缩字典。

图 8 是一个控制流程图，表示图 7 和图 10 中的未识别编码处理过程。

图 9 是一个与图 6 相似的控制流程图，但它是基于非初始化的压缩字典。

图 10 是一个与图 7 相似的控制流程图，但它是基于一个非初始化的解压缩字典。图 10 中的解压缩流程图解压缩按照图 9 产生的压缩编码。

图 11a-11e 概要表示搜索树的一部分，表示压缩一个典型的输入数据字符流时，压缩字典的连续状态。

图 12a-12g 概要表示搜索树的一部分，表示输入数据字符流是一个重复的字符组时，压缩字典的连续状态。

具体实施方式

参照图 1，图 1 表示一个数据压缩子系统 10，它把施加在输入端 11 的一个输入数据字符信号流压缩为输出端 12 的一个相应的压缩编码信号流。

用一个诸如 RAM 或 CAM 之类的存储器来起到存储字符串的字典 13 的作用，一般是按照上述参考文献中所描述的方式。字符串存储在一个搜索树数据库结构中，其方式易于理解。搜索树是由存储在字典 13 的存储单元中的相互连接的节点构成。利用众所周知的方法通过地址 14 来访问字典 13 的存储单元。

搜索树节点的数据结构用节点 15 来表示，它包括一个节点数 16，一个字符域 17 和用作相关节点指针的域 18。节点数 16 标志树节点，为了方便，将存储节点 15 的存储器地址 14 用作节点数。字符域 17 用来包含这个节点的数据字符值。域 18 包含一些指针，用易于理解的方式将节点 15 与相关的树节点，如父节点，子节点和兄弟节点等相连。

压缩子系统 10 包括一个搜索和更新控制部分 20，通过一个双向数据总线 21 和一个双向控制总线 22 与字典 13 耦合。搜索和更新控制部分 20 包括表示为当前字符寄存器 23 的工作寄存器，一个当前匹配寄存器 24 和一个先前匹配寄存器 25。搜索和更新控制部分 20 进一步包括一个编码生成器 26，用来为存储在字典 13 中的字符串指派压缩编码值。编码生成器 26 可以顺序或通过诸如散列之类的方法伪随机地指派编码数。被指派的编码通过存储器地址 14 来访问字典 13 的存储单元。这样，容易理解，地址 14(节点数 16)被用作存储在字典 13 中串的压缩编码。

搜索和更新控制部分 20 包括控制 27，控制 27 利用以下将描述的方式，按照图 6 和图 9 中的操作流程图来控制压缩子系统 10 的操作。

压缩子系统 10 包括一个字符寄存器 30，缓存输入端 11 接收到的输入数据字符流。各输入数据字符用将要叙述的方式，通过总线 31 从字符寄存器 30 作用到当前字符寄存器 23 中。搜索和更新控制部分 20 通过一个控制总线 32 来控制从字符寄存器 30 中获得输入数据字符的操作。

压缩子系统 10 的操作简述如下。将输入数据字符连续插入到当前字符寄存器 23 中，并且对照存储在字典 13 中的串进行搜索直到达到与它的最

长匹配。在这个过程中运用当前匹配寄存器 24 。在输出端 12 给出最长匹配串的节点数 16，作为压缩编码。这些搜索操作与上述参考文献中的搜索方式相同。按照本发明，当输入数据字符与存储在字典中的被搜索串相匹配时，通过利用当前输入字符扩展与先前的压缩输出编码相对应的串以便更新字典 13 。在这个过程中运用先前匹配寄存器 25 。当输入字符被继续取出和匹配时，如此被扩展的先前匹配串可用来匹配。这样，用一种与逐个字符的串搜索相互交错的方式将更新串即时添加到字典 13 中。

参考图 2 并继续参考图 1，图 2 表示一个解压缩子系统 40，从压缩子系统 10 的输出端 12 提供的压缩编码信号恢复出原始输入数据流中的字符。相应地，解压缩子系统 40 在一个输入端 41 接收输入压缩编码信号，在一个输出端 42 提供恢复的相应字符串。解压缩子系统 40 包括一个字典 43，它最好是用 RAM 存储器实现。构造和更新字典 43，以便包含与压缩子系统 10 中的字典 13 相同的搜索树数据库。在输入端 41 接收每一个输入压缩编码时，都更新字典 43，以便包含与存储在字典 13 中相同的数据字符串。存储在字典 43 中的搜索树数据库结构由字典 43 的存储单元中存储的相互连接的节点组成。用众所周知的方式通过地址 44 来访问字典 43 中的存储器存储单元。

节点 45 表示搜索树的数据结构，同上述有关字典 13 的节点 15 类似，它包括一个节点数 46，一个字符域 47 和用作相关节点指针的域 48 。按照上述字典 13，节点数 46 标志树节点，并且将存储节点 45 的存储器地址 44 用作节点数。字符域 47 用来包含这个节点的数据字符值。域 48 包含一些指针，该指针连接节点 45 与相关树节点，如父节点，子节点和兄弟节点等。

解压缩子系统 40 包括一个恢复和更新控制部分 50，通过一个双向数据总线 51 和一个双向控制总线 52 与字典 43 耦合。恢复和更新控制部分 50 包括一些表示为当前接收编码寄存器 53 的工作寄存器和一个先前串寄存器 54 。按照本发明，恢复和更新控制部分 50 包括一个未识别编码处理部分 55，将按照图 8 详细解释。

恢复和更新控制部分 50 进一步包括一个编码生成器 56，用来为存储在字典 43 中的字符串指派压缩编码值。编码生成器 56 可以顺序或按照诸如散列之类的方式伪随机地指派编码数。为了系统的兼容性。编码生成器

56运用与压缩子系统10中的编码生成器26所运用的过程和算法相同的方式来指派编码数。被指派的编码通过存储器地址44来访问字典43的存储单元。这样，如上述按照压缩子系统10所述，地址44(节点数46)被用作存储在字典43中的串的编码。

恢复和更新控制部分50包括控制57，用将要叙述的方式，按照图7，图8和图10的操作流程图来控制解压缩子系统40的操作。

解压缩子系统40包括一个编码寄存器60，该寄存器缓存在输入端41接收到的压缩编码信号。各压缩编码信号按照将要叙述的方式，通过一个总线61从编码寄存器60中作用到当前接收编码寄存器53。恢复和更新控制部分50通过一个控制总线62控制从编码寄存器60中获得的压缩编码。

解压缩子系统40的操作简述如下。插入到当前接收编码寄存器53中的一个输入压缩编码信号通过地址44访问存储在字典43中的相应串。当恢复过程运用相关节点指针48通过搜索树反向跟踪串的节点时，从字符域47恢复字符串。按照适当的顺序在输出端42给出恢复的字符串。这些串恢复操作与上述参考文献中所叙述的方式相同。通过 利用当前恢复的串中的每一个字符扩展先前恢复的串以更新字典43。在这个过程中运用了先前串寄存器54。

其相应串没有存储在字典43中的未识别压缩编码信号会响应于压缩一个重复的字符或字符组串的压缩子系统10而被接收。当接收未识别的压缩编码信号后，运用未识别编码处理过程55，来恢复对应于未识别编码信号的串。另外，与这个压缩过程中压缩子系统10的字典13中所存储的内容相对应的更新串也被存储在解压缩子系统40的字典43中。未识别编码处理过程55的细节会根据图8在下文解释。

参考图3a，图3a表示字典13和43的搜索树的节点的一个有代表性的数据结构。因为，在发明的最佳实施方式中，在压缩子系统10和解压缩子系统40中，都运用了相同的节点数据结构，图1和图2中共同的参考数码示于图3a中。节点数16，46和字符域17，47已经在上文中解释过了。相关节点指针域18，48包括一个父节点指针域66和一些子节点指针域67。用众所周知的方式，父节点指针域66包含当前节点15，45的父节点的节点数，并且子节点指针域67包含当前节点15，45的子节点的节点数。

利用技术人员熟知的方式，压缩子系统 10 按下面方式向下搜索搜索树。当驻留在当前节点时，检查子节点的字符值，以确定是否有子节点与当前输入字符相匹配。如果出现匹配，子节点就成为当前节点，并用下一个输入字符不断重复这个过程，直到遇到这样一个 没有与当前输入字符相匹配的子节点的。当前节点在这种情况下，在字典 13 中找到了最长匹配串且其节点数被用作这个最长匹配串的压缩编码信号。从父节点指针域 66 所在的根节点开始通过搜索树进行的前向搜索将会含有一个空值。

用一种等效的方法，如已知的那样，可以从当前节点数和当前输入字符的一个散列函数开始，通过寻找下一个搜索节点来实现前向搜索。在该实施方式中，不会用到子节点指针域 67 。 Welch 专利(302 号)公开了 LZW 算法的散列搜索实施方式。

用一种众所周知的方式，在通过搜索树进行的反向搜索中，解压缩子系统 40 运用图 3a 的数据结构，恢复与压缩编码信号相对应的数据字符串。压缩编码信号使用编码数 46 ，并且存储字符域 47 中的字符值。利用父节点指针域 66 中的节点数来访问父节点，并且存储其中的字符值。继续这个过程，直到到达根节点。由于该过程以相反的次序来恢复字符，所以应用一种结构，例如一个 LIFO 堆栈或者一个适当构造的输出缓冲器来反转字符次序，因而恢复原始的数据字符串。

按照下述过程扩展存储在压缩字典 13 或解压缩字典 43 中的串。由编码生成器 26 或 56 提供一个空存储单元的下一个可行编码，并且将其节点数加到被扩展节点的子节点指针 67 中。将被扩展节点的节点数插入到空存储单元的父节点指针域 66 中。将扩展字符的字符值插入到空存储单元的字符域 17 或 47 中。

参考图 3b，图 3b 表示字典 13 和 43 的搜索树中节点的实际数据结构。这个数据结构和它在压缩和解压缩子系统中的实现过程在 Clark 专利(591 号)中叙述。正象按照图 3a 所解释的，在压缩字典 13 和解压缩字典 43 中都可运用图 3b 的数据结构，并且展示了图 1 和图 2 中的共同参考数字。再者，节点数 16， 46 和字符域 17， 47 在上文被解释。在图 3b 的数据结构中相关的节点指针域 18， 48 包含一个父指针域 70，一个子节点指针域 71 和一个兄弟节点指针域 72 。与上面按照图 3a 的父节点指针域 66 所描述的内

容相同，运用父节点指针域70。子节点指针域71和兄弟节点指针域72代替了图3a中的子节点指针域67。在图3b的数据结构中一个父节点用它的子节点指针域71来指向它的一个子节点的节点数，所指向的子节点利用它的兄弟节点指针域72来指向它的一个兄弟节点的节点数。而所指向的兄弟节点又利用它的兄弟节点指针域72来指向更多的兄弟节点。用这种方式，一个父节点的所有子节点的指针都包含在子节点的一个兄弟节点链表中。

除非在兄弟节点列表中搜索到存在一个输入字符根据图3a按照上述方式，进行向下搜索 。根据图3a用上述方法实现串恢复的目的，进行反向搜索，通过把子节点及其所有兄弟节点的父节点指针域设置为父节点的节点数来实现。为了简化搜索，兄弟列表应按照字符值上升的顺序排列。

如上所述，通过指派下一个可用编码，指定下一个可用的空存储单元，把这个空存储单元的节点数插入到被扩展节点的子节点指针域71来扩展没有子节点(子节点指针域=0)的串。将新创建的子节点的父节点指针域70，设置为父节点的节点数并且将扩展字符值插入到新创建的子节点的字符域中。如果要被扩展的节点已经有了子节点，则创建一个新的兄弟节点，并通过调整兄弟节点列表中适当节点的兄弟节点指针域而插入到兄弟节点列表中。将父节点的节点数插入到新创建的兄弟节点的父节点指针域70中。

参考图4和图4a，图4概要表示按照图3a中的数据结构的一个搜索树节点80。地址(节点数)，字符值，父节点和子节点如图例所示。图4a表示一部分搜索树，表示运用图4中节点80的结构来进行数据存储。图4a中的一部分搜索树由存储串ab， ac和ad的节点81， 82， 83和84组成。这样，父节点81的子节点指针域67(图3a)将会包含子节点82， 83和84的节点数，而每个子节点82， 83和84的父节点指针域66会包含父节点81的节点数。

参考图5和图5a，图5概要表示与图3b中数据结构相一致的搜索树节点90。地址(节点数)，字符值，父节点，子节点和兄弟节点如图例所示。图5a表示一部分搜索数，它运用图5中的节点90的结构，由节点91， 92，93和94组成。图5a中这部分搜索树存储图4a中存储的相同串。这样，将父节点91的子节点指针域71(图3b)设置为子节点92的节点数。将子节点92的兄弟节点指针域72设置为兄弟节点93的节点数，并将兄弟节点93

的兄弟节点指针域72设置为兄弟节点94的节点数。将每个子节点92，93和94的父节点指针域70设置为父节点91的节点数。

在以下对图6-10的详细描述中，将按照图3b的数据结构、图5中相应的节点结构和图5a中相应的搜索树结构来解释操作过程。编码被认为是由编码生成器26和56顺序指派的，尽管可以采用诸如散列法之类的伪随机指派方法。在一个散列法实施方式中，一个节点数编码和一个字符被散列，以便确定下一个跟随地址。在这样一个散列实施方式中，不会用到子节点指针和兄弟节点指针。然而，在图6-10的流程图中，操作框CODE=NEXT AVAILABLE CODE或包含下一个顺序编码，或包含下一个散列编码。在顺序编码指派实施方式中，这些操作方框具体为CODE=CODE+1。

参照图6，并继续参照图1和图3b，图6表示一个控制流程图，表现搜索和更新控制部分20要执行的操作细节，以遵照本发明进行数据压缩。控制27被视为包含适当的电路，例如状态机来控制操作的执行。

