(19)대한민국특허청(KR)
(12) 공개특허공보(A)

(51) 。Int. Cl.
H04N 7/24 (2006.01)

(11) 공개번호 10-2006-0116384
(43) 공개일자 2006년11월15일

(21) 출원번호 10-2005-0038598
(22) 출원일자 2005년05월09일

(71) 출원인 임혜숙
서울특별시 강남구 도곡동 91-5 도곡동삼성래미안아파트 106-1301

(72) 발명자 임혜숙
서울특별시 강남구 도곡동 91-5 도곡동삼성래미안아파트 106-1301

(74) 대리인 이철희
송해모

심사청구 : 있음

(54) 균형 이진 검색 트리를 이용한 허프만 디코딩 방법 및디코딩 장치

요약

본 발명은, 코드워드(Codeword) 중 일부 길이를 먼저 검색하게 하기 위해 포인터 필드, 엔트리 수 필드 및 길이 필드를 포함하는 레인지 테이블; 상기 레인지 테이블과 연결되어 상기 레인지 테이블에 저장된 코드워드 심볼을 제외한 코드워드 필드, 상기 코드워드의 길이 필드 및 심볼 필드를 포함하는 디코딩 테이블에서 균형 이진 검색 트리(Balanced Search Tree)를 이용하여 허프만 디코딩을 수행하는 방법에 있어서, (a) 입력 비트 스트림 중 기설정된 레인징 코드 길이의 레인징 비트 스트림을 상기 레인지 테이블의 인덱스로 사용하여 상기 레인지 테이블의 엔트리에 접근하는 단계; (b) 상기 인덱스의 상기 엔트리 수 필드 값이 0이면 상기 인덱스의 상기 포인터 필드에 저장된 심볼로 디코딩하는 단계; 및 (c) 상기 인덱스의 상기 엔트리 수 필드 값이 0이 아니면 상기 인덱스의 상기 포인터 필드 값을 인덱스로 가지는 상기 디코딩 테이블의 시작 엔트리부터 상기 인덱스의 상기 엔트리 수 필드 값만큼 떨어진 엔트리 사이에서, 상기 입력 비트 스트림과 상기 코드워드 필드에 저장된 값을 비교하여 이진 검색(Binary Search)을 수행하고, 이진 검색 수행 결과 일치되는 필드에 저장된 심볼로 디코딩하는 단계를 포함하는 것을 특징으로 하는 균형 이진 검색 트리를 이용한 허프만 디코딩 방법 및 디코딩 장치에 관한 것이다.

대표도

도 3

색인어

균형 이진 검색 트리, 허프만 디코딩, 레인지 테이블, 디코딩 테이블, 심볼

명세서

도면의 간단한 설명

도 1은 코드워드를 이진 검색 트리로 구성한 것,

도 2는 균형 이진 검색 트리를 만드는 알고리즘을 이용하여 균형 이진 검색 트리의 생성예를 나타낸 것,

도 3은 본 발명의 바람직한 제 1 실시예에 따른 균형 이진 검색 트리를 이용한 허프만 디코딩 장치를 개략적으로 나타낸 블록 구성도,

도 4는 본 발명의 바람직한 제 1 실시예에 따른 균형 이진 검색 트리를 이용한 허프만 디코딩 과정을 나타낸 순서도,

도 5는 본 발명의 바람직한 제 2 실시예에 따른 테이블 구조를 나타낸 도면,

도 6은 본 발명의 바람직한 제 2 실시예에 따른 균형 이진 검색 트리를 이용한 허프만 디코딩 장치를 개략적으로 나타낸 블록 구성도,

도 7은 본 발명의 바람직한 제 2 실시예에 따른 균형 이진 검색 트리를 이용한 허프만 디코딩 과정을 나타낸 순서도,

도 8은 본 발명의 바람직한 제 2 실시예에 따른 레인지 테이블 및 디코딩 테이블을 사용하였을 때 구성되는 균형 이진 검색 트리의 예를 나타낸 것,

도 9는 실험 영상 Peppers를 이용해 레인지 테이블의 엔트리 수에 대한 메모리 사용과 디코딩 시 메모리 접근 횟수를 비교한 그래프이다.

