(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2017년02월09일
(11) 등록번호 10-1705461
(24) 등록일자 2017년02월03일

(51) 국제특허분류(Int. Cl.)
H03M 7/40 (2006.01)
(52) CPC특허분류
H03M 7/40 (2013.01)
(21) 출원번호 10-2015-0122016
(22) 출원일자 2015년08월28일
심사청구일자 2015년08월28일
(56) 선행기술조사문헌
KR1019990051881 A*
KR1020120134916 A*
JP07135471 A*
KR101515660 B1*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
서울과학기술대학교 산학협력단
서울특별시 노원구 공릉로 232 (공릉동, 서울과학기술대학교)
(72) 발명자
장지훈
경기도 남양주시 순화궁로 18, 4115동 1902호 (별내동, 신안인스빌아파트)
이승은
서울특별시 강서구 허준로 23, 106동 601호 (가양동, 한강타운)
(74) 대리인
특허법인 무한

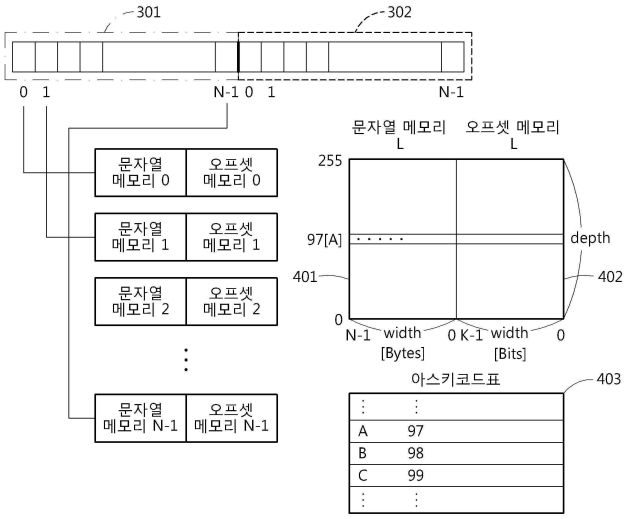
전체 청구항 수 : 총 9 항 심사관 : 조춘근

(54) 발명의 명칭 **문자열 압축 및 해제를 위한 방법 및 장치**

(57) 요 약

하드웨어 상에서 구현될 수 있는 사전 기반 압축 방법이 개시된다. 전체 문자열에서 윈도우에 대응하는 부분 문자열을 복수의 사전들 각각에서 검색하는 단계, 복수의 사전들 각각에서 상기 부분 문자열이 검색되는 경우, 상기 부분 문자열을 압축하는 단계 및 복수의 사전들 각각에서 상기 부분 문자열이 검색되지 않는 경우, 상기 부분 문자열을 복수의 사전들 중 어느 하나의 사전에 등록하는 단계를 포함할 수 있다.

대 표 도 - 도4



명 세 서

청구범위

청구항 1

전체 문자열에서 윈도우에 대응하는 부분 문자열을 복수의 사전들 각각에서 검색하는 단계;

상기 복수의 사전들 중 적어도 하나의 사전에 등록된 문자열과 상기 부분 문자열의 일치하는 부분이 검색되는 경우, 상기 일치하는 부분의 길이와 상기 전체 문자열에 대한 상기 일치하는 부분의 위치를 이용하여 상기 부분 문자열 내에서 상기 일치하는 부분을 압축하는 단계;

상기 복수의 사전들 각각에서 상기 부분 문자열이 기 설정된 최소 길이 이상 일치하는 것으로 검색되지 않는 경우, 상기 부분 문자열을 상기 복수의 사전들 중 어느 하나의 사전에 등록하는 단계

를 포함하는 문자열 압축 방법.

청구항 2

제1항에 있어서,

상기 검색하는 단계는,

상기 전체 문자열에서 상기 윈도우를 이동하면서 추출한 복수의 부분 문자열을 상기 복수의 사전들 각각에서 검색하는 문자열 압축 방법.

청구항 3

제1항에 있어서,

상기 검색하는 단계는,

상기 윈도우 내의 각각의 위치에 대응하는 문자로부터 시작하는 부분 문자열을 상기 복수의 사전들 각각에서 검색하는 문자열 압축 방법.

청구항 4

제1항에 있어서,

상기 검색하는 단계는,

상기 윈도우에 포함되는 문자의 개수에 대응하는 상기 복수의 사전들에서 상기 부분 문자열을 검색하는 문자열 압축 방법.

청구항 5

제1항에 있어서,

상기 검색하는 단계는,

상기 부분 문자열에서 시작 문자를 변환한 주소값에 기초하여 상기 부분 문자열을 상기 복수의 사전들 각각에서 검색하는 문자열 압축 방법.

청구항 6

제5항에 있어서,

상기 검색하는 단계는,

상기 부분 문자열에서 시작 문자를 아스키 코드표에 의해 변환한 주소값에 기초하여 상기 부분 문자열을 상기 복수의 사전들 각각에서 검색하는 문자열 압축 방법.

청구항 7

제1항에 있어서,

상기 등록하는 단계는,

상기 복수의 사전들 중 상기 윈도우 내의 각각의 위치에 대응하는 문자의 위치와 관련된 사전에 상기 부분 문자열의 시작 문자를 변환한 주소값에 기초하여 상기 부분 문자열 및 상기 전체 문자열에 대한 상기 부분 문자열의 위치를 등록하는 문자열 압축 방법.

청구항 8

제7항에 있어서,

상기 등록하는 단계는,

상기 윈도우 내의 각각의 위치에 대응하는 문자의 위치에 대응되는 사전의 상기 부분 문자열의 시작 문자를 변환한 주소값에 다른 부분 문자열 및 상기 다른 부분 문자열의 위치가 등록된 경우, 전체 문자열에서 등장하는 빈도수를 비교하여 상기 부분 문자열과 상기 다른 부분 문자열 중 어느 것을 등록할지 결정하는 단계를 더 포함하는 문자열 압축 방법.

청구항 9

복수의 사전에 등록된 문자열과 압축되기 전의 부분 문자열의 일치하는 부분의 길이와 전체 문자열에 대한 상기 일치하는 부분의 위치를 이용하여 압축된 부분 문자열에서 압축된 부분을 검색하는 단계;

상기 위치에 대응하는 상기 복수의 사전에 등록된 문자열을 이용하여 상기 일치하는 부분의 길이만큼의 상기 압축된 부분 문자열을 복구하는 단계; 및

상기 복구된 문자열로 상기 압축된 부분을 대체하는 단계를 포함하고,

상기 압축된 부분 문자열은,

상기 전체 문자열에서 윈도우에 대응하는 상기 압축되기 전의 부분 문자열을 상기 복수의 사전들 각각에서 검색하고, 상기 복수의 사전들 중 적어도 하나의 사전에 등록된 문자열과 상기 압축되기 전의 부분 문자열의 일치하는 부분이 검색되는 경우, 상기 일치하는 부분의 길이와 상기 전체 문자열에 대한 상기 일치하는 부분의 위치를 이용하여 상기 압축되기 전의 부분 문자열 내에서 상기 일치하는 부분을 압축하고, 상기 복수의 사전들 각각에서 상기 압축되기 전의 부분 문자열이 기 설정된 최소 길이 이상 일치하는 것으로 검색되지 않는 경우 상기 압축되기 전의 부분 문자열을 복수의 사전들 중 어느 하나의 사전에 등록하여 도출되는

문자열 압축 해제 방법.

발명의 설명

기 술 분 야

[0001] 이하의 실시예들은 데이터 압축 및 해제 방법에 관한 것으로, 보다 구체적으로는 사전 방식의 데이터 관리 방식을 이용하여 데이터를 압축하는 방법 및 해제하는 방법에 관한 것이다.

배 경 기 술

[0002] 데이터 압축 기술은, 결과적인 표현이 원래의 표현방식이 사용한 것보다 더 적은 수의 비트를 갖도록 데이터를 인코딩하는, 즉 평상시보다 공간을 덜 점유하도록 데이터를 저장하는 프로세스이다. 압축기술은 통신장치들이 같은 양의 데이터를 더 적은 비트 수로 전송 또는 저장하는 것을 가능하게 한다. 압축 작업은 원본 데이터를 받아들여 압축된 데이터를 생성하는 소정의 인코딩 알고리즘을 포함한다.

[0003] 데이터 압축은 백업 유틸리티, 스프레드시트 애플리케이션 및 데이터베이스 관리 시스템에서 광범위하게 이용된다. 비트-맵(bit-mapped) 그래픽과 같은 어떤 종류의 데이터는 데이터 압축을 통하여 원래 사이즈의 몇 분의 1로 압축이 가능하다.