图6的流程图是基于用全部单一字符串初始化的压缩字典13。相应地，方框100提供用存储在各个编码（节点数）中的单一字符串将字典13清零和初始化的功能。这个操作运用编码生成器26进行，编码生成器26为存储这些单一字符串而顺序指派节点数。在ASCII实现方式中，编码生成器26指派前256个编码，用以存储256个单一字符串。通过把要初始化的存储器存储单元的字符域17设置为字母表的各个字符的字符值，实现初始化，利用该字母表产生压缩。将以上被初始化的存储单元的父节点指针域70、子节点指针域71和兄弟节点指针域72设置为0。可以理解被初始化的存储单元给出根节点，以存储字典13中的串，因此，这些被初始化的存储单元的父节点指针域70会保持为0。

通过以上操作，字典13的初始存储单元被设置为包含各个单一字符串。在一个ASCII实施方式中，字典13的前256个存储单元包含相应的256个字符串。在方框100的操作中，通过把其所有的域设置为0来清除字典13中的剩余存储单元。在ASCII实现方式中，节点数为266和大于266的字典存储单元被清零。

在方框101中，将当前匹配寄存器24设置为0，并且在方框102中将

先前匹配寄存器25设置为0。在方框103中，将下一个输入字符放置在当前字符寄存器23中。

在框104中，进行搜索以确定与当前字符相连接的当前匹配串是否在字典中。可以应用任何已知的合适的字典搜索过程，如在上述参考文献中描述的那些。具体而言，此时对于一个非0当前匹配，当前匹配寄存器24包含当前匹配串的节点数。由当前匹配节点的子节点指针域71所指向的子节点与输入字符进行比较。如果输入字符与当前节点的子节点匹配，则判断框104为肯定应答，并选YES路径。如果子节点与当前字符不匹配，则检查由子节点所指的兄弟节点表，以确定当前字符是否与一个兄弟节点匹配。如果找到匹配，则采取YES路径。然而，如果当前节点没有子节点，或者当前字符与当前节点的任何一个子节点都不匹配，则采取框104中的NO路径。

如果当前匹配为0，则框104在有与当前字符同值字符的字典13中寻找根节点。因为字典13用所有单一字符串初始化，所以自动选择从框104的YES分支。

当选取框104的YES分支后，图6的压缩处理过程表示在字典13中找到与当前字符相连的当前匹配，并继续进行更长串的搜索。相应地，在方框105中更新当前匹配，以便将新的当前匹配设置为连接当前字符的现有当前匹配。通过适当地更新当前匹配寄存器24中的节点数就能够实现以上处理。如上所述，当当前匹配非0时，用与当前字符匹配的子节点(或兄弟节点)的节点数来更新当前匹配寄存器24。如果当前匹配为0，用与当前字符匹配的单一字符串的节点数来更新当前匹配寄存器24。单一字符节点数可用算法以众所周知的方法来获得，或者可以通过搜索当前字符值的初始化存储单元来找到。

如果选取框104的NO分支，则与当前字符相连接的当前匹配与存储在字典13中的串不匹配。在字典中已经发现的当前匹配提供输入数据字符流的最长匹配，在框104中与当前匹配相连的当前字符“打破”了以上匹配。此时，方框106提供代表最长匹配的压缩编码信号。在当前匹配寄存器24中找到这个最长匹配的编码，即为当前匹配的节点数。

在方框107中，将当前匹配寄存器24的内容传送到先前匹配寄存器

25。这样，先前匹配寄存器25现存储代表当前最长匹配的存储单元的节点数。然后用下面将进一步讨论的方式，将利用先前匹配寄存器25更新字典13。

在将当前匹配存为先前匹配后，将当前字符存储为框110中的当前匹配。因而方框110开始搜索下一个最长匹配串，运用打破上一匹配的失配输入字符串，作为下一匹配的根字符或第一个字符。因而在方框110中，将当前匹配寄存器24设置为具有当前字符值的初始化单一字符根节点的节点数。这个过程既可以通过算法完成，也可以通过按方框105描述的方式进行搜索。方框110进入将要叙述的字典更新逻辑。

此外，也可以按照下述逻辑来实现方框110。不将方框110所表示的当前匹配设置为当前字符，可以将当前匹配设置为0，处理过程可以回到框104的输入端。因为字典的初始化，这种处理过程的结果通常导致选取框104的YES路径，从而到达方框105。如图所示，可以理解，通过方框110可以用更少的处理步骤得到同样的结果。然而，这个逻辑被用于图9的非初始化压缩处理。

当到达框105时，已经在字典中找到当前匹配的当前字符扩展，并且，按照本发明，当前字符被用于扩展当前匹配，以提供将要存储在字典13中的更新串。然而，在只接收第一个输入字符以后，当到达方框105时，不存在先前匹配并且字典13将不会在此处被更新。相应地，判断框111决定先前匹配是否为0。通过方框102中检查被初始化为0的先前匹配寄存器25来完成以上过程。如果先前匹配是0，就绕过字典更新选择从框111出来的YES路径。在第一个输入字符被处理之后，当到达方框105时，先前匹配不是0，则选取从框111出来的NO路径，以便按照本发明进行字典更新。

相应地，在方框112中，编码生成器26提供下一个可用编码。下一个可用编码将是字典中用来存储更新串的下一个可用的空存储单元的节点数。在方框113中，与当前字符相连的先前匹配被存储在字典13中的下一个可用空存储单元，利用这个下一个可用编码来访问这个空存储单元。

方框113的存储是按照下述方式实现的。由方框112中下一个可用编码所访问的下一个可用空存储单元的父节点指针域70，被设置为在先前匹

配寄存器25中找到的先前匹配的节点数。将此下一个空存储单元的字符域17设置为当前字符寄存器23中的当前字符值。按下述方式，将先前匹配父节点连接到这个新创建的节点中。先前匹配父节点的节点数位于先前匹配寄存器25中。如果先前匹配父节点没有子节点(子节点指针域=0)，就将从方框112中得到的下一个可用编码，即新创建的子节点的节点数，插入到此先前匹配父节点的子节点指针域71。如果先前匹配父节点已有子节点，就将此新创建节点的下一个可用编码节点数插入到先前匹配父节点的子节点的兄弟节点列表中。通过调整列表中兄弟节点的兄弟节点指针域72来完成以上处理，以便安置新创建的兄弟节点，并且相应地把一个适当的兄弟节点数插入到新创建的兄弟节点的兄弟节点指针域72。

运用方框114来更新先前匹配寄存器25，以便按方框113描述的那样，指向被当前字符扩展的先前匹配串的节点数。如同按照方框113所述，通过把新创建的子节点或兄弟节点的节点数，插入到先前匹配寄存器25中来实现以上处理。这个节点数就是按照方框112描述的下一个可用编码，并且由编码生成器26提供。

在按照方框112-114完成字典更新后，进入判断框115，以确定当前字符寄存器23中的当前输入字符是否为输入数据流中的最后一个输入字符。通过从111引出的YES路径也进入框115，以便象上述那样绕过字典更新。如果当前字符是最后一个字符，则采取从框115引出的YES路径到达方框116，在方框116输出当前匹配编码。按照方框116给出的压缩输出编码在当前匹配寄存器24中找到。按照方框116输出压缩编码之后，进入方框117结束处理过程。

然而，如果在当前字符寄存器23中的当前字符不是最后一个输入字符，则采用从框115引出的NO分支，该分支通过路径118回到方框103的输入端。按照方框103，将下一个输入字符插入到当前字符寄存器23中，并继续图6的数据压缩处理。

如果需要暂时停止处理过程，则在路径118中的保持方框119中执行暂停。

从前面的过程中可以理解，方框103-105控制对字典13的搜索，以找到最长匹配串，并且方框106提供与最长匹配相应的压缩编码输出。方框

110 用先前串匹配循环中导致失配的字符，开始搜索下一个最长匹配。

方框 107 和 112-114 遵照本发明控制字典 13 的更新。当框 104 确定当前输入字符已经成功地扩展了当前匹配时，方框 112-114 就把该字符与扩展过程中的先前匹配串相连接。这样，字典更新是即时的，并与以逐个字符为基础的串互相交错。

可以理解，当被匹配的当前串与被扩展的先前串在数据树的相同路径上时，就能够实现本算法有效压缩重复字符或字符组的特性。这样用两个压缩编码信号来压缩一个输入串，而不考虑其长度，参照图 8 和图 12 将更加清楚。

参考图 7，并继续参看图 2 和图 3b，图 7 表示一个控制流程图，描述为了解压缩按照图 6 生成的压缩编码，由恢复和更新控制部分 50 执行的操作。图 7 是基于用全部单一字符串初始化的解压缩字典 43 。控制 57 被视为包含适当的电路，如状态机，以便控制操作的执行。

按照方框 130，解压缩字典 43 被清零和初始化。方框 130 按照字典 43 进行的操作与上面按照方框 100 和压缩字典 13 所述方式相同。

在方框 131 中，先前串寄存器 54 被清零，在方框 132 中，将一个输入压缩编码插入到当前接收编码寄存器 53 中。

用判断框 133 继续这个处理过程，框 133 确定在当前接收编码寄存器 53 中的当前接收编码是否在字典 43 中具有相应的串。通常，字典 43 会包含一个与当前接收编码相对应的串。当从压缩装置中得到的当前接收编码遇到一个重复的字符或字符组串时，会发生例外。框 133 的判定通过用当前接收编码作为地址访问字典 43，并且确定被访问的字典存储单元是否被清零来实现。如果字典存储单元被清零，则与当前接收编码相对应的串不在字典中。用另一种方式，在顺序编码指派实施方式中，框 133 的判定可以通过确定当前接收的编码是否少于或等于编码生成器 56 中的扩展编码来实现。当当前接收编码少于或等于编码生成器 56 的扩展编码时，与当前接收编码相对应的串在字典 43 中。然而，如果当前接收编码大于扩展编码，则与当前接收编码相对应的串并不在字典 43 中。

如果当前接收编码串不在字典 43 中，则采用从框 133 引出的 NO 路径到达框 55，以执行未识别编码处理。未识别编码处理的详细过程将按照图

8在下文叙述。

当当前接收编码在字典43中有相应的串时，采用从框133引出的YES路径到达方框134。在方框134中，利用一个适当的已知的查字典过程(例如Welch专利(第302号)图5或Clark专利(第591号)图5)，恢复与当前接收编码相对应的字符串。在方框135中给出一个参数n，并设置为方框134中恢复的串中的字符数。在方框136中索引i被设置为1。索引i被用于遍历在方框134中恢复的串中的n个字符，以便从其第一个字符开始输出所恢复的串中的字符。相应地，方框137给出当前接收编码串的第i个字符的输出。

包括一个判定方框140，以便确定先前串是否为0。通过确定先前串寄存器54中的内容是否为0来实现该测试。如果先前串寄存器54为0，则采用从方框140引出的YES路径，以绕过字典更新。方框140的功能同上面按照图6中框111所叙述的相似，因而，只在响应第一个接收的输入编码时，采用从方框140引出的YES路径。

当先前串寄存器54不为0时，采用从方框140引出的NO路径，并且在方框141，由编码生成器56给出字典43中为下一个空存储单元准备的下一可用编码。在方框142中，与当前接收编码相应的串中的第i个字符相连接的先前串被存储在此下一个空存储单元中。在方框143中，更新先前串寄存器54，以存储方框142中扩展的先前串的节点数。在方框141-143中执行的操作同上面按照图6方框112-114叙述的方式相同。在图7中，使用了先前串寄存器54，而在图6中包括先前匹配寄存器25。

在方框144中，索引i增加1。如上所述，在框144输入端又采用了从判断框140引出的YES路径，以便绕过方框141-143的字典更新处理。在判断框145中进行测试，以确定索引i是否已经达到值n+1。当索引i不等于n+1时，采用从框145引出的NO路径，回到方框137输入端的处理过程。继续进行方框137和140-145的处理过程，直到i=n+1。

用这种方式，在方框137中以正确的顺序输出当前接收的编码串中的n个字符，并利用当前接收编码串的所有前缀扩展先前恢复的串。方框141-143的处理过程在解压缩字典43存储的串，与图6中方框112-114在压缩字典13中储的串相同。在解压缩字典43中按照方框141-143存储的串被

存储在与压缩字典 13 中按照方框 112-114 存储的串相同的相应地址中。

当索引 i 达到值 n+1 时，采用从框 145 引出的 YES 路径到达方框 146。在方框 146 中，当前接收的编码串代替先前串，以备处理下一个输入编码。通过把当前接收编码寄存器 53 中的内容插入到先前串寄存器 54 中来实现以上处理。未识别编码处理过程 55 退出，作为框 146 的输入。

如果刚刚处理的当前接收编码不是最后一个输入编码，则判断框 147 通过框 147 的 NO 路径，将处理过程返回到方框 132 的输入端。通过路径 148，处理过程返回到方框 132，以开始处理下一个输入压缩编码。如果需要暂停处理，则在路径 148 的保持方框 149 中执行暂停。在解压缩装置中的保持方框 149 与压缩装置中的保持方框 119 相对应。

如果在框 147 中，当前接收编码是最后一个输入编码，则采用从框 147 引出的 YES 路径，到达方框 150，以结束处理。

从上述过程中可以理解，方框 134 和方框 137 恢复和输出与当前接收编码相对应的串中的字符，而方框 141-143 通过存储利用当前恢复串中的前缀扩展的先前恢复的串来更新字典 43。

参考图 8，图 8 表示未识别编码处理过程 55 的细节。在框 160 中，索引 i 被设置为 1，并且由于叙述的原因，将会按模 n 增量。在方框 161 中，恢复先前串中的 n 个字符。先前串有 n 个字符，是由于该串是上一个串恢复循环中的当前接收编码串。运用已知的与上文按照方框 134 所述相同的字典串恢复过程，通过用先前串寄存器 54 中的内容访问字典 43 来执行方框 161。