<도면의 주요 부분에 대한 부호의 설명>

300: 비교부 310: 디코딩부

500: 레인지 테이블 510: 디코딩 테이블

600: 메모리

620: 디코딩부

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 균형 이진 검색 트리(Balanced Binary Search Tree)를 이용한 허프만 디코딩 방법 및 디코딩 장치에 관한 것이다. 더욱 상세하게는, 기존의 이진 검색 트리와 달리 비어있는 노드를 포함하지 않는 새로운 방식의 균형 이진 검색 트리를 구성하여 허프만 디코딩을 수행하는 허프만 디코딩 방법 및 디코딩 장치에 관한 것이다. 또한 이에 더하여, 트리를 여러 개로 나누어 보다 짧은 트리로 만들기 위한 레인지 테이블(Ranging Table)을 이용함으로써 발생 빈도가 높은 짧은 길이의 코드워드(Codeword)를 효율적으로 디코딩하기 위한 디코딩 방법 및 디코딩 장치에 관한 것이다.

허프만 코드(Huffman Code)는 디지털 전송에서 평균 부호의 길이를 가장 짧게 할 수 있는 가변 길이 부호(Variable Length Code)의 일종으로, 구현이 간단하면서 압축 효율이 높은 장점으로 인해 1952년에 발견된 이래로 다양한 데이터의 압축에 이용되고 있으며, 특히 이미지나 비디오 압축 표준인 JPEG, MPEG-1, MPEG-2, MPEG-4 등에서 널리 사용되고 있다.

어떤 심볼 세트(Set)에 대하여 출현 빈도에 관한 확률 분포가 주어졌음을 가정할 때, 허프만 코드는 출현 빈도가 높은 심볼은 짧은 길이의 코드워드로 표현하고, 출현 빈도가 낮은 심볼은 긴 길이의 코드워드로 표현하여 평균 전송 데이터의 양을 줄이는 것을 목적으로 한다. 심볼을 같은 길이의 코드워드로 표현한 경우에는 연속적으로 들어오는 입력 비트 스트림에 대

해 일정한 길이로 잘라서 디코딩하면 되지만, 길이가 다른 경우에는 가드 비트(Guard Bit)가 없으므로 하나의 코드워드는 하나의 심볼에만 매핑(Mapping)될 수 있도록 하여야 한다. 이는, 임의의 코드워드가 다른 코드워드의 프리픽스가 되지 않도록 함으로서 가능한데, 허프만 코드는 이러한 원리를 사용하여 만들어진 코드이다.

이러한 코드워드의 특징으로 인하여 허프만 인코딩에서는 이진 검색 트리 형태를 주로 이용한다. 이진 검색 트리를 만드는 방법은 먼저 출현 빈도가 가장 낮은 두 개의 심볼을 선택하여 노드에 매핑한 후 각 노드에 0과 1을 할당하고, 이 둘을 묶어 부모(Parent) 노드를 만드는 것이다. 두 노드의 출현 빈도의 합이 부모 노드의 출현 빈도가 되는데, 이러한 과정을 반복적으로 진행하면 그 결과로 이진 검색 트리가 형성되는 것이다. 각 심볼은 트리의 리프(Leaf) 노드에만 매핑되고, 트리의 루트로부터 리프에 이르는 패스(Path)에 따라 그 심볼의 코드워드가 정해진다.