[0004] 데이터 압축은 주로 두 종류의 압축, 즉 '무-손실(lossless) 압축'과 '유-손실(lossy) 압축'으로 나뉘어 진다. 무-손실 압축은 가역적이어서 원래의 데이터가 재구성될 수 있다. 반면, 유-손실 데이터 압축 체계는 약간의 데이터 손실이 발생할 수 있지만 더 높은 압축률을 달성할 수 있다.

[0005] 무-손실 데이터 압축 체계는 부분 문자열, 실행 프로그램 등의 데이터에 적용될 수 있다. 데이터 압축을 통하여 많은 양의 저장 공간이 절약될 수 있다. 데이터 압축은 데이터 압축 알고리즘을 이용하여 달성된다. 수행될 데이터 압축의 종류에 따라서 여러 개의 개별적인 알고리즘들이 사용될 수 있다. 사전 기반 압축(혹은 사전형 압축)은 무-손실 압축에 해당한다.

[0006] 허프만 코딩(Huffman coding), 산술적 코딩(arithmetic coding), 사전 기반/ 치환 알고리즘(Dictionary based/ Substitutional algorithm), 동적 생성형 사전(dynamically generated dictionary) 등과 같은 다양한 알고리즘들을 활용하여 데이터 압축이 가능하다. 상기 사전들은 복잡한 데이터 유형, 빈번한 데이터 변화 및/또는 명백한 경계가 없는 데이터 값들로써 데이터 압축비를 향상시킬 수 있다.

[0007] 소프트웨어로 구현된 사전형 압축 알고리즘은 보통 해쉬 함수를 사용하여 사전의 메모리 주소를 구한다. 일반적인 해쉬 함수들은 32-bit 또는 16-bit의 해쉬 값을 반환하여 이를 메모리 주소로 이용한다. 다만, 일반적인 PC의 경우 MMU 또는 캐쉬 등을 통해 변환된 주소의 크기에 해당하는 메모리가 필요하지 않을 수 있다. 하지만 소프트웨어로 구현된 사전형 압축 알고리즘을 하드웨어로 제조하는 경우, 시스템은 원칙적으로 변환된 주소의 크기에 해당하는 메모리가 필요하다. 소프트웨어 기반 압축의 사전은 4GB 메모리 공간 하나가 필요할 수 있다.

[0008] 4GB의 메모리 공간이 필요한 경우에도, MMU 등을 통해 가상메모리를 사용하여 4GB보다 더 작은 물리적 메모리로 시스템을 운용할 수 있다. 하지만 이 경우 추가적인 처리과정이 필요할 수 있다.

[0009] 따라서, 하드웨어로 압축 시스템을 구현할 때 소프트웨어로 구현된 사전형 압축 알고리즘을 메모리 공간에 대한 별도의 처리 없이 그대로 적용할 경우 매우 큰 메모리를 필요로 한다. 또한, 필요한 메모리의 크기를 줄이기 위해서는 복잡한 별도의 처리가 필요하다.

발명의 내용

해결하려는 과제

[0010] 본 발명의 일실시예로서 문자열을 복수의 사전에서 검색하는 구성, 검색되는 경우 문자열을 압축하는 구성 및 검색되지 않는 경우 사전에 등록하는 구성을 제공함으로써 하드웨어 상에서 별도의 처리 없이 압축 처리량을 향상시키는 방법을 제공한다.

[0011] 본 발명의 일실시예로서 압축된 부분을 검색하는 구성, 압축된 부분에 대응하는 데이터를 사전에서 추출하는 구성 및 압축된 부분을 추출된 데이터로 대체하는 구성을 제공함으로써 하드웨어 상에서 사전 기반 압축 알고리즘으로 압축된 데이터를 압축 해제하는 방법을 제공한다.

과제의 해결 수단

[0012] 본 발명의 일실시예에 따른 문자열 압축 방법은, 전체 문자열에서 윈도우에 대응하는 부분 문자열을 복수의 사전들 각각에서 검색하는 단계, 복수의 사전들 각각에서 상기 부분 문자열이 검색되는 경우, 상기 부분 문자열을 압축하는 단계 및 복수의 사전들 각각에서 상기 부분 문자열이 검색되지 않는 경우, 상기 부분 문자열을 복수의

사전들 중 어느 하나의 사전에 등록하는 단계를 포함할 수 있다.

[0013] 상기 문자열 압축 방법은 상기 검색하는 단계는, 상기 전체 문자열에서 상기 윈도우를 이동하면서 추출한 복수의 부분 문자열을 복수의 사전들 각각에서 검색하는 문자열 압축 방법일 수 있다.

[0014] 상기 문자열 압축 방법은, 상기 검색하는 단계는, 윈도우 내의 각 문자로부터 시작하는 부분 문자열을 복수의 사전들 각각에서 검색하는 문자열 압축 방법일 수 있다.

[0015] 상기 문자열 압축 방법은, 상기 검색하는 단계는, 상기 윈도우에 포함되는 문자의 개수에 대응하는 복수의 사전에서 상기 부분 문자열을 검색하는 문자열 압축 방법일 수 있다.

[0016] 상기 문자열 압축 방법은, 상기 검색하는 단계는, 상기 부분 문자열에서 시작 문자를 변환한 주소값에 기초하여 상기 부분 문자열을 복수의 사전들 각각에서 검색하는 문자열 압축 방법일 수 있다.

[0017] 상기 문자열 압축 방법은, 상기 등록하는 단계는, 상기 복수의 사전들 중 상기 윈도우 내의 각 문자의 위치에 대응되는 사전에 상기 부분 문자열의 시작 문자를 변환한 주소값에 기초하여 상기 부분 문자열 및 상기 전체 문자열에 대한 상기 부분 문자열의 위치를 등록하는 문자열 압축 방법일 수 있다.

[0018] 상기 문자열 압축 방법은, 상기 등록하는 단계는, 상기 윈도우 내의 각 문자의 위치에 대응되는 사전의 상기 부분 문자열의 시작 문자를 변환한 주소값에 다른 부분 문자열 및 상기 다른 부분 문자열의 위치가 등록된 경우, 전체 문자열에서 등장하는 빈도수를 비교하여 상기 부분 문자열과 상기 다른 부분 문자열 중 어느 것을 등록할지 결정하는 단계를 더 포함하는 문자열 압축 방법일 수 있다.

[0019] 상기 문자열 압축 방법은, 상기 압축하는 단계는, 복수의 사전들 중에서 상기 부분 문자열 중 최소 길이 이상의 일치하는 문자열이 검색되는 경우, 상기 일치하는 문자열 대신 상기 부분 문자열의 일치하는 길이와 전체 문자열에 대한 상기 부분 문자열의 위치를 출력하는 문자열 압축 방법일 수 있다.

[0020] 본 발명의 일실시예에 따른 문자열 압축 해제 방법은, 부분 문자열의 일치하는 길이와 전체 문자열에 대한 상기 부분 문자열의 위치로 압축된 문자열에서 압축된 부분을 검색하는 단계, 압축된 부분의 상기 위치에 대응하는 문자열로부터 압축된 길이만큼의 문자열을 복구하는 단계 및 상기 복구된 문자열로 상기 압축된 부분을 대체하는 단계를 포함할 수 있다.

발명의 효과

[0021] 본 발명의 일실시예로서 문자열을 복수의 사전에서 각 사전이 가지고 있는 문자열만큼 동시에 검색하는 구성, 검색되는 경우 문자열을 압축하는 구성 및 검색되지 않는 경우 사전에 등록하는 구성을 제공함으로써 하드웨어 상에서 별도의 처리 없이 압축 처리량을 향상시킬 수 있다.

[0022] 본 발명의 일실시예로서 압축된 부분을 검색하는 구성, 압축된 부분에 대응하는 데이터를 사전에서 추출하는 구성 및 압축된 부분을 추출된 데이터로 대체하는 구성을 제공함으로써 하드웨어 상에서 사전 기반 압축 알고리즘으로 압축된 데이터를 압축 해제할 수 있다.

도면의 간단한 설명

[0023] 도 1은 일실시예에 따른 사전 기반 압축 알고리즘을 하드웨어 상에서 구현한 장치를 나타낸다.

도 2는 일실시예에 따른 메모리 상에서 복수의 사전이 구현된 형태를 나타낸다.

도 3은 일실시예에 따른 압축하려는 데이터에 대해 윈도우(window) 및 제2(look-ahead window) 윈도우가 이동하는 모습을 나타낸다.