在方框 162 中，编码生成器 56 为字典中下一个空存储单元提供下一个可用编码。在方框 163 中，将利用先前串的第 i 个字符扩展的先前串存储在下一个空存储单元中。在方框 164 中，通过更新先前串寄存器 54，利用方框 163 中被扩展的先前串替换先前串，以便存储以上被扩展的先前串的节点数。方框 162-164 的字典更新操作与上文中按照方框 141-143 叙述的相似。正象上文所解释的那样，方框 141-143 实现的字典更新过程已经按照图 6 的方框 112-114 详细叙述过。在方框 162-164 的处理过程中，运用了先前串寄存器 54。

在判断框 165 中进行测试，以确定编码生成器 56 当前提供的编码是否

等于当前接收编码寄存器 53 中的当前接收编码。如果编码生成器 56 的扩展编码不曾达到当前接收编码的值，则采用从框 165 中引出的 NO 路径到达方框 166，在方框 166 索引 i 增加 1(模 n)。然后，处理过程循环回到方框 162 的输入端，以存储更多的更新串，直到编码生成器 56 的扩展编码等于当前接收编码。

当框 165 表明编码生成器 56 的扩展编码等于当前接收编码寄存器 53 中的当前接收编码时，采用从框 165 中引出的 YES 路径到达方框 167。可以理解，当框 165 表明编码等于当前接收编码时，与以上未识别的当前接收编码相对应的串现在被存储在解压缩字典 43 中。

在方框 167 中，恢复与当前接收编码相对应的串中的字符，并且从第一个字符开始输出每一个字符。运用已知的上文中按照方框 134 所叙述的字典串恢复过程，按照当前接收编码寄存器 53 中的内容，通过访问字典 43，执行方框 167。

从上述过程中可以理解，通过方框 160-167 的处理过程，能够构造与未识别的压缩输入编码相对应的串，将其存储在解压缩字典 43 中，并且恢复其字符以便输出。进一步可理解，当图 6 的压缩装置产生并存储与未识别编码有关的串时，在与传送的编码相对应的串之外，压缩装置还存储 n 个串。根据图 12 这会进一步得到澄清。解压缩装置构造和存储这 n 个串，的处理过程如下文所述。

处理过程向前到方框 170，在方框 170 索引 i 被增加 1(模 n)。方框 171-173 分别重复方框 162-164 的处理过程。然后处理过程前进到方框 174，在方框 174 参数 n 减 1。然后在判断框 175 中测试参数 n，以确定 n 是否等于 0。如果 n 不等于 0，则采用从框 175 中引出的 NO 路径，回到方框 170 的输入端。当参数 n 达到值 0 时，在框 175 的 YES 路径中退出未识别编码处理过程。

在方框 166 和 170 中，索引 i 被增量(模 n)以便为按照方框 163 和 172 存储的串提供合适的字符值。所运用的 n 值是　在方框 174 中的减操作之前由方框 135(图 7)提供的。字符值是按照方框 161 所恢复的先前串中的 n 个字符，并且被 i 索引。按照方框 161 所恢复的先前串中的 n 个字符，为按照方框 163 和 172 构造和存储的串形成一个 n 个字符的前缀。

图 8 中的处理过程对编码生成器 26 和 56 所使用的任何类型的编码指派过程起作用，包括顺序或诸如散列之类的伪随机编码指派。当编码指派过程是顺序进行的时候，图 8 中的逻辑可以简化为下述内容。

对于顺序编码指派，框 165 的测试变成“ Code=Current Received Code+n(编码=当前接收编码+n)”。删除方框 170-175 并且从方框 167 退出未识别编码处理过程。

从上述过程中可以理解， 当出现重复的字符或字符组串时，如同上文中按照图 6 所叙述的，方框 167 恢复与未识别的接收编码相对应的串中的字符，并且方框 163 和 172 在解压缩字典 43 中存储的串与压缩字典 13 中存储的相同。

参考图 9，并继续参考图 1 和图 3b，图 9 表示搜索和更新控制部分 20 执行的详细操作的控制流程图，以便按照本发明进行数据压缩。控制 27 被视为包含合适的电路，例如状态机，以控制操作的执行。图 9 的流程图是基于一个非初始化的压缩字典 13 。在图 9 中这个非初始化的实施方式中，当第一次遇到一个字符时，以非压缩的方式，在传送这个字符之后跟随着传送一个 0 编码。这个 0 编码为解压缩装置提供指示，表明这个字符已经被压缩装置传送。第一次遇到的字符被存储在压缩装置字典 13 中，按照遇到字符的次序，用作存储的单一字符串或根节点。除了安置第一次遇到的字符，图 9 的流程图按照与图 6 中相同的方式工作。

在方框 180 中字典 13 被清零。可以通过把图 3b 中的域 17 和 70-72 设置为 0 来实现字典清零。

图 9 中的流程图包括方框 181-187 和 190-194 。这些方框分别与图 6 中方框 101， 103-107 及 112-117 相同。除下文所述以外，上文中按照图 6 中这些框图所给出的解释可应用于图 9 中的相应框图。

在框 183 中，当当前匹配是 0 时，在字典 13 中进行搜索，以确定单一字符串是否已经作为根节点进入字典。如果单一字符串已经存在于字典中，则采用从框 183 引出的 YES 路径，否则采用 NO 路径。在方框 184 中，当当前匹配是 0 时，用框 183 中匹配的单一字符根节点的节点数来更新当前匹配寄存器 24 。另外，对于顺序编码指派实施方式中的方框 187，编码指派从 1 开始，因为在以上非初始化的实施方式中，字典所有的存储单元

都可用于存储输入量中遇到的字符串。也要注意，在图 9 中不需要图 6 中绕过字典更新的方框 111 。这是由于，在这个非初始化的实施方式中，第一个输入字符是第一次遇到的字符，按照下述方式为下一次迭代提供先前匹配。

另外，方框 185 与图 6 中的方框 106 相对应。在方框 185 中，如在方框 106 中一样，在当前匹配寄存器 24 中找到压缩的输出编码。然而在方框 185 中，当当前匹配为 0 时，输出此 0 编码，表明当前输入字符是一个第一次遇到的字符。

图 9 中的流程图包括一个判定框 195，框 195 确定当前匹配寄存器 24 中的内容是否为 0 。如果当前匹配不为 0，则采用从框 195 中引出的 NO 路径达到框 186，在框 186 按照图 6 中相应方框 107 所描述的方法，将当前匹配寄存器 24 的内容传送到先前匹配寄存器 25 。在方框 196 中，将当前匹配寄存器 24 设置为 0，并且将处理过程传送到框 183 的输入端。方框 186 进行字典的更新，以进行下一个串的语法分析操作，并且方框 196 从当前字符寄存器 23 中的字符开始，形成对下一个串搜索。

如果采用了从框 195 引出的 YES 路径，则当前匹配等于 0 并且当前字符寄存器 23 中的字符已经被第一次遇到。这样，在方框 197 中，输出该字符。因为方框 185 已经输出了值为 0 的当前匹配，这个第一次遇到的当前字符，即在方框 197 中输出的字符，如上文所述，其前面是 0 编码。在方框 200 中，由编码生成器 26 给出下一个可用编码，表明字典 13 中下一个可用的空存储单元。在方框 201 中，在当前字符寄存器 23 中的下一个空存储单元中存储当前字符，作为一个单一字符串根节点。通过把当前字符寄存器 23 中的字符存储到下一个空存储单元的字符域 17 中来实现方框 201 的功能。父节点指针域 70，子节点指针域 71 和兄弟节点指针域 72 都已经在方框 180 中预先被置为 0 。

在方框 202 中，先前匹配寄存器 25 被设置为方框 201 中所创建的根节点，以便在下一个字符串语法分析操作中进行字典更新。通过把方框 201 中新创建的根节点的节点数插入到先前匹配寄存器 25 中实现该处理。以上节点数将是方框 200 刚刚给出的编码。

包括一个判断框 203，以确定当前字符寄存器 23 中的字符是否是最后

一个输入字符。如果不是，则采用从方框 203 中引出的 NO 路径达到方框 182 的输入端，以便从输入流中获得下一个数据字符信号。然而，如果在方框 203 中测试的当前字符是最后一个输入字符，则采用从方框 203 引出的 YES 路径到达方框 194，以结束处理过程。

从上述过程可以理解，方框 182-184 控制搜索最长匹配字符串字典 13，并且方框 185 提供与最长匹配相应的压缩编码输出。方框 185 也在传送首次遇到的字符之前给出 0 编码输出。方框 196 通过将当前匹配寄存器 24 置 0，开始搜索下一个最长匹配。这样，搜索下一个最长匹配字符开始于在先前串匹配循环中导致失配的字符，这个字符在当前字符寄存器 23 中。

方框 186，187，190 和 191 按照本发明控制对字典 13 的更新。当框 183 确定当前输入字符已经成功地扩展了当前匹配时，方框 187，190 和 191 把这个字符与被扩展过程中的先前匹配串相连接。框 195，197 和 200-202 控制管理第一次遇到的字符。方框 202 为下一个串匹配循环中的潜在扩展处理提供一个字符用作先前匹配。从图 9 中的逻辑可以理解，如果顺序地接收几个第一次遇到的字符，则只有最后一个字符会在下一个串匹配循环中被扩展。

参考图 10，并继续参考图 2 和图 3b，图 10 表示一个控制流程图，说明为了对根据图 9 所产生的压缩编码进行解压缩，由恢复和更新控制部分 50 执行的操作。图 10 的流程图是基于一个非初始化的解压缩字典。控制 57 被视为包含适当的电路，例如状态机，以控制操作的执行。

在方框 210 中，解压缩字典 43 用按照图 9 中方框 180 叙述的方式清零。这样，解压缩字典 43 按照与图 9 中非初始化实施方式中字典 13 相同的方式清零。

如上文中按照图 9 所述，除了管理第一次遇到的字符的方法以外，由图 10 中解压缩流程图进行的操作与图 7 中解压缩流程图进行的操作相同。相应地，图 10 包括方框 55，211-217 以及 220-226，完成的功能分别与图 7 中方框 55，132-137，141-147 以及 150 相同。上文中按照图 7 中方框 55，132-137，141-147 以及 150 给出的描述适用于图 10 中的相应方框。

在图 10 中没有使用与图 7 中方框 140 相应的方框。正如上文所述，当

先前串寄存器54为0时，方框140绕过了字典更新。这个过程发生在图7中初始化实施方式中接收第一个输入编码时。如下所述，在图10的非初始化实施方式中，第一个接收到的输入编码是关于第一次遇到的字符，从而给出一个先前串以便在随后的循环中更新。

方框217，220和221的处理过程在解压缩字典43中存储的串，与图9中方框187，190，191在字典13中存储的串相同。在压缩字典43中按照方框217，220，221存储的串，存储在与在压缩字典13中方框187，190和191中存储的串相同的地址中。进而，当图10的上下文环境中执行方框55(图8)中未识别编码处理过程时，当按照图9中压缩装置的操作，出现重复的字符或字符组串时，图8中方框163和172在解压缩字典43中存储的字符串与压缩字典13中存储的相同。

具体而言，框212与图7中框133相对应。除了在接收压缩装置首次遇到的编码之前按照0接收编码以外，上文中按照框133所叙述的内容适用于框212。然而可以理解，当处理过程到达框212时，由于这个情况在图10的另外一个下文将要叙述的分支中控制，所以当前接收编码将不会是0。

在方框211中，将一个输入压缩编码信号插入到当前接收编码寄存器53中。一个判断框230测试当前接收编码寄存器53的内容，以确定当前接收编码是否为0。如果当前接收编码不是0，则采用从框230引出的NO路径作为框212的输入端，在框212的处理过程按照上文所述进行。

然而，如果当前接收编码是0，则期望有图9中压缩装置首次遇到的字符。相应地，在方框231中输入该字符。可以利用当前接收编码寄存器53来临时保持该字符。在框232中，编码生成器56提供下一个可用编码，该编码与解压缩字典43中下一个空存储单元相对应。在方框233中，输入字符被存储在此下一空存储单元中，作为一个单一字符根节点。在方框233中，以与上文中按照图9中方框201所述内容相同的方式来实现以上处理。这样，解压缩子系统40在解压缩字典43中存储的单一字符串，及串所在地址，与图9中非初始化实施方式中压缩子系统10中存储在压缩字典13中的字符串及其地址相同。

在方框234中，将在方框231中接收到的字符输出，以维持解压缩装

置恢复数据输出和压缩装置接收的输入数据之间的一致性。可以通过从暂时存储字符的当前接收编码寄存器53中输出该字符来实现以上处理。

在方框235中，将参数n设置为1。如果这个字符，即首次遇到并且刚刚处理过的字符，在压缩装置的输入端重复，而生成一个未识别压缩编码，则n=1是以上未识别编码处理过程55的正确值，在下一个解压缩循环中执行未识别编码处理过程。

在方框236中，将先前串寄存器54设置为方框233中创建的根节点，以便在下一个串恢复循环中进行字典更新。通过将以上在方框233中新创建的根节点的节点数插入到先前串寄存器54中来实现以上处理。此节点数将是刚刚在方框232中给出的编码。从图10的逻辑中可以理解，如果顺序接收几个字符，而每个字符都是首次遇到，则只有最后一个字符会用与按照图9所叙述的内容相似的方式在下一个串恢复循环中被扩展。

方框236的输出提供给方框225的一个输入端，以便继续进行上述处理。

参考图11a-11e，图11a-11e表示当压缩一个典型的输入数据流时，压缩字典13的连续状态。用一部分搜索树来概要地表示字典的状态。如上所述，被压缩的输入流是“ abcfghx ”。所示的逗号表示串的语法分析，是虚拟的，并不在输入流中。下划线表示当前输入字符。正象图11a所示，字典13最初存储串“ abc ”和“ fgh ”。在图11a中串“ abc ”已经刚刚被匹配为最长匹配串并且输出编码“ abc ”(箭头240所示)。因而，如虚拟逗号241所示，已经根据输入对串“ abc ”进行了语法分析。