이미지나 영상 데이터의 크기가 커지고 인터넷 애플리케이션들이 발달함에 따라 실시간 데이터의 양이 늘어나면서 메모리를 적게 사용하면서도 빠르게 디코딩을 수행할 수 있도록 하는 구조가 요구되고 있다. 이진 검색 트리 구조는 최대 코드워드 길이가 늘어날수록 트리의 깊이가 깊어지기 때문에 비어있는 내부 노드의 수가 증가하게 되는데, 비어 있는 내부 노드에도 하위 노드로 가는 포인터를 저장해야 하므로 메모리 사용에 있어 매우 비효율적이다. 또한, 디코딩에 걸리는 시간은 메모리 접근 횟수와 매우 밀접한 관계를 갖는데, 이진 검색 트리 구조는 코드워드의 길이에 따라 거쳐가야 하는 노드의 수가 정해지기 때문에 최대 디코딩 시간이 가장 긴 코드워드의 길이에 비례하게 된다. 이에 따라 좀 더 효율적으로 메모리를 사용하고, 디코딩 시간을 줄이기 위한 많은 연구가 진행되고 있다.

이하에서는 허프만 디코딩을 위해 사용되는 기존의 구조에 대해 살펴보기로 한다.

엔트로피 코딩은 확률 분포에 따라 코드 길이를 다르게 정함으로써 데이터를 압축하는 방법으로 어떤 심볼 집합 S 에 대한 엔트로피는 수학식 1로 표현된다.

$$H(S) = \sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i}$$

여기서 p_i는 집합 S에 속하는 심볼 s_i 가 일어날 확률이 되고 N은 전체 심볼의 개수를 나타낸다. 수학식 1에서 $\log_2(1/p_i)$는 심볼 s_i 에 포함된 정보의 양으로 볼 수 있는데, 다시 말해 심볼 s_i를 인코딩하는데 필요한 비트 수라 할 수 있다. 엔트로피 H(S)는 가중치를 갖는 $\log_2(1/p_i)$ 의 합이므로, 심볼 집합 S에 포함되어 있는 각 심볼들을 표현하는 데 필요한 평균 비트 수의 최소값으로 볼 수 있다.

수학식 1을 통해서 보면, 엔트로피 코딩은 확률 분포가 고르게 퍼져 있을 때보다 한쪽으로 몰려 있을 때 더 낮은 엔트로피를 가지게 됨을 알 수 있고, 이 특성을 이용하여 압축 효율을 높인 것이 바로 엔트로피 코딩이다. 허프만 코딩은 이러한 엔트로피 코딩 중 가장 널리 알려진 것이다.

표 1은 심볼의 확률 분포에 따라 코드워드를 할당한 허프만 코드의 예를 보여준다. 여기서 심볼의 확률은 2의 지수 승의 역으로 표현됨을 가정하였다.

[표 1]

심볼 Si	확률 Pi	코드워드
S_1	1/4	00
S_2	1/32	01000
S_3	1/32	01001
S_4	1/16	0101
S_4	1/64	011000
S_4	1/64	011001
S_7	1/32	01101
S_8	1/32	01110

S_7	1/32	01111
S_{10}	1/8	100
S_{11}	1/8	101
S_{12}	1/16	1100
S_{13}	1/16	1101
S_{14}	1/8	111

허프만 디코딩을 위한 가장 단순한 구조는 가장 긴 코드워드의 길이를 h라고 했을 때 2^h 크기의 룩업 테이블(Lookup Table)을 만들어 입력 값을 메모리 인덱스로 하여 검색을 수행하는 구조이다. 이 구조는 한 번의 메모리 검색으로 디코딩이 가능하나, 심볼 수에 비해 너무 큰 메모리를 필요로 하기 때문에 실용적이지 않다.

앞서 간단히 언급한 바와 같이 허프만 디코딩을 위해 가장 널리 쓰이는 것으로 이진 검색 트리를 이용한 구조를 들 수 있다.

도 1은 코드워드를 이진 검색 트리로 구성한 것이다.