도 4는 일실시예에 따른 윈도우와 복수의 사전의 대응 관계, 각 사전을 구성하는 문자열 메모리와 오프셋 메모리의 구조 및 아스키 코드표와 주소와의 관계를 나타낸다.

도 5는 일실시예에 따른 압축을 수행하는 알고리즘을 나타낸다.

도 6은 일실시예에 따른 검색 단계를 나타낸다.

도 7은 일실시예에 따른 등록 단계를 나타낸다.

도 8은 일실시예에 따른 압축 단계를 나타낸다.

발명을 실시하기 위한 구체적인 내용

[0024] 이하, 본 발명의 실시예를 첨부된 도면을 참조하여 상세하게 설명한다.

[0025] 도 1은 일실시예에 따른 사전 기반 압축 알고리즘을 하드웨어 상에서 구현한 장치를 나타낸다.

[0026] 메모리(130)는 복수의 사전을 위한 저장 공간을 제공할 수 있다. 각 사전은 문자열 메모리와 오프셋 메모리의 쌍으로 구성될 수 있다. 문자열 메모리는 압축 하려는 전체 문자열 중의 부분 문자열을 저장하고, 오프셋 메모리는 전체 문자열 중의 부분 문자열의 위치를 저장할 수 있다.

[0027] 프로세서(110)는 입출력부(120)로부터 입력 받은 전체 문자열에서 윈도우(window)를 이동하면서 윈도우가 포함하는 각 문자열을 시작점으로 하는 부분 문자열에 대해 검색단계, 등록단계 및 압축 단계를 진행할 수 있다. 여기서 윈도우는 현재 검색, 등록 및 압축 단계를 수행하려는 범위를 의미할 수 있다. 프로세서(110)는 부분 문자열을 메모리(130)에 저장된 사전에서 검색할 수 있다. 부분 문자열이 사전에서 검색된 경우, 프로세서(110)는 압축을 수행할 수 있다. 압축된 결과는 입출력부(120)를 통해 출력될 수 있다. 부분 문자열이 사전에서 검색되지 않은 경우, 프로세서(110)는 압축을 수행하지 않고 사전에 부분 문자열을 등록할 수 있다.

[0028] 입출력부(120)는 압축하려는 전체 문자열을 입력 받아 프로세서(110)에 전송할 수 있다. 입출력부(120)는 전체 문자열 중 압축되지 않은 부분과 압축된 부분을 연결하여 출력할 수 있다.

[0029] 도 2는 일실시예에 따른 메모리(130) 상에서 복수의 사전이 구현된 형태를 나타낸다.

[0030] 메모리(130)는 복수의 사전을 저장하기 위한 공간을 제공할 수 있다. 각 사전은 문자열 메모리(210)와 오프셋 메모리(220)로 구성될 수 있다. 문자열 메모리(210)는 압축하려는 전체 문자열에서 윈도우에 포함된 각 문자로부터 시작하는 부분 문자열이 복수의 사전에서 검색되지 않을 경우 부분 문자열을 등록하기 위해 필요한 저장 공간일 수 있다. 문자열 메모리(210)는 깊이(depth)와 넓이(width)를 가질 수 있다. 여기서 깊이는 문자열 메모리의 주소를 의미할 수 있다.

[0031] 오프셋 메모리는 전체 문자열 중 부분 문자열의 위치를 저장하기 위해 필요한 공간을 제공할 수 있다. 즉, 부분 문자열이 사전에서 검색되지 않아 특정 사전의 문자열 메모리(210)에 부분 문자열이 등록되는 경우, 동일한 사전의 오프셋 메모리에 부분 문자열의 위치를 저장할 수 있다. 오프셋 메모리는 깊이(depth)와 넓이(width)를 가질 수 있다. 여기서 깊이는 문자열 메모리의 주소를 의미할 수 있다.

[0032] 도 3은 일실시예에 따른 압축하려는 데이터에 대해 윈도우가 이동하는 모습을 나타낸다.

[0033] 제1 윈도우(301)는 전체 문자열 중에서 현재 압축을 시도하려는 문자열을 포함할 수 있다. 제2 윈도우(302)는 다음 구간에서 압축을 시도하려는 문자열을 포함할 수 있고 제1 윈도우(301)에 바로 붙어있을 수 있다. 제2 윈도우(302)는 제1 윈도우(301)와 같은 크기일 수 있다. 명세서 전체에 걸쳐 단순한 "윈도우"라는 표현은 제1 윈도우를 의미할 수 있다.

[0034] 소프트웨어로 구현된 압축 알고리즘의 경우와 달리 하드웨어로 구현된 압축의 경우 제1 윈도우(301)와 같은 길이의 부분 문자열을 사전에 저장해야 하기 때문에 다음에 어떤 부분 문자열이 등장해야 하는지 알고 있어야 한다. 따라서 제2 윈도우(302)가 필요할 수 있다.

[0035] 프로세서(110)는 검색, 등록 및 압축 단계를 제1 윈도우(301) 내의 각 문자로부터 시작하는 전체 부분 문자열에 대해 동시에 수행할 수 있다. 이 점에서 프로세서(110)는 병렬 처리를 수행할 수 있다. 프로세서(110)가 제1 윈도우(301) 내의 각 문자로부터 시작하는 모든 부분 문자열에 대해 상기 과정을 완료한 경우, 제1 윈도우(301)는 제2 윈도우(302)의 위치로 이동할 수 있고 제2 윈도우(302)는 다음 구간으로 이동할 수 있다. 제1 윈도우(301)는 제1 윈도우(301) 내에 문자를 포함하지 않을 때까지 이동할 수 있다. 제1 윈도우(301)가 문자를 포함하지 않는다면 압축 과정이 종료된다.

[0036] 도 4는 일실시예에 따른 제1 윈도우(301)와 복수의 사전의 대응 관계, 각 사전을 구성하는 문자열 메모리(401)와 오프셋 메모리의 구조(402) 및 아스키 코드표(403)와 주소와의 관계를 나타낸다.

[0037] 제1 윈도우(301) 내의 각 문자의 위치는 복수의 사전 각각에 대응할 수 있다. 제1 윈도우(301)가 포함하는 문자의 개수는 사전의 개수와 동일할 수 있다. 예를 들어, 윈도우가 포함하는 문자의 개수가 N이라면 사전의 개수도 N개일 수 있다. 제1 윈도우(301)와 사전의 대응은 등록 단계에서 의미가 있다. 예를 들어, 등록 단계에서 윈도우(301)에 포함된 5번째 문자로부터 시작하는 부분 문자열은 5번째 사전에 등록될 수 있다. 다른 실시예

로서, 사전의 개수는 윈도우가 포함하는 문자의 개수보다 크거나 작을 수 있다. 이 경우 프로세서(110)은 미리 설정된 기준에 따라 복수의 사전과 윈도우가 포함하는 문자의 위치를 연관 지을 수 있다.

[0038] 문자열 메모리(401)는 깊이(depth)와 넓이(width)를 가질 수 있다. 깊이는 문자열 메모리의 주소를 의미할 수 있다. 일실시예로서 부분 문자열의 첫 번째 문자를 주소로 변환한 값을 주소로 사용하는 경우 알파벳 문자 하나는 1 바이트이므로 깊이는 0 ~ 255의 주소, 즉 28개의 주소를 가질 수 있다. 문자를 주소로 변환하는 함수는 아스키 코드 변환에 의한 것일 수 있다. 문자열의 최대 압축 길이를 제1 윈도우(301)가 포함할 수 있는 문자의 수로 정한 경우, 넓이는 제1 윈도우(301)가 포함할 수 있는 문자의 수와 동일할 수 있다. 예를 들어, 제1 윈도우(301)가 포함할 수 있는 문자의 개수가 N이라면, 문자열 메모리(401)의 넓이도 N일 수 있다. 따라서 프로세서(110)가 부분 문자열을 최대로 압축하는 경우, 프로세서(110)는 제1 윈도우(301)가 포함하는 문자의 개수의 길이만큼의 부분 문자열(길이 N)을 문자열 메모리(401)에 저장된 문자열(길이 N)로 압축할 수 있다.

[0039] 문자열 메모리(401)는 현재 압축이 수행되는 제1 윈도우(301)가 포함하는 문자의 개수와 같은 길이의 문자열을 저장할 수 있다. 제1 윈도우(301) 내에서 첫 번째 이외의 문자로부터 시작하는 부분 문자열은 제1 윈도우(301)의 길이보다 짧기 때문에 프로세서(110)는 부족한 길이는 제2 윈도우(302)의 문자를 가져와 저장할 수 있다. 즉, 프로세서(110)는 문자열 메모리(401)의 넓이와 동일한 길이의 부분 문자열을 저장할 수 있다.