在图11b中，下一个输入字符“ f ”与箭头242所示的串“ fgh ”的第一个字符匹配。根据本发明，利用箭头243所示的字符“ f ”扩展在图11a中作为先前压缩编码传送的串“ abc ”。

在图11c中，下一个输入字符“ g ”与存储的串“ fgh ”的第二个字符匹配，并且现在利用匹配的输入字符“ g ”扩展先前扩展的字符串“ abcf ”。类似地，在图11d中，输入字符“ h ”与串“ fgh ”的第三个字符匹配，并且将其添加到不断增长的先前串上，现在形成串“ abcfgh ”。

在图11e中，下一个输入字符“ x ”打破了串“ fgh ”的匹配，并且输出最长匹配“ fgh ”的编码，如箭头244所示。如虚拟逗号245所示，已

经根据输入对串“ fgh ”进行了语法分析。

从图 11a-11e 可以理解，已经利用匹配串“ fgh ”的所有前缀扩展了所存储的串“ abc ”，并且，字典的更新是即时的而且与当前匹配串的每一个字符的匹配相交错。这样，在图 11b 中，串“ abcf ”可用于下一次迭代中的匹配。在图 11c 中，串“ abcfg ”可用于下一次迭代中的匹配，并且在图 11d 中，串“ abcfgh ”可用于下一次迭代中的匹配。

从图 11a-11e 可以进一步理解，解压缩装置接收到“ abc ”和“ fgh ”的不间断的压缩输出编码。当接收到“ fgh ”的输出编码时，解压缩装置运用与先前接收的压缩编码相对应的恢复串“ fgh ”和串“ abc ”，构造和存储图 11a-11e 所描述和表示的串。

参考图 12a-12g，图 12a-12g 表示具有行程长度编码优点的本发明的即时和交错字典更新方式。正如上文所述，本发明将号一个重复的编码或编码组压缩为两个压缩编码符号而不考虑行程的长度。图 12a-12g 概要表示搜索树的一部分，该搜索树表示当输入数据字符流是重复的字符组时，压缩字典 13 的连续状态。输入数据流表示为“ abababax ”。请注意，在此输入示例中，这个重复的字符组行程以一段字符组结束。下面描述的操作也适用于用整个字符组结束行程。如图 11a-11e 所示，虚拟逗号表示串的语法分析，且下划线表示当前输入字符。

在图 12a 中压缩字典 13 正在存储串“ ab ”，串“ ab ”与前两个输入字符匹配。这样，输出编码“ ab ”，如箭头 250 所示。虚拟逗号 251 表示输入中串“ ab ”的语法分析。

在图 12b 中，当前输入字符“ a ”与存储串“ ab ”的第一个字符匹配，如箭头 252 所示。因为“ ab ”的编码作为先前压缩编码被传送，所以将匹配字符“ a ”附加到串“ ab ”上，如箭头 253 所示。压缩字典 13 现在存储串“ aba ”，串“ aba ”可以用来进行匹配。

在图 12c 中，下一个输入字符“ b ”与存储串“ aba ”中的第二个字符匹配，因此，利用该字符扩展串“ aba ”。这样，压缩字典 13 现在存储串“ abab ”，串“ abab ”可用来进行匹配。

图 12d-12f 表示下面三个输入字符的输入字符匹配和扩展顺序。这样，在图 12d 中存储串“ ababa ”，并且串“ ababa ”可用于匹配，在图 12e

中，存储串“ababab”并且串“ababab”可用于匹配，而在图 12f 中存储串“abababa”并且串“abababa”可用于匹配。

可以理解，当这个重复的顺序继续下去时，输入字符的匹配和存储串的扩展在树的同一个分支上进行，并且不断继续，直到接收到一个打破该匹配的字符。当出现以上情况时，编码生成器 26(图 1)就不断地给出新的编码，以便在新创建的节点存储串。此时，如图 12b-12f 所示，解压缩装置并不会意识到发生在压缩装置中的这些活动。正如按照图 12a 所叙述的那样，解压缩装置上次接收的信息是串“ab”的输出编码。

在图 12g 中，因为没有在压缩字典 13 中发现串“ababax”，所以输入字符“x”打破了以上匹配。因此，输出最长匹配“ababa”的编码，如箭头 254 所示。因而，已经根据输入对上述串进行了语法分析，如虚拟逗号 255 所示。从图 12g 中可以理解，在最长匹配串之外，在压缩字典 13 中又存储了两个串。这些串由节点 256 和 257 来表示。

这样，从图 12a-12g 中可以理解，重复字符组“ababa”被压缩为两个编码符号，如参考数字 250 和 254 所表示。可以进一步理解，对于比如图所示的更长行程而言，如果字符组“ab”的重复行程超过图 12f 所示，则仍然只使用两个编码符号进行压缩。可以看到，重复行程在一段重复字符组结束，但是也可以继续，以在完整的组“ab”处终止。

从前面的过程中可以理解图 12g 中由参考数字 254 所表示的输出编码，不会被解压缩装置识别，这是由于解压缩装置并未存储串“ababa”。图 8 中未识别编码的处理过程构造并存储串“ababa”，以及在图 12b 和 12c 中所示的前缀。未识别编码处理过程进一步地构造和存储由图 12g 中的参考数字 256 和 257 表示的另外的扩展串。

在图 8 中，方框 160-167 的处理过程构造，存储并输出与箭头 254 所表示的未识别压缩编码相对应的串“ababa”。方框 170-175 的处理过程构造和存储另外的串 256 和 257。

在图 12a-12g 所示的例子中，用于未识别编码处理过程中的先前串是串“ab”，当解压缩装置按照图 12a 执行串恢复循环时，已经在解压缩装置字典中存储了串“ab”。在图 8 中，参数 n 是 2，并且增加索引 i(模 2)，因而顺序地和重复地为方框 163 和 172 提供字符“ab”，以便构造图

12b-12f 所示的字符串。

从上述过程中可以进一步理解，当通过一个失配字符或输入字符的结束来终止重复的字符或重复的字符组行程时，未识别编码处理过程构造并恢复合适的串。可以由重复字符组行程中的多字符组中的一段，或者由整个字符组来终止以上行程。图 12a-12g 表示通过失配字符“x”来终止，并且进一步用重复组的一段来终止。在图 12g 中，用重复组“ab”的一部分“a”来终止行程。当用整个组“ab”来终止重复组行程时，容易理解上述的叙述中的适当操作。

从上述过程中可以理解，上述的处理过程为存储在字典 13 和 43 中的串保留了前缀性质，在这些字典中也存在存储串的前缀。

上述实施方式压缩一个输入数据字符信号流。这些输入数据字符可以按照有任意对应字符位的任何尺寸的字母表。例如数据字符可以为文本，如 ASCII 字符，可以超过一个字母表，如 8 比特字符的 256 个字符的 ASCII 字母表。输入数据也可以是两字符二进制字母表 1 和 0(1 比特尺寸的字符)上的二进制字符。这种类型的字母表出现在，例如位图图象的情况下。可以理解，可以在基础二进制数据的两字符字母表上压缩文本数据。

上述实施方式是按照搜索最长匹配叙述的。可以理解，本发明的即时、交错的更新处理也可以用于匹配串不必为最长的系统中。

可以理解，可以用硬件、固件、软件或其组合的形式来实现本发明的上述实施方式。可以用分离电路元件来轻易地实现所述的各种功能。在软件实施方式中，可以应用合适的微处理器，其中利用根据上述流程图容易地生成的编码对微处理器进行编程。

虽然利用最佳实施方式说明了本发明，应该理解的是，所用到的词汇是描述性词汇而并不是限定性词汇，并且在附加权利要求书的范围中，可以在不背离本发明的实质和范围的情况下，在更大的方面进行改变。

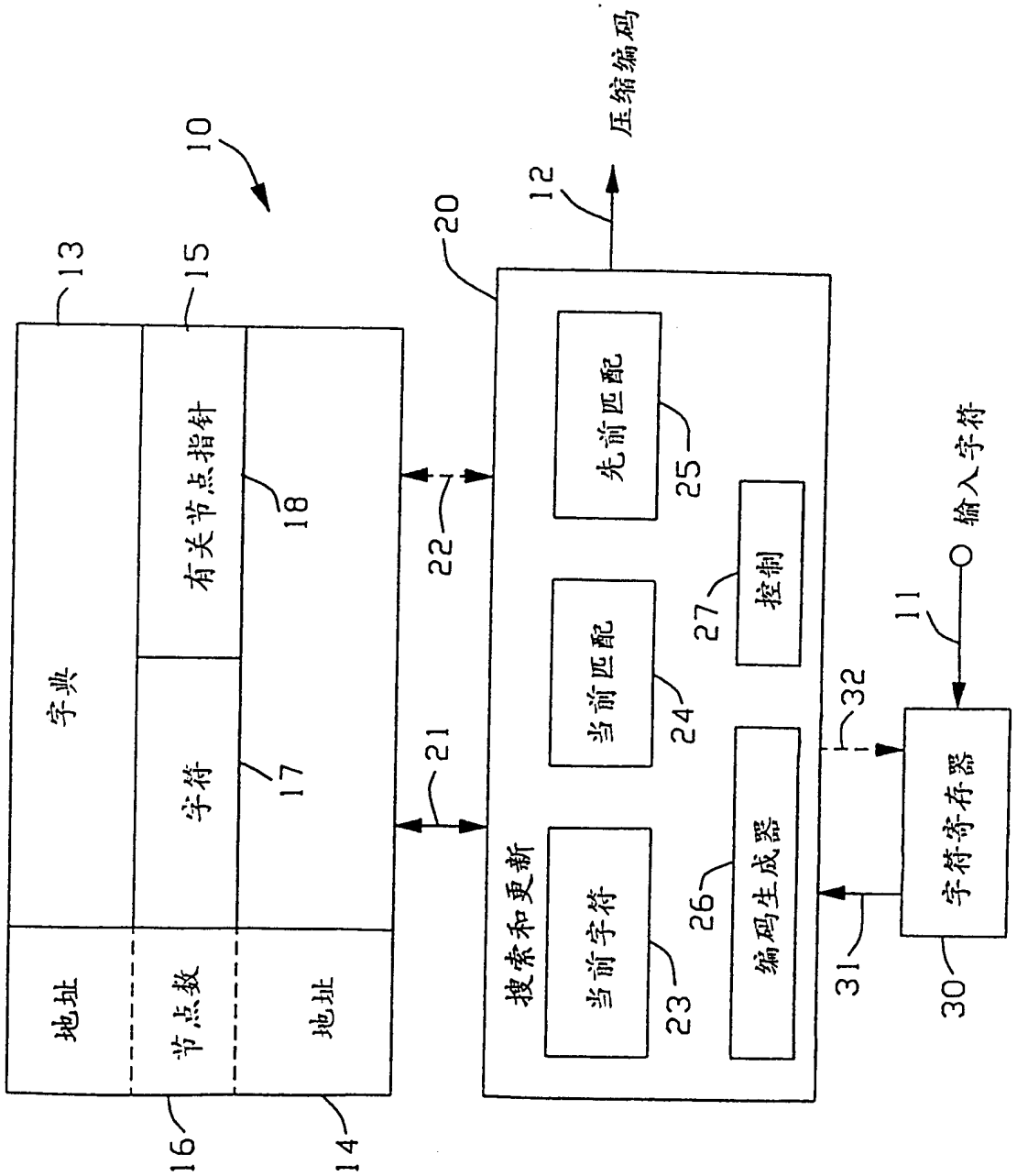

10
13
15
字典
有关节点指针
字符
18
17
地址
节点数
地址
16
14
20
12
压缩编码
22
21
搜索和更新
当前字符
23
当前匹配
24
先前匹配
25
编码生成器
26
控制
27
32
31
11
输入字符
字符寄存器
30