도 1에서 검은색 노드가 심볼이 저장된 노드이고 흰 노드는 비어있는 노드를 나타낸다. 이진 검색 트리는, 루트 노드에서 출발하여 입력으로 들어온 비트가 0이면 왼쪽으로, 1이면 오른쪽으로 노드를 옮기는 방식으로 한 비트씩 검색을 수행하되, 리프(Leaf) 노드를 만날 때까지 검색을 진행하여 디코딩하는 구조이다. 그러나 이러한 이진 검색 트리 구조는 트리를 구성함에 있어 내부에 빈 노드가 많게 되어 메모리 효율이 떨어질 뿐 아니라, 메모리 검색 횟수가 트리의 최대 높이인 가장 긴 코드워드의 길이가 되는 단점이 있다.

한편, Level Compressed 트리(LC-tree)에서는 이진 검색 트리를 BFS(Breadth First Search) 방식으로 저장하여 효율적으로 메모리를 사용하는 알고리즘을 제안하였다. LC-tree에서는 이진 검색 트리의 루트에서 가장 짧은 코드의 길이를 d라고 할 때, d-1의 높이만큼 트리의 윗부분을 자르고 남은 트리를 앞의 노드부터 순차적으로 저장하는 코드워드 어레이를 만든다. 즉, 레벨 d부터 심볼이 저장되어 있으므로 레벨 d부터 오른쪽으로 지나가면서 코드워드 어레이를 만드는 것이다. 중간에 비어있는 내부 노드에는 점프(Jump) 값을 미리 계산하여 저장하는데, 점프 값은 자신의 첫 번째 자식(Child) 노드로 가기 위한 포인터가 된다. 심볼이 저장되어 있지 않은 내부 노드는 유효(Valid) 비트 0을 갖고, 심볼이 저장된 리프 노드는 유효 비트 1을 갖는다. 이제 코드워드 어레이를 따라가며 디코딩을 하게 되는데 먼저 코드워드의 가장 짧은 길이까지를 인덱스로 사용해 코드워드 어레이에 접근하고 저장된 값이 점프 값이라면 입력된 비트 스트림의 다음 비트에 점프 값을 더한 값을 인덱스로 사용하여 다시 코드워드 어레이에 접근하고 일치되는 심볼을 만날 때까지 이 과정을 반복한다. 이진 검색 트리 구조와 비교할 때, LC-tree 구조는 길이 d 이전 트리의 빈 노드만을 제거한 구조로서 d 레벨 이하에 있는 빈 노드들은 점프 값을 갖는 노드로서 저장되어야 하는 단점을 가진다.

또한, First-Bit-Change 구조에서는, 처음 비트가 바뀌는 위치에 따라 전체 코드워드 트리를 여러 개의 트리로 분류한 후, 분류된 각 트리에 속한 코드워드들을 같은 길이를 갖는 코드워드로 확장하여 디코딩하는 알고리즘이다. 이렇게 하면 확장된 트리에서 코드워드들의 수치적(Numerical) 값은 1씩 차이가 나므로 이 특성을 이용하여 빠르게 디코딩을 할 수 있다. First-Bit-Change 구조에서는 비트 스트림이 들어오면 한 비트씩 보다가 0에서 1 또는 1에서 0으로 바뀌는 위치에 따라 그 비트 스트림이 속한 트리가 정해진다. 해당된 트리에서 정한 길이까지의 입력을 저장된 코드워드와 비교하고, 그 차이를 인덱스로 하여 심볼이 저장된 테이블에 접근하여 디코딩을 수행한다. 이 알고리즘은 항상 두 번의 메모리 접근으로 디코딩을 수행할 수 있다는 특성을 갖지만, 코드워드들을 같은 길이로 확장해야 함에 따라 매우 큰 메모리를 필요로 한다는 문제점이 있다.