[0040] 소프트웨어로 구현된 사전 기반 압축 알고리즘은 주소 변환 시 해쉬 함수를 사용한다. 이때 해쉬 테이블의 버킷이 32비트라면 232의 메모리 공간이 필요하게 된다. 이를 하드웨어에서 그대로 구현할 경우 매우 큰 메모리가 필요하기 때문에, 하드웨어로 구현된 본 발명의 일실시예는 소프트웨어로 구현된 압축 알고리즘 보다 단순한 해쉬 함수를 이용할 수 있다.

[0041] 프로세서(110)는 아스키 코드표(403)를 이용하여 단순한 해쉬 함수를 구현할 수 있다. 프로세서(110)는 압축하려는 각각의 부분 문자열의 첫 문자의 아스키 코드 표(403)에 따라 나온 숫자를 주소값으로 사용할 수 있다. 아스키 코드는 0 ~ 255의 값을 가지고 있으므로 아스키 코드 표(403)에 따라 도출된 숫자는 도 4의 문자열 메모리(401) 또는 오프셋 메모리(402)의 깊이에 대응될 수 있다.

[0042] 오프셋 메모리(402)는 깊이(depth)와 넓이(width)를 가질 수 있다. 오프셋 메모리(402)의 깊이는 문자열 메모리(401)의 깊이와 동일한 방식으로 주소를 의미할 수 있다. 넓이는 부분 문자열의 위치를 저장하기 때문에 전체 문자열의 길이에 따라 다를 수 있다. 전체 문자열의 길이가 256이라면 오프셋 메모리(402)는 256개의 위치를 저장할 수 있어야 하기 때문에 넓이는 1바이트일 수 있다. 즉, K는 8일 수 있다.

[0043] 도 5는 일실시예에 따른 압축을 수행하는 알고리즘을 나타낸다.

[0044] 프로세서(110)는 전체 문자열에서 윈도우(301)를 이동하면서 윈도우(301) 내의 각 문자열을 시작점으로 하는 부분 문자열에 대해 전체 과정을 진행할 수 있다. 윈도우(301)는 제1 윈도우(301)를 의미할 수 있다. 단계(510)에서 프로세서(110)는 현재 윈도우(301) 내에 있는 각 문자로부터 시작하는 부분 문자열 각각을 복수의 사전에서 검색할 수 있다. 복수의 사전에서 검색하는 것은 각 부분 문자열 전체에 대해 동시에 수행될 수 있다. 따라서, 프로세서(110)는 윈도우(301) 단위로 병렬 처리를 수행할 수 있다. 이때, 프로세서(110)는 각 문자의 변환된 주소값에 기초하여 복수의 사전의 문자열 메모리에 저장된 문자열과 부분 문자열을 비교할 수 있다.

[0045] 소프트웨어로 구현된 사전 기반 압축 알고리즘은 1바이트씩 입력 버퍼에서 값을 읽어와서 윈도우(301)를 1바이트씩 슬라이딩 해가면서 직접 비교하기 때문에 직렬적으로 처리한다. 하지만 본 발명의 일실시예에 따르면, 하드웨어로 구현된 사전 기반 압축 알고리즘은 윈도우(301)내에 포함된 각 문자로부터 시작하는 부분 문자열 전체에 대해 복수의 사전에서 동시에 검색 과정이 진행되기 때문에 병렬 처리를 수행할 수 있다. 예를 들어 윈도우(301)가 8바이트라면 프로세서(110)는 8개의 문자로부터 시작하는 부분 문자열 8개를 동시에 복수의 사전과 비교하기 때문에 프로세서(110)는 병렬 처리를 수행할 수 있다.

[0046] 부분 문자열의 길이는 윈도우(301)(또는 제1 윈도우)의 길이와 동일할 수 있다. 이때 부족한 길이의 문자는 제2 윈도우(302)에서 가져와 보충할 수 있다.

[0047] 단계(520)에서 프로세서(110)는 압축하려는 부분 문자열과 사전(문자열 메모리(210))에서 검색된 문자열의 일치 길이를 판단할 수 있다. 프로세서(110)는 일치 길이가 미리 설정한 최소 길이 이상인 경우 압축을 수행할 수 있다. 일실시예로 위치(offset)와 일치 길이가 보통 3바이트를 차지하므로 4바이트 이상 일치해야 이득이 1바이트일 수 있다. 따라서 4바이트 이상 일치해야 압축을 수행하게 될 수 있다. 여기서 최소 길이는 4바이트일 수 있다.

[0048] 단계(530)에서 프로세서(110)는 부분 문자열의 압축을 수행할 수 있다. 프로세서(110)는 현재 압축이 수행되어야 하는 부분 문자열이 사전(문자열 메모리(210))에 등록되어 있는 경우, 그 문자열이 등장한 위치(offset)와 일치 길이(match length)로 현재 압축이 수행되어야 하는 문자열을 대체할 수 있다.

[0049] 압축하려는 부분 문자열의 위치(offset)는 윈도우(301)의 위치에서 오프셋 메모리(220)에서 읽어온 값을 빼서 구할 수 있다. 다시 말해 프로세서(110)는 부분 문자열과 동일한 문자열(최소 길이 이상 일치하는 문자열)이 등장한 위치 (절대적 위치) 대신 부분 문자열과 사전에 등록된 동일한 문자열(최소 길이 이상 일치하는 문자열)의 상대적인 위치를 구할 수 있다. 절대적 위치보다 상대적 위치가 일반적으로 더 작기 때문에 상대적 위치를 압축 결과로 출력할 경우 압축 효과를 높일 수 있다.

[0050] 일치 길이가 최소 길이 미만인 경우 프로세서(110)는 부분 문자열이 사전에서 검색되지 않았는지 판단할 수 있다. 다시 말해 프로세서(110)는 부분 문자열과 사전(문자열 메모리(210))에 등록된 문자열의 일치 길이가 0인지 판단할 수 있다. 일치하는 길이가 0인 경우 부분 문자열은 사전에 등록되지 않은 문자열이므로 프로세서(110)는 다음에 검색할 부분 문자열의 압축을 위해 사전에 일치 길이가 0인 부분 문자열을 등록할 수 있다. 이때, 등록하고자 하는 사전 및 주소의 위치가 이미 등록된 문자열과 겹치는 경우 출현 빈도수를 기초로 더 높은 빈도수를 가진 문자열을 등록할 수 있다.

[0051] 다른 실시예로 프로세서(110)는 부분 문자열과 사전(문자열 메모리(210))에 등록된 문자열의 일치 길이가 최소 길이 미만인지 판단할 수 있다. 일치하는 길이가 최소 길이 미만인 경우 부분 문자열을 사전에 등록할 수 있다. 이때, 단계(540)에서 등록하고자 하는 사전 및 주소가 겹치는 경우, 부분 문자열은 일치하는 부분을 포함하는 이미 등록된 문자열과 부분 문자열의 출현 빈도수를 비교하여 빈도수가 큰 문자열을 사전에 등록할 수 있다.

[0052] 단계(550)에서 프로세서(110)는 복수의 사전들 중 윈도우(301)에 포함된 문자의 위치에 대응되는 사전을 찾을 수 있다. 즉, 프로세서(110)가 윈도우(301)에 포함된 특정 문자로부터 시작하는 부분 문자열을 등록하려는 경우, 프로세서(110)는 부분 문자열의 첫 문자(특정 문자)가 윈도우가 포함하는 문자 중에서 몇 번째 문자인지를 파악한 후, 같은 순번의 사전에 부분 문자열을 등록할 수 있다. 다른 실시예로서, 프로세서(110)은 미리 설정된 기준에 따라 복수의 사전 중 어느 사전에 부분 문자열을 등록할 지 결정할 수 있다. 다른 실시예로서, 프로세서(110)은 사전이 선택된 경우 미리 설정된 기준에 따라 사전의 문자열 메모리의 어느 주소에 부분 문자열을 등록할 지 결정할 수 있다. 또한, 프로세서(110)는 부분 문자열의 시작 문자를 변환한 주소값에 기초하여 부분 문자열을 등록할 주소를 찾을 수 있다. 프로세서(110)는 찾아낸 사전의 해당 주소에 기초하여 문자열 메모리(210)에 부분 문자열을 등록하고, 오프셋 메모리(220)에 전체 문자열에 대한 부분 문자열의 위치를 등록할 수 있다.