图 1

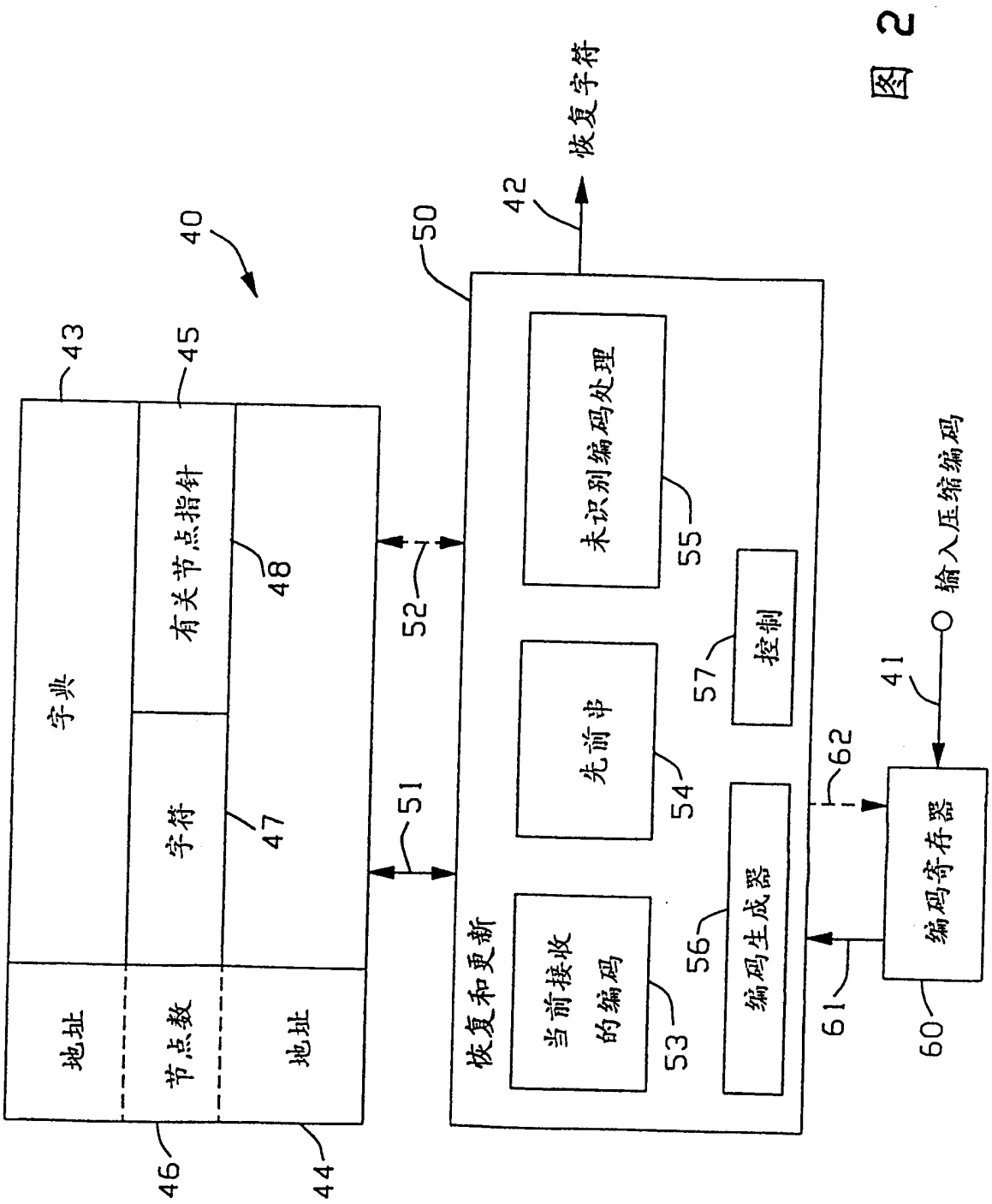

40
43
45
字典
有关节点指针
字符
48
47
地址
节点数
地址
46
44
51
52
50
42
恢复字符
恢复和更新
当前接收的编码
53
先前串
54
未识别编码处理
55
控制
57
编码生成器
56
62
61
编码寄存器
60
41
输入压缩编码

图 2

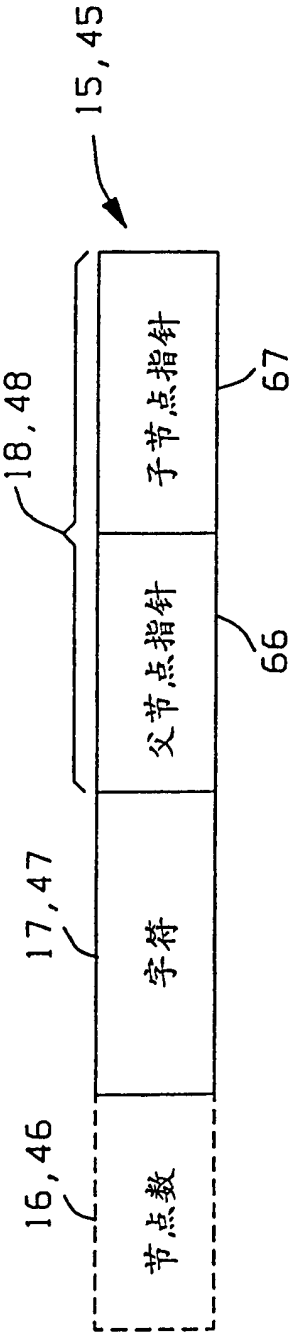

15,45
18,48
17,47
16,46
节点数
字符
父节点指针
子节点指针
66
67

图 3a

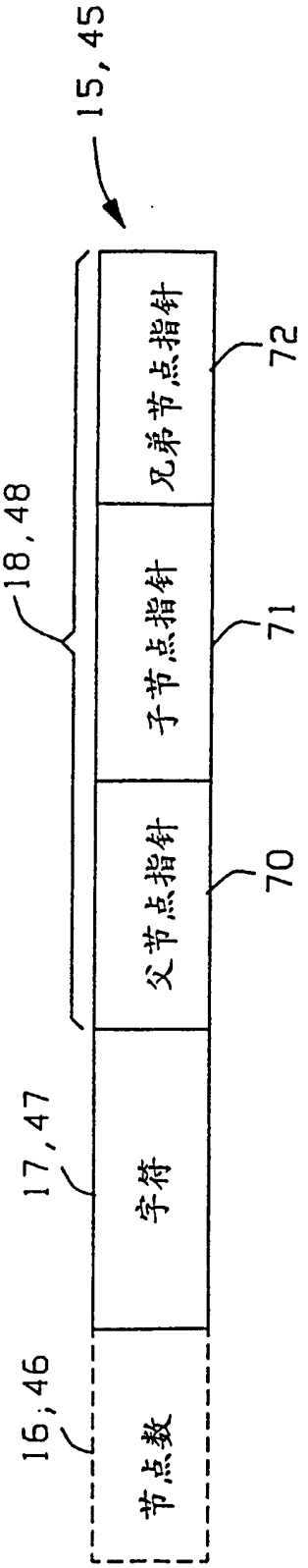

15,45
18,48
17,47
16,46
节点数
字符
父节点指针
子节点指针
兄弟节点指针
70
71
72

图 3b

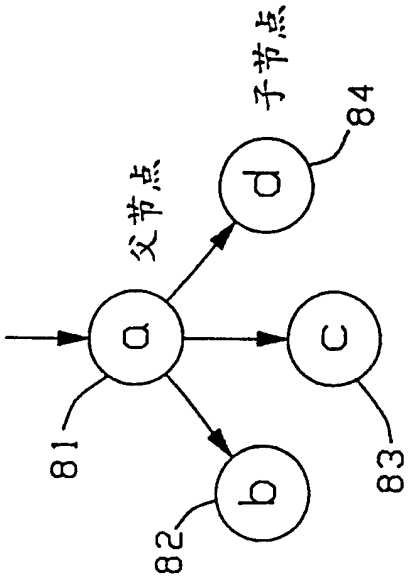

a
父节点
d
子节点
84
81
b
c
82
83

图 4a

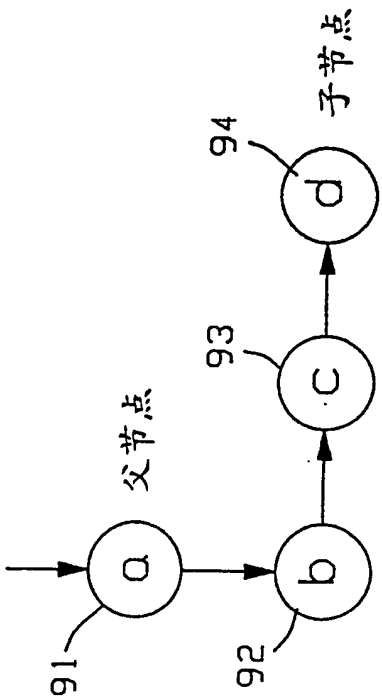

a
父节点
94
d
子节点
93
c
91
b
92

图 5a

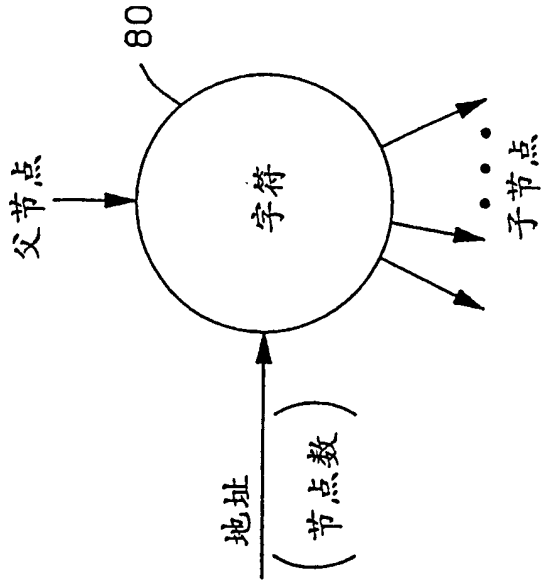

80
父节点
字符
子节点
地址
(节点数)

图 4

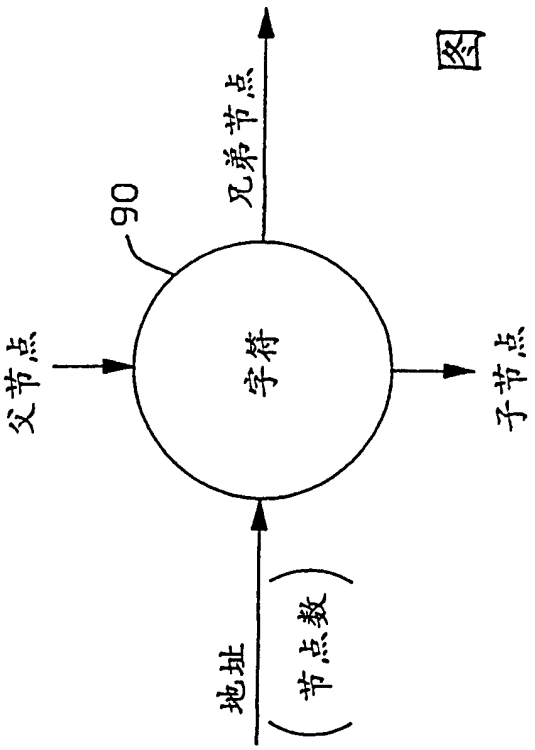

90
兄弟节点
父节点
字符
子节点
地址
(节点数)

图 5

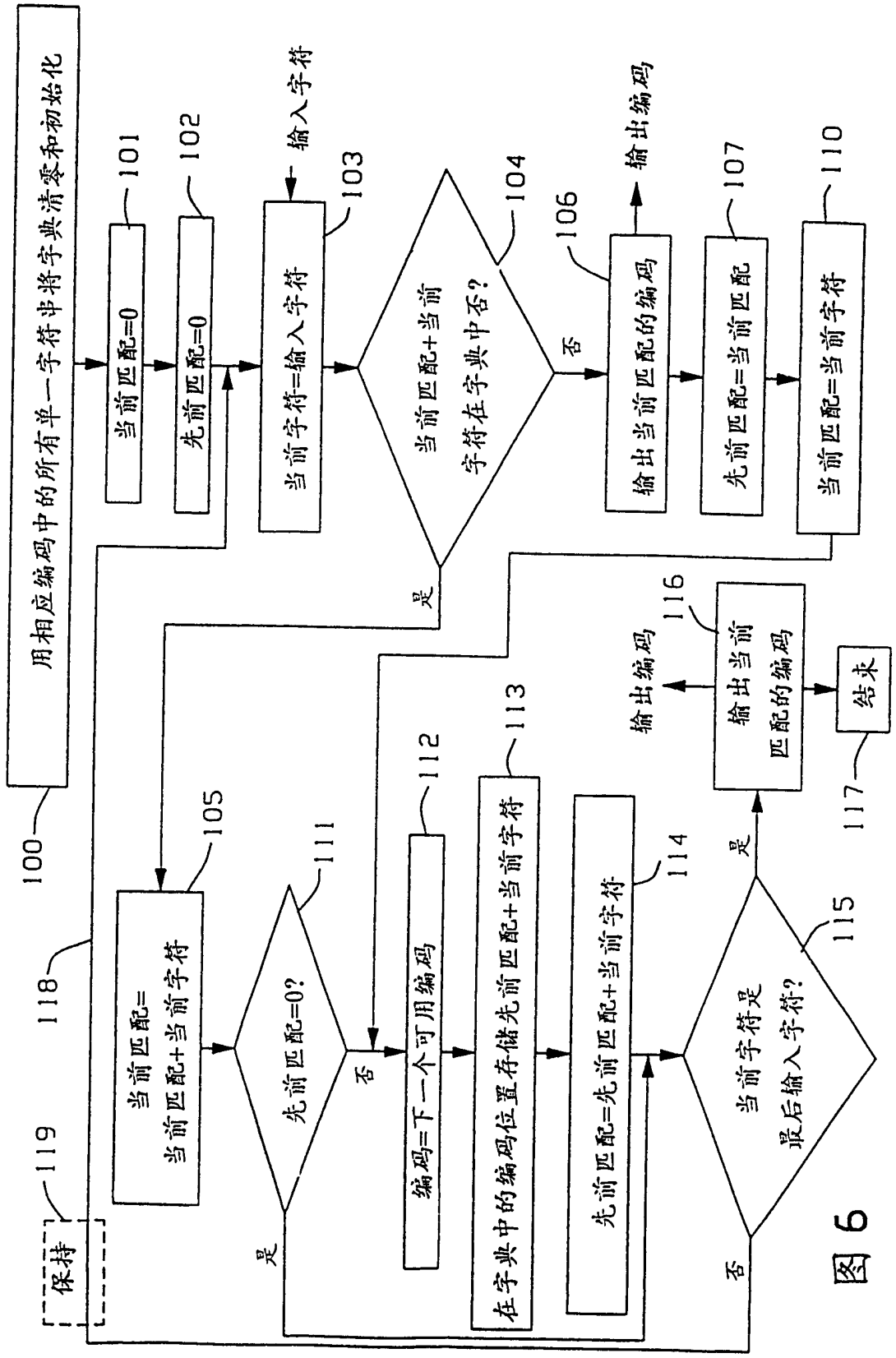
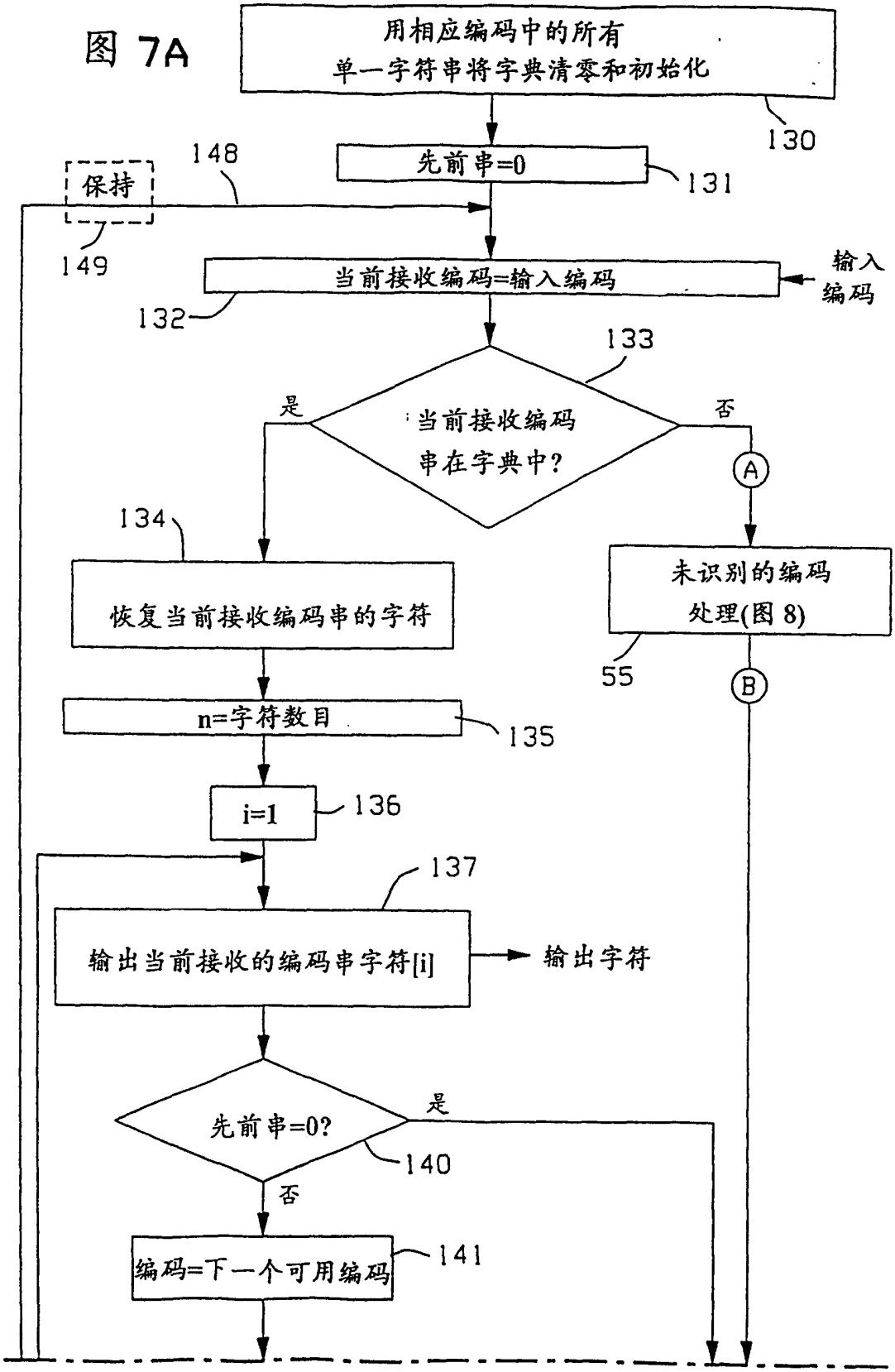