또한, 캐노니컬(Canonical) 허프만 트리 또는 SGH(Single-Side-Growing-Huffman) 트리는 이진 검색 트리의 모양이 한쪽으로 쏠리도록 코드워드를 할당한 트리이다. 즉 왼쪽으로 쏠리는 SGH 트리에서, 긴 길이의 코드워드는 항상 짧은 길이의 코드워드보다 왼쪽 노드에 위치하게 되는 구조이다. 한편, CHT(Condensed-Huffman-Table)에는 길이 정보와 각 길이에 해당하는 코드워드들 중 제일 작은 코드워드, 그리고 심볼이 저장된 또다른 테이블의 인덱스가 저장되어 있다. 입력 비트 스트림에 대해 CHT를 순차적으로 검색하여 값을 비교함으로써 길이 정보를 찾고, 일치된 길이에 저장된 코드워드 값과 입력과의 차를 메모리 포인터로 사용하여 디코딩을 수행한다. 이는 매우 효율적인 구조이나, 캐노니컬 트리의 형태로

인코딩된 경우에만 적용할 수 있다는 제약점을 가지며, 짧은 길이의 코드가 확률적으로 더 많이 나온다는 것을 이용하여 코드 길이별로 순차적인 검색을 수행하므로 최악의 경우(Worst Case) 검색 횟수는 가장 긴 코드 길이인 h에 비례한다는 단점을 지닌다.

발명이 이루고자 하는 기술적 과제

이러한 문제점을 해결하기 위해 본 발명은, 기존의 이진 검색 트리와 달리 비어있는 노드를 포함하지 않는 새로운 방식의 균형 이진 검색 트리를 구성하여 허프만 디코딩을 수행함으로써 메모리 사용 등에 있어 보다 효율적인 허프만 디코딩 방법 및 디코딩 장치를 제공하는 것을 목적으로 한다.

또한, 본 발명은 트리를 여러 개로 나누어 보다 짧은 트리로 만들기 위한 레인지 테이블(Ranging Table)을 이용함으로써 발생 빈도가 높은 짧은 길이의 코드워드(Codeword)를 효율적으로 디코딩함으로써, 검색 시간 면에 있어서 보다 유리한 허프만 디코딩 방법 및 디코딩 장치를 제공하는 것을 목적으로 한다.

발명의 구성 및 작용

본 발명의 제 1 목적에 의하면, 코드워드(Codeword) 중 일부 길이를 먼저 검색하게 하기 위해 포인터 필드, 엔트리 수 필드 및 길이 필드를 포함하는 레인지 테이블; 상기 레인지 테이블과 연결되어 상기 레인지 테이블에 저장된 코드워드 심볼을 제외한 코드워드 필드, 상기 코드워드의 길이 필드 및 심볼 필드를 포함하는 디코딩 테이블에서 균형 이진 검색 트리(Balanced Search Tree)를 이용하여 허프만 디코딩을 수행하는 방법에 있어서, (a) 입력 비트 스트림 중 기설정된 레인징 코드 길이의 레인징 비트 스트림을 상기 레인지 테이블의 인덱스로 사용하여 상기 레인지 테이블의 엔트리에 접근하는 단계; (b) 상기 인덱스의 상기 엔트리 수 필드 값이 0이면 상기 인덱스의 상기 포인터 필드에 저장된 심볼로 디코딩하는 단계; 및 (c) 상기 인덱스의 상기 엔트리 수 필드 값이 0이 아니면 상기 인덱스의 상기 포인터 필드 값을 인덱스로 가지는 상기 디코딩 테이블의 시작 엔트리부터 상기 인덱스의 상기 엔트리 수 필드 값만큼 떨어진 엔트리 사이에서, 상기 입력 비트 스트림과 상기 코드워드 필드에 저장된 값을 비교하여 이진 검색(Binary Search)을 수행하고, 이진 검색 수행 결과 일치되는 필드에 저장된 심볼로 디코딩하는 단계를 포함하는 것을 특징으로 하는 균형 이진 검색 트리를 이용한 허프만 디코딩 방법을 제공한다.