[0053] 윈도우(301) 내의 각 문자로부터 시작하는 부분 문자열에 대한 검색, 등록, 압축이 모두 완료된 경우, 단계(560)에서 윈도우(301)는 다음 구간으로 이동할 수 있다. 단계(570)에서 프로세서(110)는 윈도우(301)에 남아있는 문자가 없는 지를 판단할 수 있다. 남아있는 문자가 없다면 프로세서(110)는 전체 문자열에 대해 압축이 종료되었다고 판단할 수 있다. 남아있는 문자가 있다면 프로세서(110)는 단계(510)부터 전체 과정을 반복할 수 있다.

[0054] 도 6은 일실시예에 따른 검색 단계를 나타낸다.

[0055] 전체 문자열(610)은 세 구간으로 나뉘어져 있을 수 있다. 제1 윈도우(301)는 현재 1구간에 위치할 수 있다. 이때 제1 윈도우(301)에 포함된 각 문자로부터 시작되는 제1 윈도우(301)와 동일한 길이의 부분 문자열들(620)이 검색 대상이 될 수 있다. 프로세서(110)는 윈도우가 포함하는 문자의 개수의 길이와 동일한 길이의 문자열 단위로 검색을 수행하므로, 부족한 문자는 제2 윈도우(302)에 포함된 문자를 통해 보충할 수 있다. 예를 들어, HABCDEYU의 경우 제1 윈도우(301) 내에는 H밖에 없기 때문에 프로세서(110)는 제2 윈도우(302)로부터 ABCDEYU를 보충할 수 있다.

[0056] 문자열 메모리(630)와 오프셋 메모리(640)는 제1 윈도우(301)에 포함된 문자의 개수와 동일한 넓이를 가질 수 있다. 검색단계에서 각 부분 문자열(620)은 동시에 모든 문자열 메모리(630)의 해당 주소에 저장된 문자열과 비교될 수 있다. 다시 말하면, 부분 문자열(620) 전체에 대해 동시에 검색이 이루어지므로 프로세서(110)는 병렬 처리를 수행할 수 있다. 여기서 해당 주소는 각 부분 문자열의 첫 문자를 주소 변환한 값을 의미할 수 있다. 예를 들어, 도 4의 아스키 코드표(403)에서 A는 97로 변환되므로 프로세서(110)는 문자열 메모리(630)의 97번 주소에 저장된 문자열과 ABCDEFGH를 비교할 수 있다.

[0057] 프로세서(110)는 부분 문자열(620)과 문자열 메모리(630)에 저장된 문자열의 일치 길이를 판단할 수 있다. 일치 길이가 최소 길이 이상인 경우 도 5의 압축단계(530)를 진행할 수 있다. 일치 길이가 최소길이 미만인 경우 아예 일치하는 길이가 없다면, 즉 검색되지 않은 문자열로서 사전에 등록되지 않은 문자열이라면 도 5의 등록단계(550)를 진행할 수 있다.

[0058] 일치 길이가 있지만 최소길이 미만인 경우 프로세서(110)는 압축을 수행하지 않고 그대로 출력할 수 있다. 이는 최소 길이 미만의 부분 문자열을 압축하는 경우 압축의 이득이 없기 때문이다.

[0059] 도 7은 일실시예에 따른 등록 단계를 나타낸다.

[0060] 도 7에서 제1 윈도우(301)는 1구간에 위치하고 있다. 프로세서(110)는 부분 문자열이 사전에서 검색되지 않았는지 판단할 수 있고 검색되지 않은 경우 (혹은 일치 길이가 0인 경우) 등록단계를 수행할 수 있다. 제1 윈도우(301)가 포함하는 문자의 개수와 사전의 개수가 동일할 수 있고, 제1 윈도우(301)가 포함하는 각 문자의 위치는 복수의 사전 각각에 대응할 수 있다.

[0061] 프로세서(110)는 복수의 사전들 중 제1 윈도우(301)가 포함하는 문자의 위치에 대응되는 사전을 찾을 수 있다. 또한, 프로세서(110)는 부분 문자열의 시작 문자를 변환한 주소값에 기초하여 해당 주소를 찾을 수 있다. 프로세서(110)는 찾아낸 사전의 해당 주소에 기초하여 문자열 메모리(630)에 부분 문자열을 등록하고, 오프셋 메모리(710)에 전체 문자열에 대한 부분 문자열의 위치를 등록할 수 있다.

[0062] 예를 들어 도 7에서 프로세서(110)가 BCDEFGHA를 등록하는 경우, 1구간에서 제1 윈도우가 포함하는 문자 중 B는 두 번째 문자이므로 두 번째 사전인 사전 1에 대응할 수 있다. 프로세서(110)는 부분 문자열의 첫 번째 문자인 B의 아스키 코드표에 의한 변환의 결과값인 98에 해당하는 사전의 주소에 부분 문자열 BCDEFGHA을 등록할 수 있다.

[0063] 제1 윈도우(301)가 포함하는 각 문자의 위치에 대응되는 사전에서, 부분 문자열의 시작 문자를 변환한 주소값에 대응하는 문자열 메모리(630) 및 오프셋 메모리(710)에 다른 부분 문자열 및 상기 다른 부분 문자열의 위치가 이미 등록된 경우 덮어쓰는 문제가 발생할 수 있다. 이 경우 프로세서(110)는 전체 문자열에서 이미 등록된 부분 문자열과 현재 등록하려는 부분 문자열이 전체 문자열에서 나타나는 빈도수를 비교하여 어느 것을 등록할 지 결정할 수 있다.

[0064] 도 8은 일실시예에 따른 압축 단계를 나타낸다.

[0065] 제1 윈도우(301)는 1구간에서 사전 0의 주소 97에 ABCDEFGH를 등록한 후 현재 구간2에 위치해 있을 수 있다. 부분 문자열 ABCDEYUI 중 실제로 압축이 이루어지는 ABCDE를 포함하는 등록된 부분 문자열인 ABCDEFGH가 사전 0의 주소 97에 등록되어 있다. 압축이 수행되는 최소 길이를 4바이트로 설정한 경우, 부분 문자열 ABCDEYUI와 등록된 부분 문자열 ABCDEFGH은 일치 길이(5 바이트)가 최소 길이(4 바이트) 이상이므로 프로세서(110)는 ABCDE에 대해 압축을 수행할 수 있다.

[0066] 프로세서(110)는 실제 압축이 수행 되어야 할 문자열(ABCDE)이 등장한 상대적 위치(offset)와 일치 길이(match length)로 실제 압축이 수행 되어야 할 문자열 (ABCDE)을 대체할 수 있다. 위치(offset)는 제1 윈도우(301)의 위치에서 오프셋 메모리(640)에서 읽어온 값을 빼서 구할 수 있다. 여기서 오프셋 메모리(640)에서 읽어온 값은 동일한 주소의 문자열 메모리에 등록된 부분 문자열이 전체 문자열에서 나타나는 위치를 의미할 수 있다.

[0067] 제1 윈도우(301)의 위치는 제1 위도우(301)가 포함하는 첫 문자의 위치를 의미할 수 있다. 도 8에서 각 문자의 위치를 ABCDEFGH 순으로 A의 위치를 0이라고 한다면, 제1 윈도우(301)의 위치는 8일 수 있다. 오프셋 메모리(640)에서 읽어온 값은 0일 수 있다. 따라서 문자열(ABCDE)이 등장한 상대적 위치(offset)는 8 - 0 = 8이 될 수 있다.

[0068] 1구간의 각 문자로부터 시작하는 부분 문자열들은 모두 사전에서 검색되지 않는 문자열로서 어느 것도 압축되지 않는다면, 2구간의 A로부터 시작하는 부분 문자열까지 처리한 결과는 결과(820)와 같을 수 있다. 결과(820)에서 ABCDEFGH는 1구간의 문자열이 압축되지 않아서 그대로 출력된 것이며 85는 ABCDEYUI 중 ABCDE가 압축된 결과이고, 여기서 8은 위치(offset)를 의미하고 5는 일치하는 길이(ABCDE)를 의미할 수 있다.

[0069] 본 발명의 다른 실시예에 따르면, 하드웨어에서 구현된 사전 기반 압축 알고리즘에 의해 압축된 문자열을 해제하는 방법을 제공할 수 있다.