用相应编码中的所有单一字符串将字典清零和初始化
100
当前匹配=0
101
先前匹配=0
102
输入字符
当前字符=输入字符
103
当前匹配+当前字符在字典中否?
104
否
输出当前匹配的编码
106
先前匹配=当前匹配
107
当前匹配=当前字符
110
是
当前匹配=当前匹配+当前字符
105
先前匹配=0?
111
是
否
编码=下一个可用编码
112
在字典中的编码位置存储先前匹配+当前字符
113
先前匹配=先前匹配+当前字符
114
当前字符是最后输入字符?
115
是
否
输出编码
输出当前匹配的编码
116
结束
117
保持
119
118

图 6

图 7A


用相应编码中的所有
单一字符串将字典清零和初始化
130
先前串=0
131
148
保持
149
当前接收编码=输入编码
输入
编码
132
133
当前接收编码
串在字典中?
是
否
A
134
恢复当前接收编码串的字符
未识别的编码
处理(图 8)
55
B
n=字符数目
135
i=1
136
137
输出当前接收的编码串字符[i]
输出字符
先前串=0?
是
140
否
编码=下一个可用编码
141

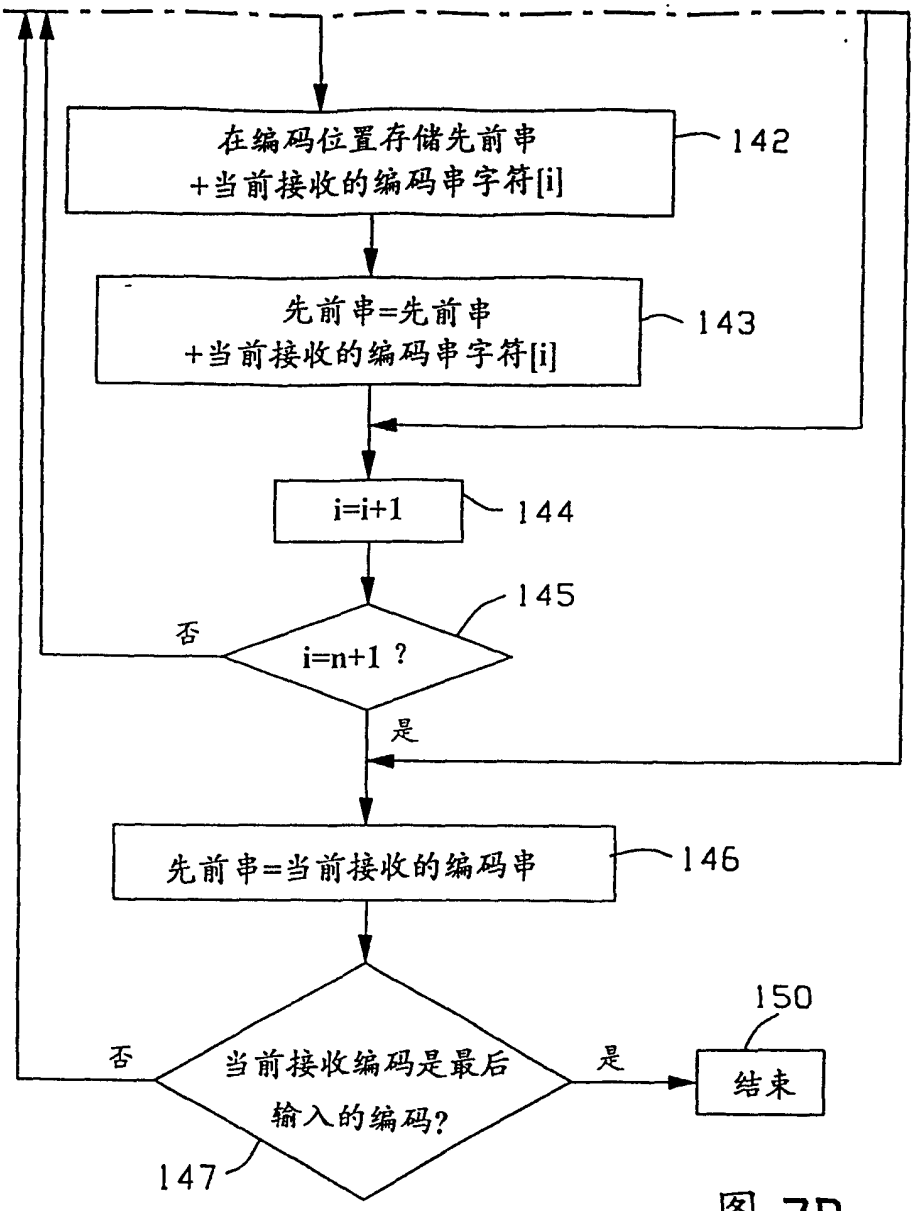


图 7B

图 7A

图 7B

图 7

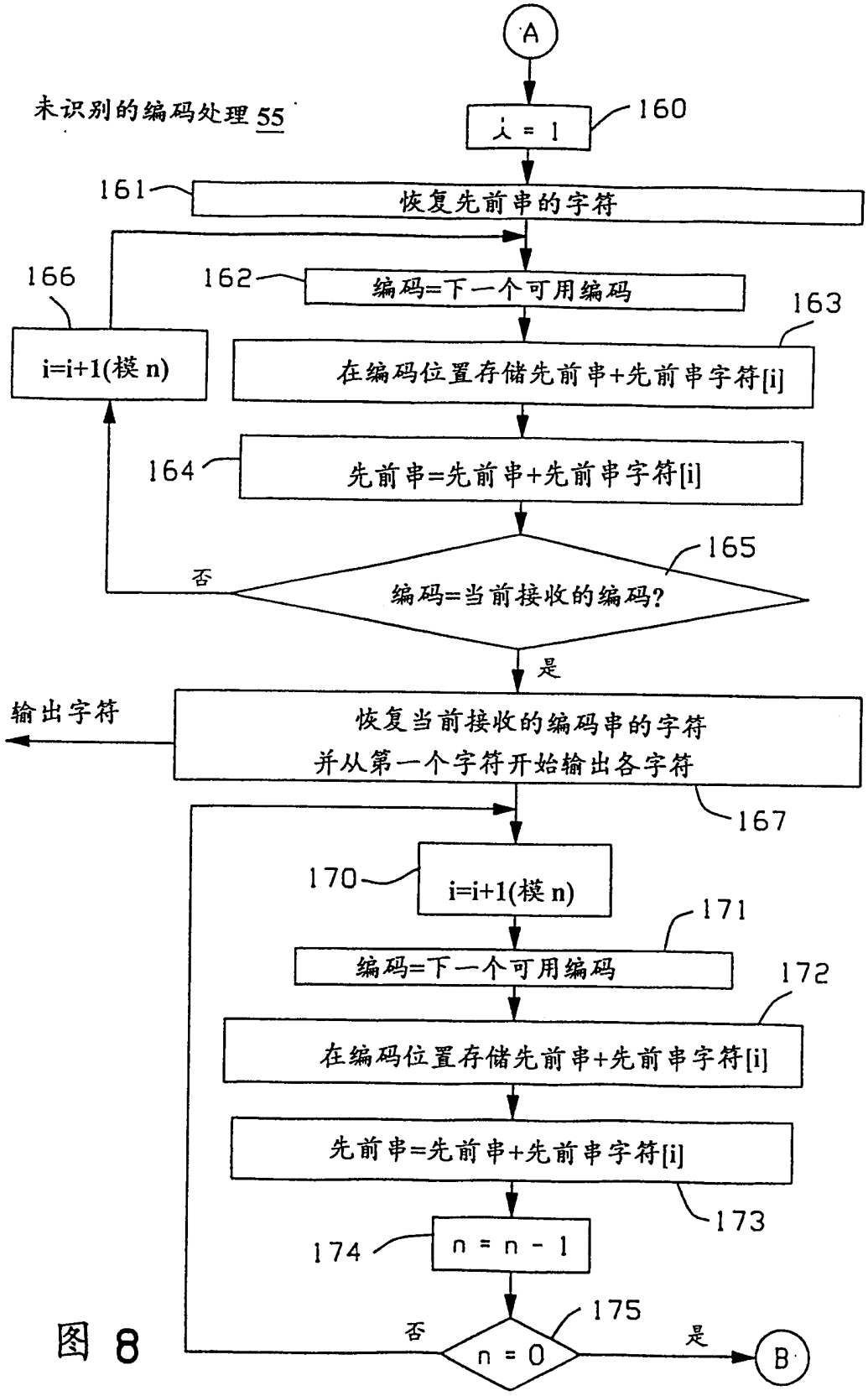

A
未识别的编码处理 55
i = 1
160
161
恢复先前串的字符
166
162
编码=下一个可用编码
163
i=i+1(模 n)
在编码位置存储先前串+先前串字符[i]
164
先前串=先前串+先前串字符[i]
165
否
编码=当前接收的编码?
是
输出字符
恢复当前接收的编码串的字符
并从第一个字符开始输出各字符
167
170
i=i+1(模 n)
171
编码=下一个可用编码
172
在编码位置存储先前串+先前串字符[i]
先前串=先前串+先前串字符[i]
173
174
n = n - 1
175
否
n = 0
是
B