[0070] 프로세서는 부분 문자열의 일치하는 길이와 전체 문자열에 대한 부분 문자열의 위치로 압축된 문자열에서 압축

된 부분을 검색할 수 있다. 이후 압축된 부분의 위치에 대응하는 문자열로부터 압축된 길이만큼의 문자열을 복구할 수 있다. 프로세서는 복구된 문자열로 상기 압축된 부분을 대체함으로써 압축 해제를 완료할 수 있다.

[0071] 이상에서 설명된 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 장치 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPGA(field programmable gate array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0072] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(signal wave)에 영구적으로, 또는 일시적으로 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0073] 실시예에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어부로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

[0074] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다. 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한다.

부호의 설명

[0075] 301: 제1 윈도우

302: 제2 윈도우

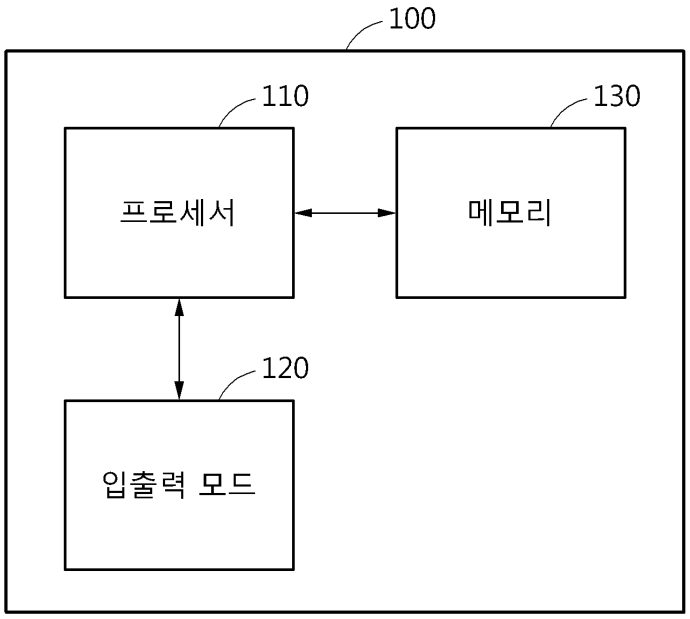
401: 문자열 메모리

402: 오프셋 메모리

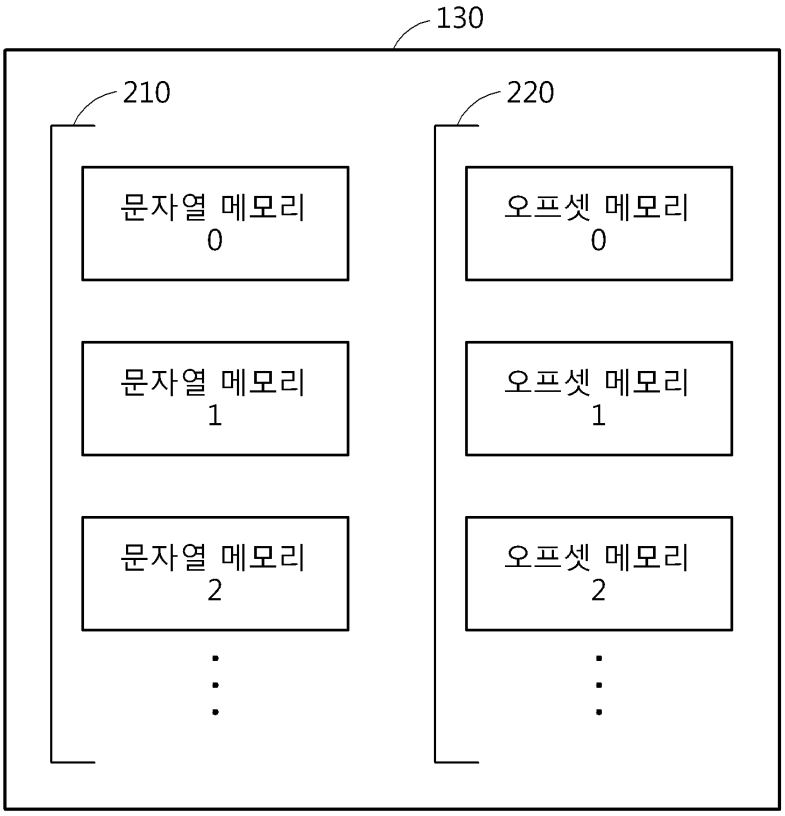
403: 아스키 코드표

도면

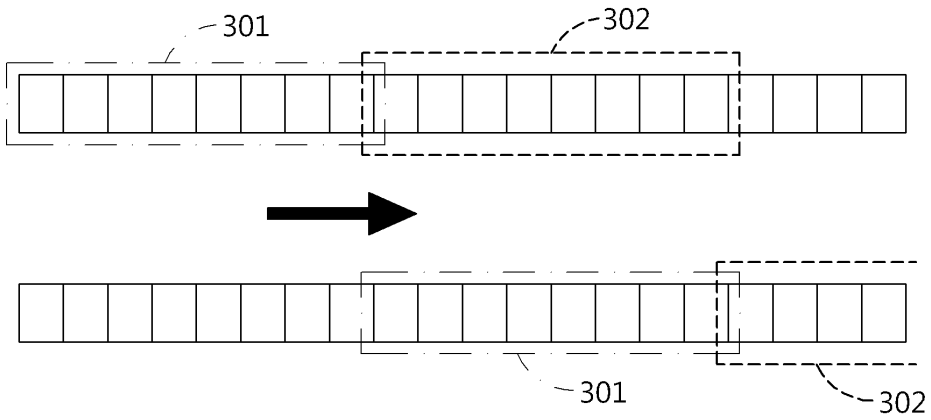
도면1


100
110
130
프로세서
메모리
120
입출력 모드

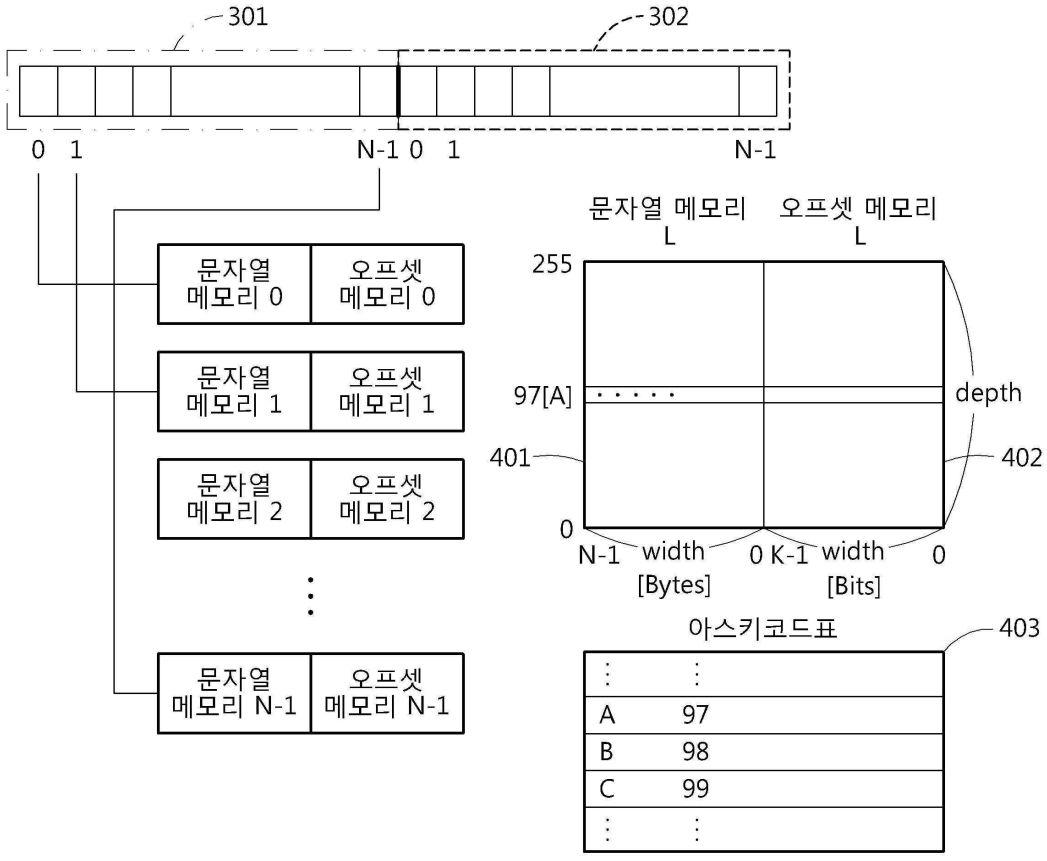
도면2


130
210
220
문자열 메모리 0
문자열 메모리 1
문자열 메모리 2
오프셋 메모리 0
오프셋 메모리 1
오프셋 메모리 2

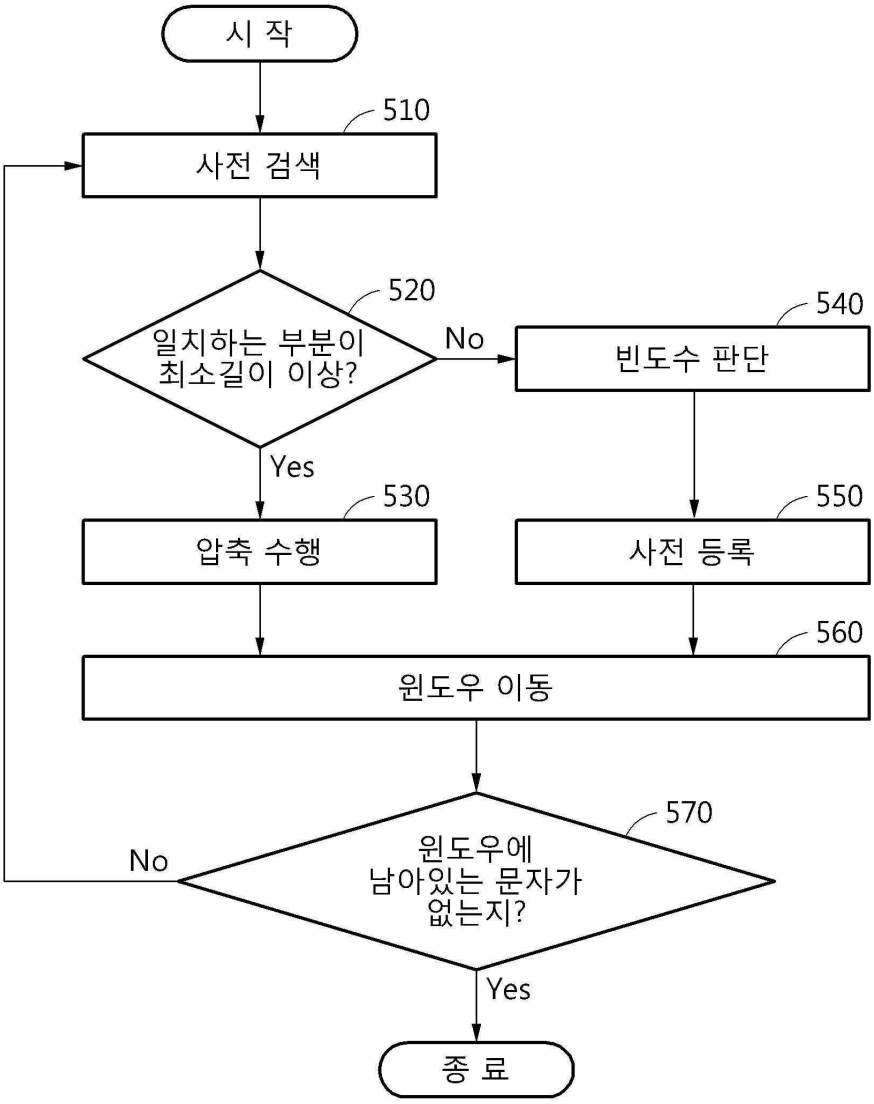
도면3


301
302
301
302

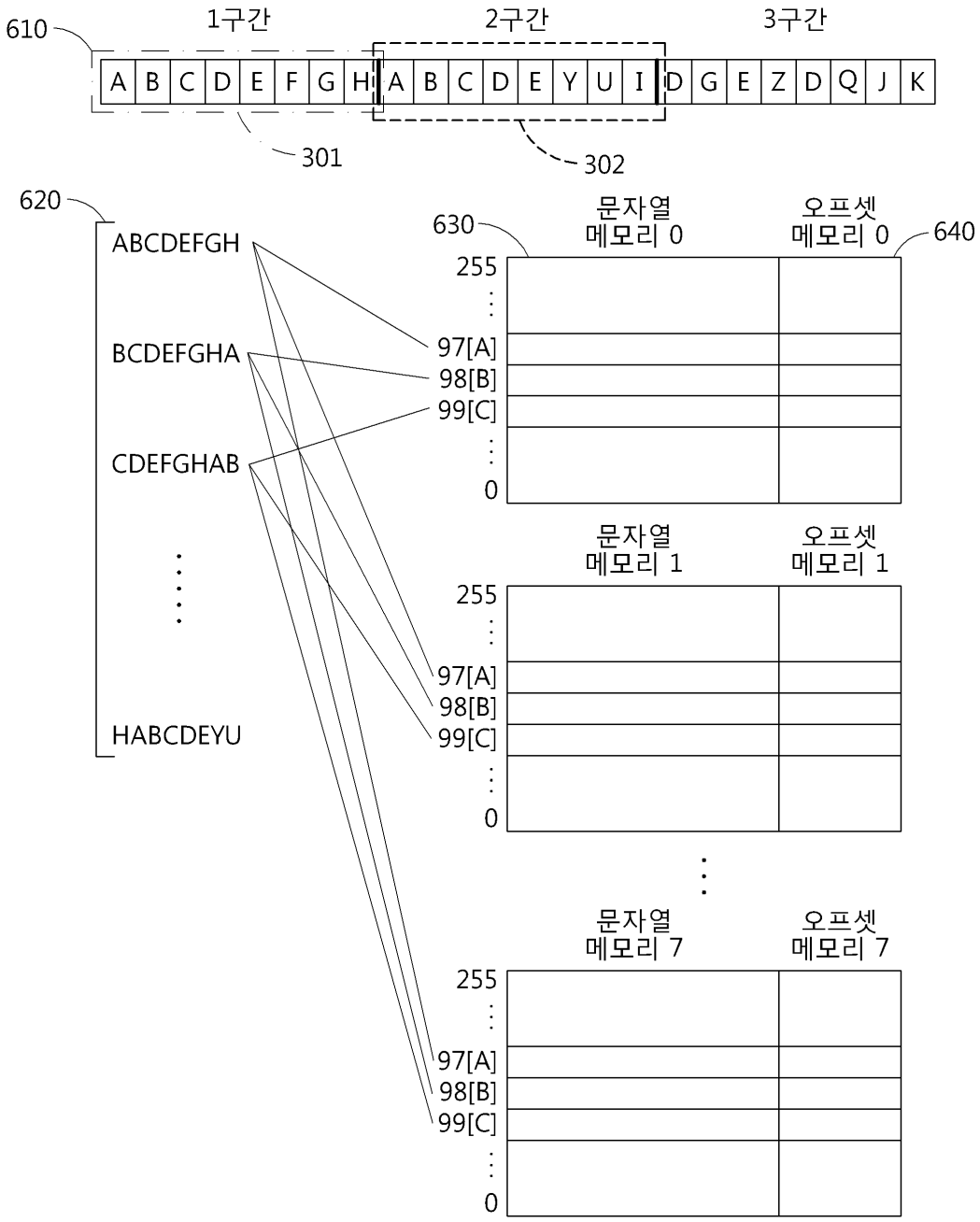
도면4


301
302
0 1
N-1 0 1
N-1
문자열 메모리 0
오프셋 메모리 0
문자열 메모리 1
오프셋 메모리 1
문자열 메모리 2
오프셋 메모리 2
문자열 메모리 N-1
오프셋 메모리 N-1
문자열 메모리
L
오프셋 메모리
L
255
97[A]
401
0
depth
402
N-1 width 0
[Bytes]
K-1 width 0
[Bits]
아스키코드표
403
A 97
B 98
C 99