图 8

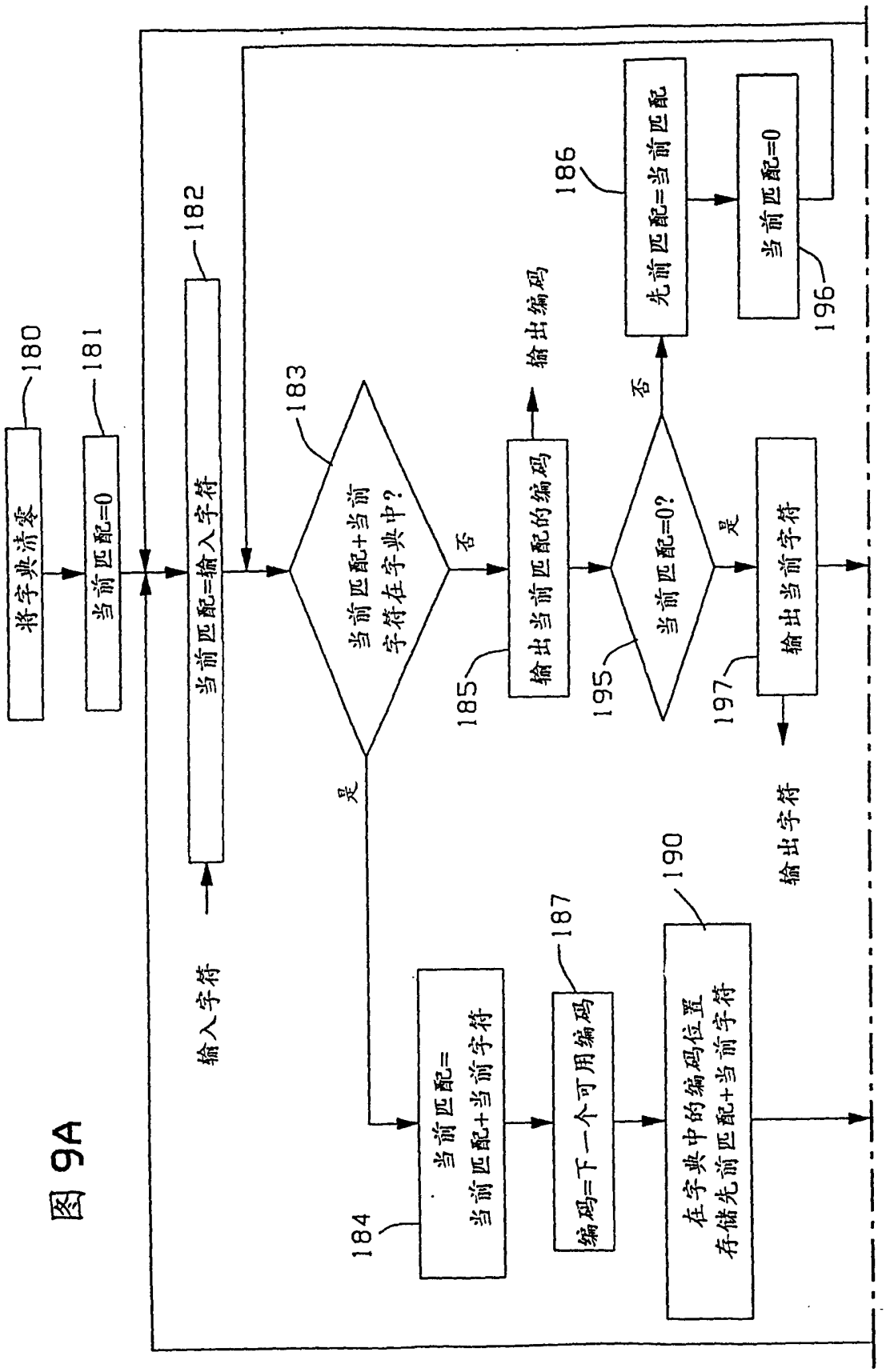

将字典清零
180
当前匹配=0
181
当前匹配=输入字符
182
输入字符
当前匹配+当前字符在字典中?
183
是
否
当前匹配=当前匹配+当前字符
184
编码=下一个可用编码
187
在字典中的编码位置存储先前匹配+当前字符
190
输出当前匹配的编码
185
输出编码
当前匹配=0?
195
是
否
先前匹配=当前匹配
186
当前匹配=0
196
输出当前字符
197

图 9A

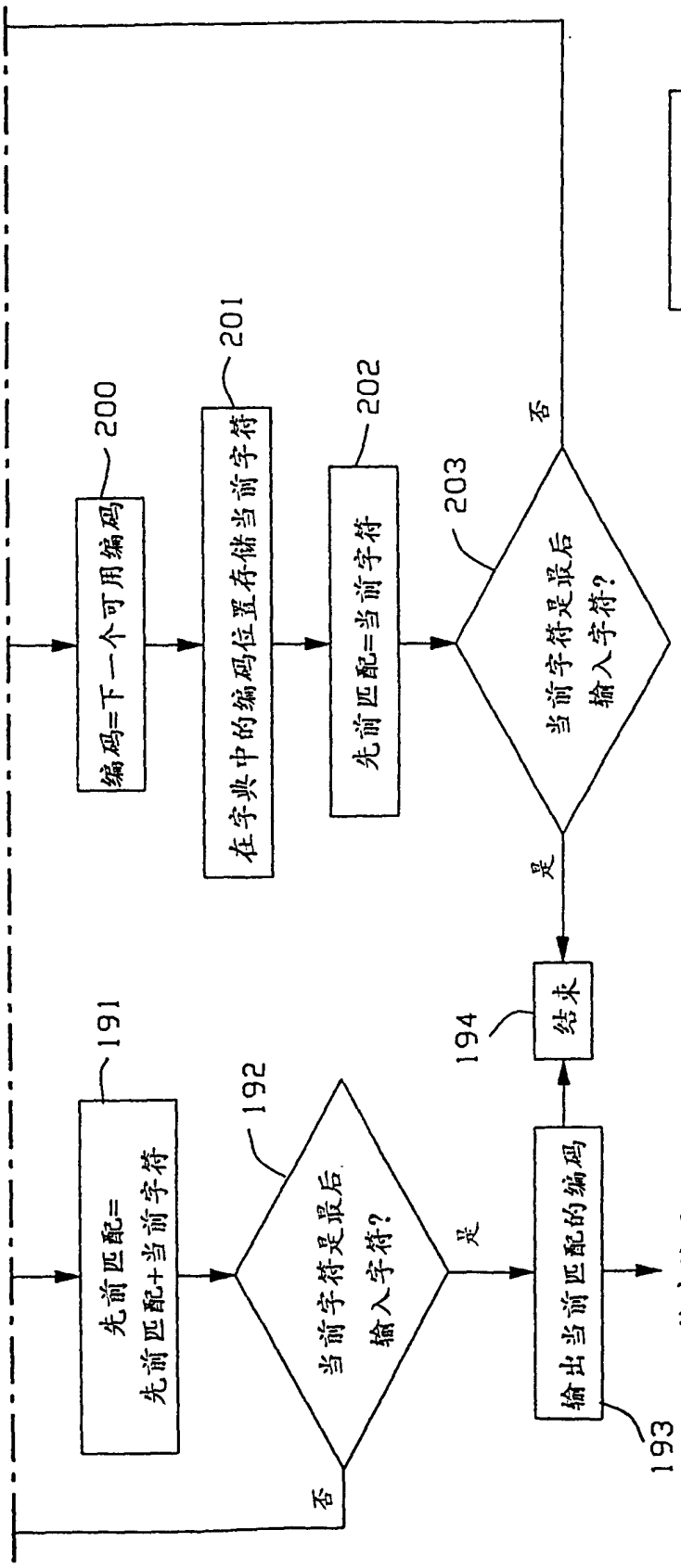

先前匹配=
先前匹配+当前字符
191
当前字符是最后
输入字符?
192
否
是
输出当前匹配的编码
193
输出编码
结束
194
编码=下一个可用编码
200
在字典中的编码位置存储当前字符
201
先前匹配=当前字符
202
当前字符是最后
输入字符?
203
否
是

图 9B

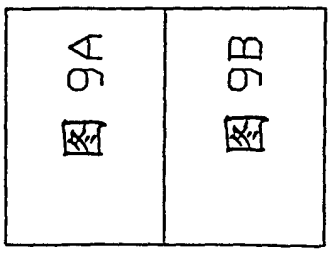


图 9A
图 9B

图 9

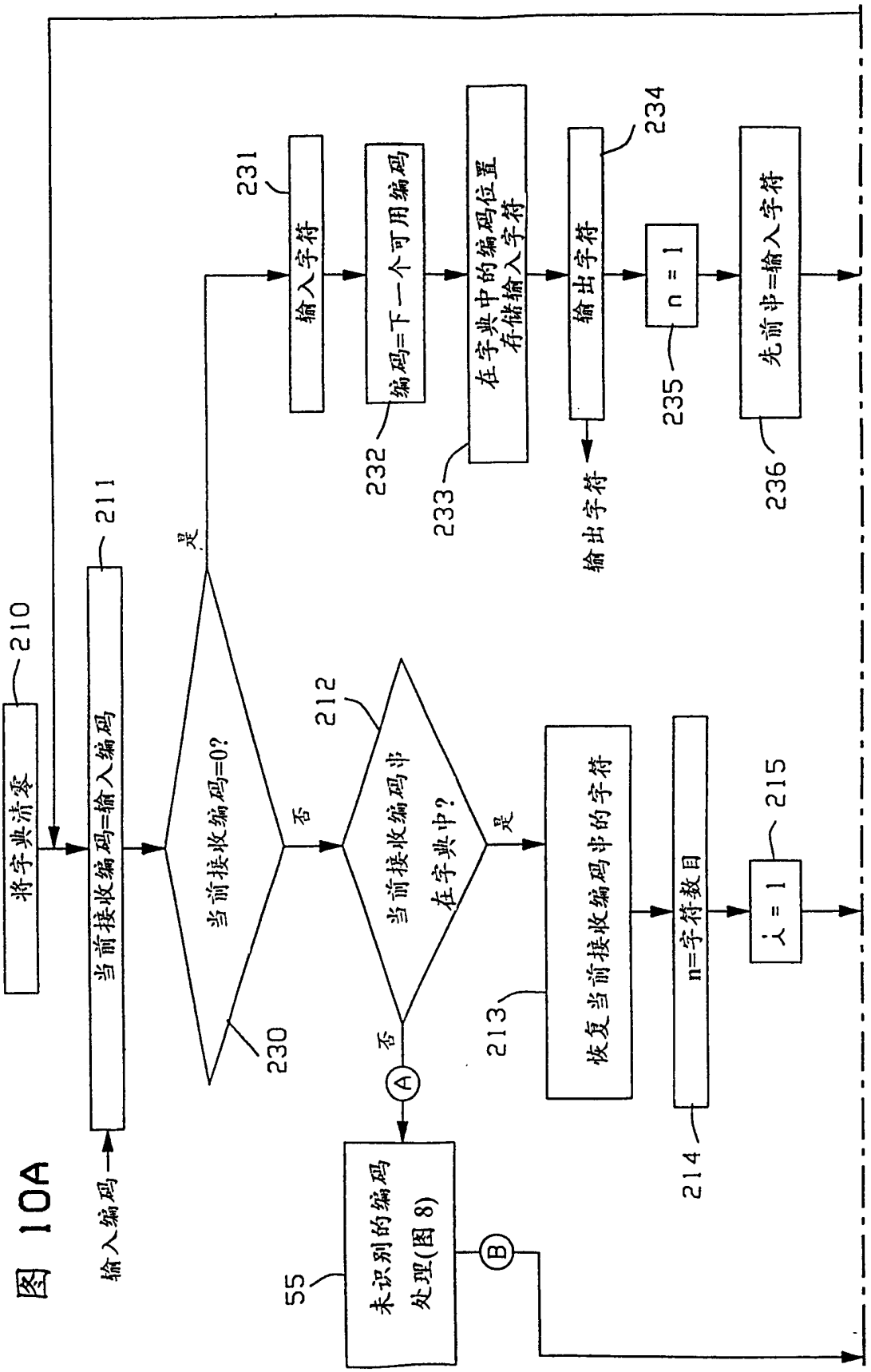

将字典清零
210
当前接收编码=输入编码
211
输入编码
当前接收编码=0?
230
是
否
输入字符
231
编码=下一个可用编码
232
在字典中的编码位置存储输入字符
233
输出字符
输出字符
234
n = 1
235
先前串=输入字符
236
当前接收编码串在字典中?
212
是
否
A
恢复当前接收编码串的字符
213
n=字符数目
214
i = 1
215
未识别的编码处理(图 8)
55
B