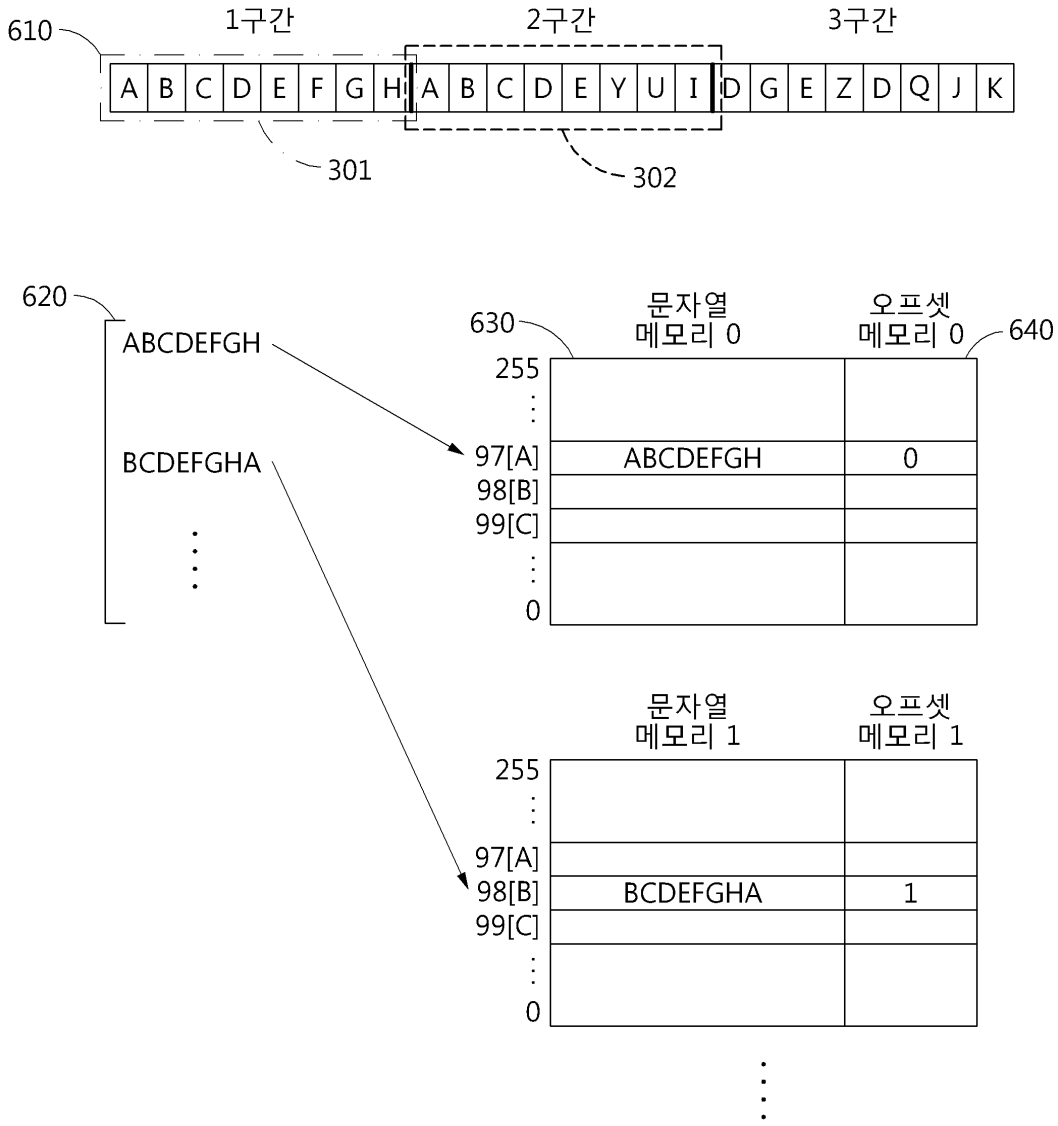
도면5


시 작
510
사전 검색
520
일치하는 부분이
최소길이 이상?
No
540
빈도수 판단
Yes
530
압축 수행
550
사전 등록
560
윈도우 이동
570
윈도우에
남아있는 문자가
없는지?
No
Yes
종 료

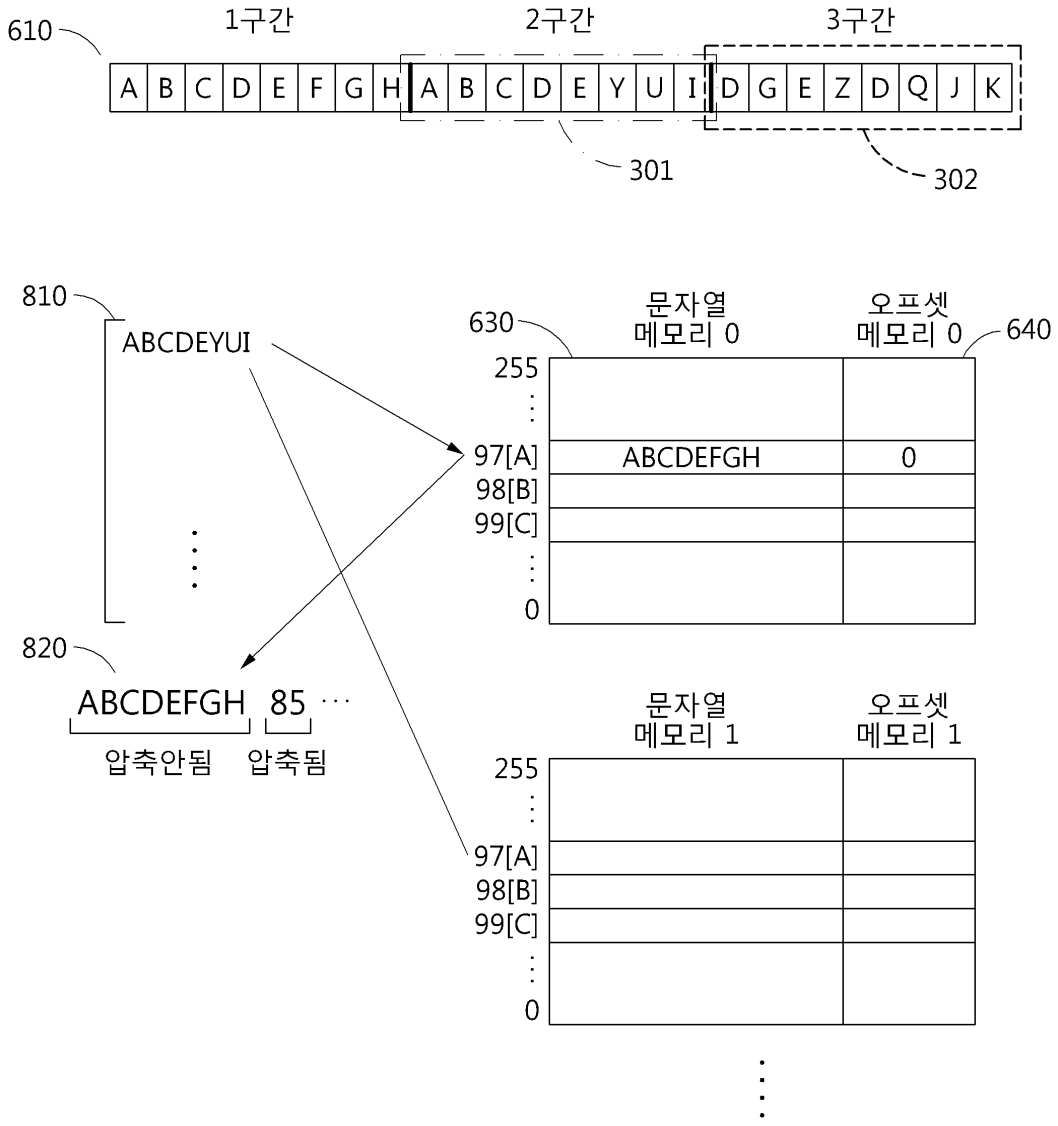
도면6


610
1구간
2구간
3구간
A B C D E F G H A B C D E Y U I D G E Z D Q J K
301
302
620
ABCDEFGH
BCDEFGHA
CDEFGHAB
HABCDEYU
630
문자열 메모리 0
오프셋 메모리 0
640
문자열 메모리 1
오프셋 메모리 1
문자열 메모리 7
오프셋 메모리 7
255
97[A]
98[B]
99[C]
0

도면7


610
1구간
2구간
3구간
A B C D E F G H A B C D E Y U I D G E Z D Q J K
301
302
620
ABCDEFGH
BCDEFGHA
630
문자열
메모리 0
오프셋
메모리 0
640
255
97[A]
98[B]
99[C]
0
ABCDEFGH
0
문자열
메모리 1
오프셋
메모리 1
BCDEFGHA
1

도면8


610
1구간
2구간
3구간
A B C D E F G H A B C D E Y U I D G E Z D Q J K
301
302
810
ABCDEYUI
820
ABCDEFGH 85 ···
압축안됨
압축됨
630
문자열
메모리 0
오프셋
메모리 0
640
255
97[A]
ABCDEFGH
0
98[B]
99[C]
0
문자열
메모리 1
오프셋
메모리 1
255
97[A]
98[B]
99[C]
0