图 10A

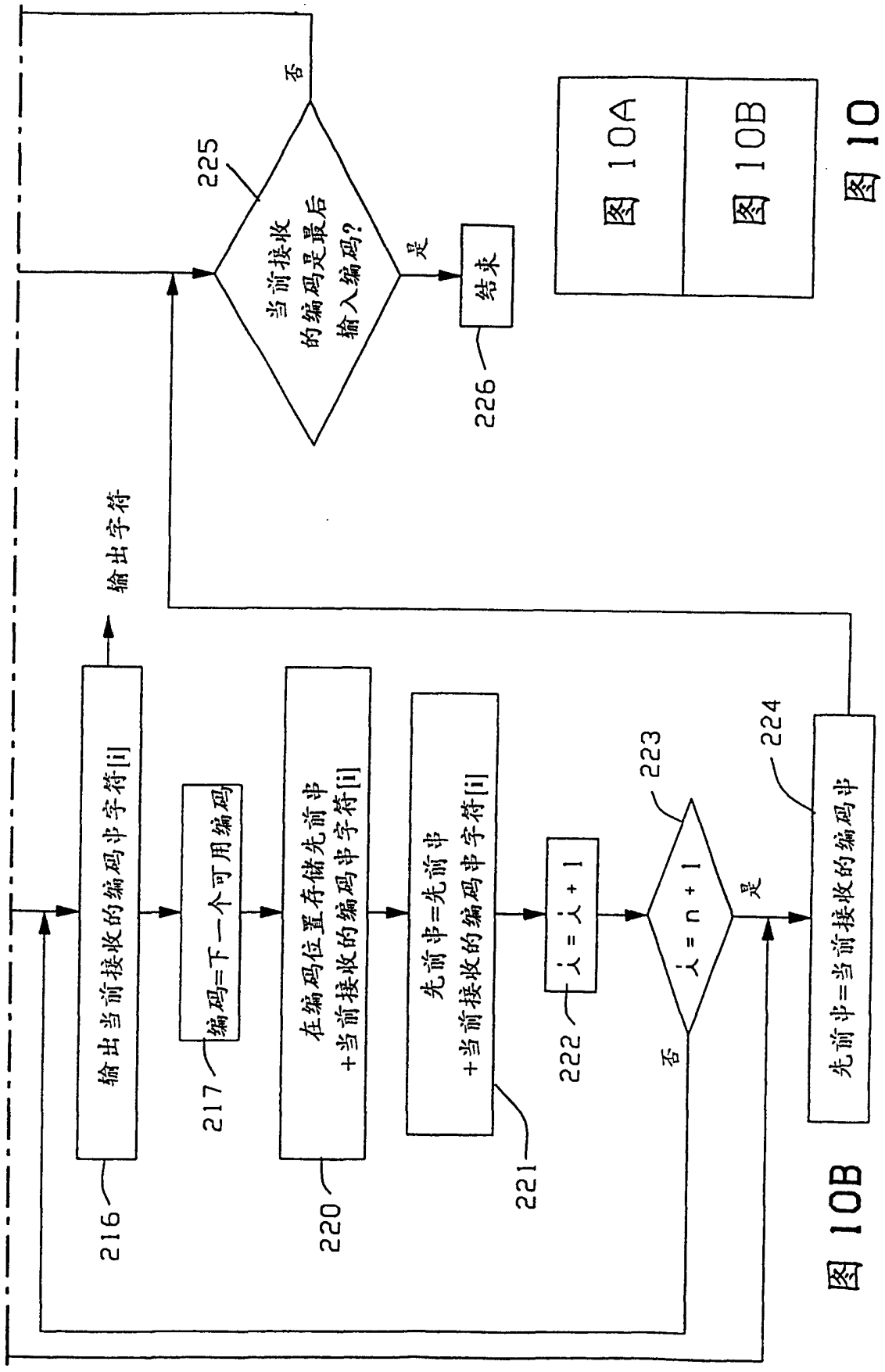


图 10A
图 10B
图 10
输出字符
输出当前接收的编码串字符[i]
编码=下一个可用编码
在编码位置存储先前串
+当前接收的编码串字符[i]
先前串=先前串
+当前接收的编码串字符[i]
i = i + 1
i = n + 1
否
是
先前串=当前接收的编码串
当前接收
的编码是最后
输入编码?
否
是
结束
216
217
220
221
222
223
224
225
226
图 10B

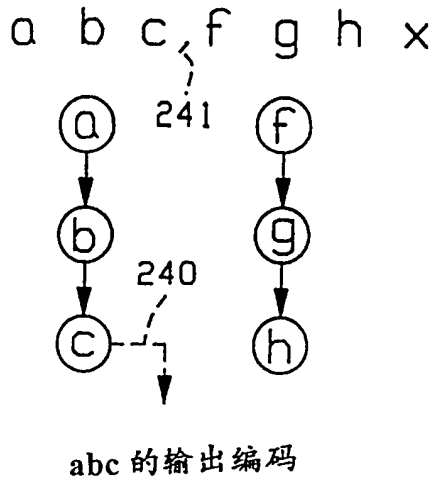


图 11a

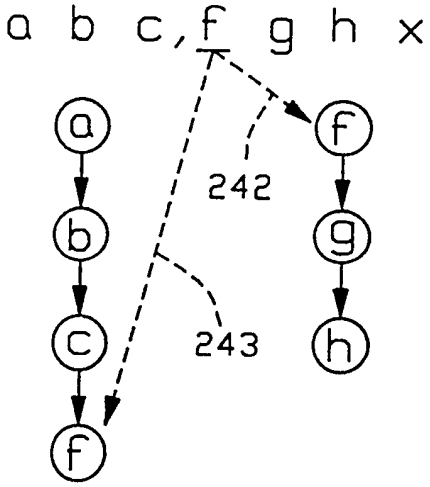


图 11b

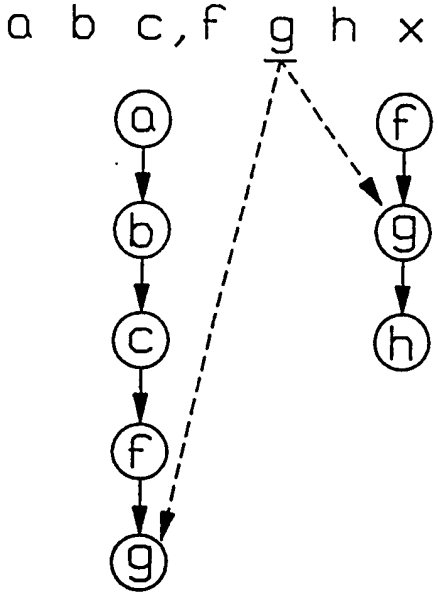


图 11c

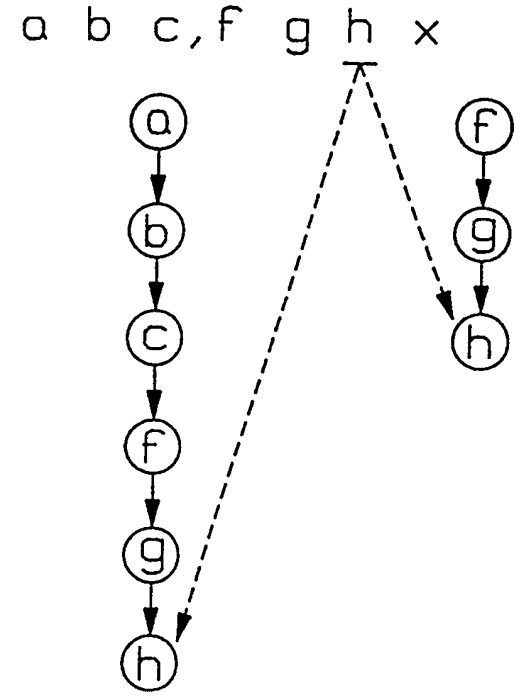


图 11d

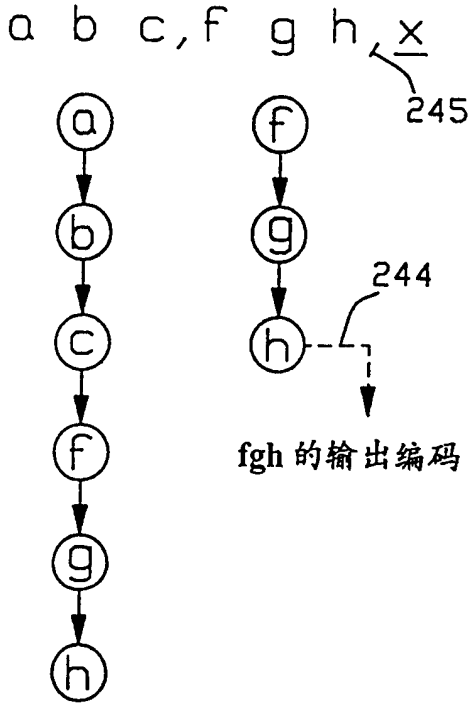


图 11e

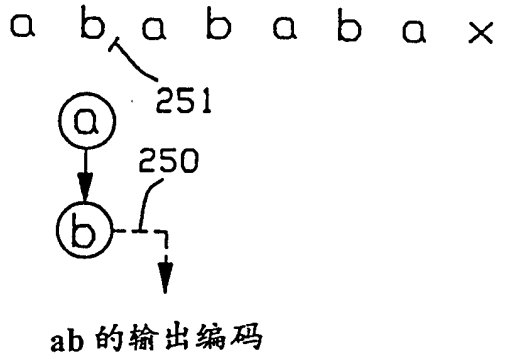


图 12a

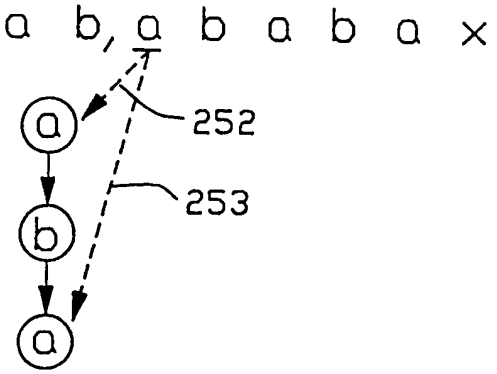


图 12b

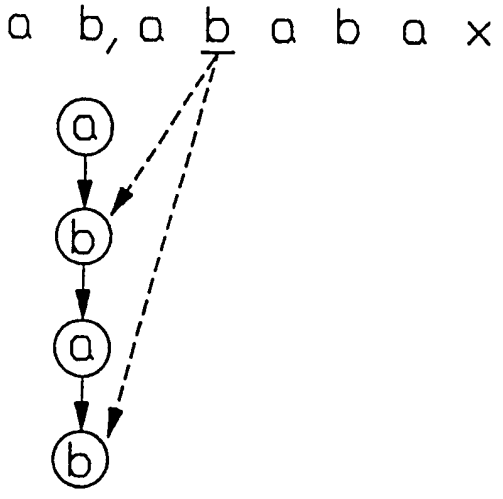


图 12c

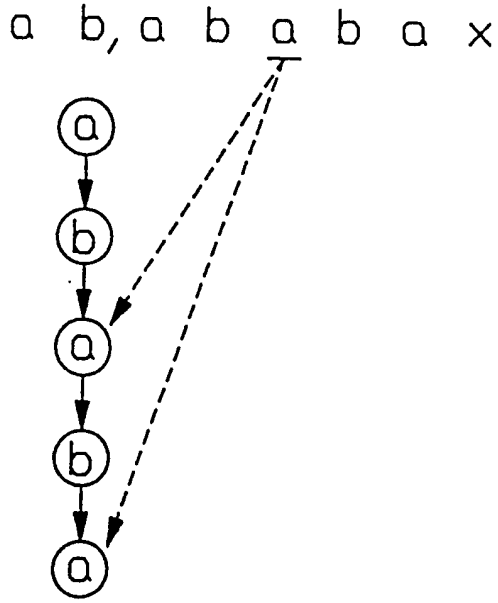


图 12d

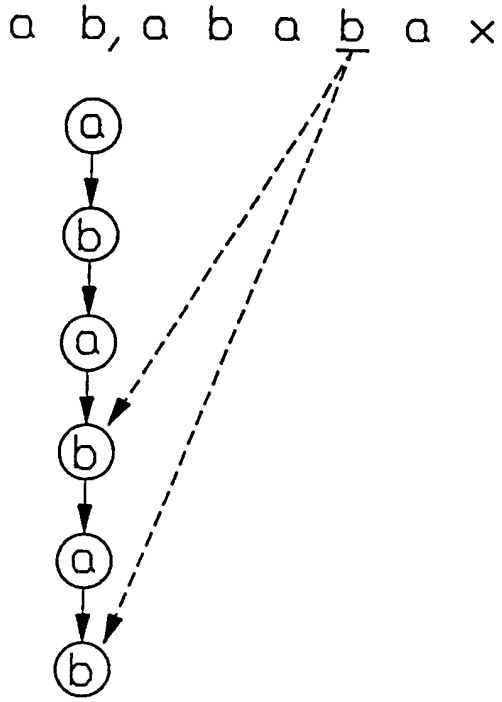


图 12e

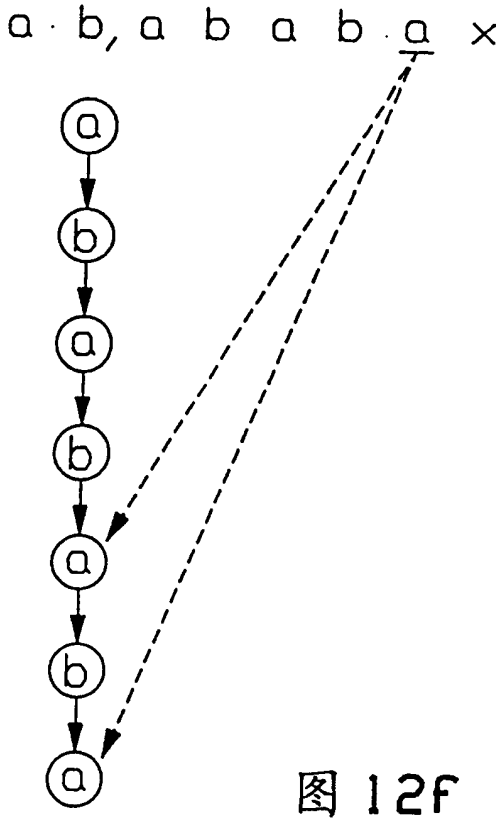


图 12f

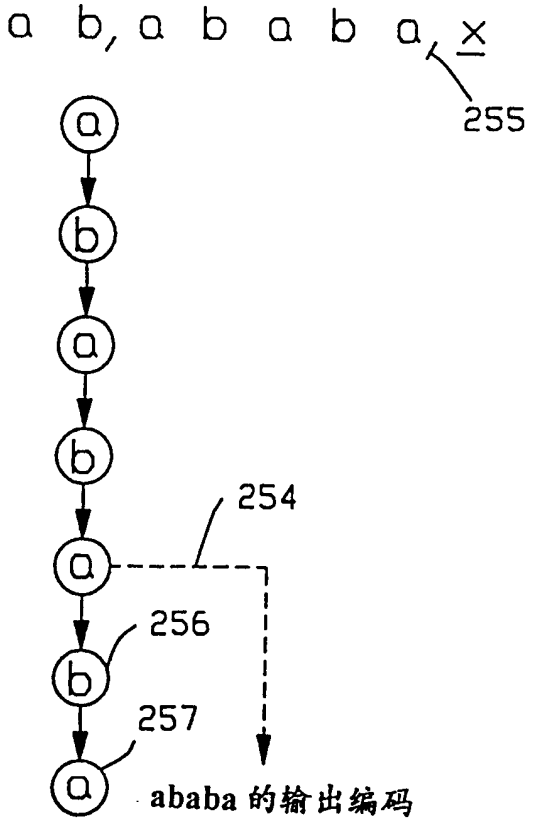


图 12g