본 발명의 제 2 목적에 의하면, 균형 이진 검색 트리를 이용하여 허프만 디코딩을 수행하는 방법에 있어서, (a) 입력 비트 스트림에서, 상기 균형 이진 검색 트리에 포함된 코드워드 길이 중 최대 코드워드 길이에 해당되는 최대 비트 스트림과 상기 균형 이진 검색 트리의 루트 노드에 저장된 코드워드를 비교하는 단계; (b) 상기 단계 (a)에서의 비교 결과, 상기 최대 비트 스트림이 상기 루트 노드에 저장된 코드워드보다 작으면 상기 루트 노드의 왼편 자식(Child) 노드로 진행하여 상기 최대 비트 스트림과 상기 왼편 자식 노드에 저장된 코드워드를 비교하고, 상기 최대 비트 스트림이 상기 루트 노드에 저장된 코드워드보다 크면 상기 루트 노드의 오른편 자식 노드로 진행하여 상기 최대 비트 스트림과 상기 오른편 자식 노드에 저장된 코드워드를 비교하되, 상기 최대 비트 스트림과 일치되는 코드워드가 검색될 때까지 상기 비교를 반복하는 단계; 및 (c) 상기 단계 (b)에서의 비교 결과, 상기 최대 비트 스트림과 일치되는 코드워드가 검색되면, 일치된 코드워드로 디코딩하는 단계를 포함하는 것을 특징으로 하되, 상기 균형 이진 검색 트리는 정렬된 코드워드 리스트의 가장 중간에 위치한 코드워드가 루트로 선택되고, 상기 루트의 왼쪽 리스트의 중간 코드워드가 상기 루트의 왼편 자식(Child) 노드로, 상기 루트의 오른쪽 리스트의 중간 코드워드가 상기 루트의 오른편 자식 노드로 선택되어 구성되는 과정을 통해 생성되되, 상기 코드워드 리스트의 모든 항목이 선택될 때까지 상기 과정이 반복되어 생성되는 것을 특징으로 하고, 상기 단계 (a) 및 단계 (b)에서의 상기 비교시, 비교 대상이 되는 두 코드워드의 길이가 다른 경우에는 짧은 길이의 코드워드까지의 크기를 비교하는 것을 특징으로 하는 균형 이진 검색 트리를 이용하여 허프만 디코딩을 수행하는 방법을 제공한다.

본 발명의 제 3 목적에 의하면, 균형 이진 검색 트리를 이용하여 허프만 디코딩을 수행하는 디코딩 장치에 있어서, 코드워드(Codeword) 중 일부 길이를 먼저 검색하게 하기 위해 포인터 필드, 엔트리 수 필드 및 길이 필드를 포함하는 레인지 테이블; 상기 레인지 테이블과 연결되어 상기 레인지 테이블에 저장된 코드워드 심볼을 제외한 코드워드 필드, 상기 코드워드의 길이 필드 및 심볼 필드를 포함하는 디코딩 테이블을 저장하는 메모리; 및 입력 비트 스트림 중 기설정된 레인징 코드 길이의 레인징 비트 스트림을 상기 레인지 테이블의 인덱스로 사용하여, 상기 인덱스의 상기 엔트리 수 필드 값이 0이면 상기 인덱스의 상기 포인터 필드에 저장된 심볼로 디코딩하고, 상기 인덱스의 상기 엔트리 수 필드 값이 0이 아니면 상기 인덱스의 상기 포인터 필드 값을 인덱스로 가지는 상기 디코딩 테이블의 시작 엔트리부터 상기 인덱스의 상기 엔트리 수 필드 값만큼 떨어진 엔트리 사이에서, 상기 입력 비트 스트림과 상기 코드워드 필드에 저장된 값을 비교하여 이진 검색(Binary Search)을 수행하고, 이진 검색 수행 결과 일치되는 필드에 저장된 심볼로 디코딩하는 디코딩부를 포함하는 것을 특징으로 하는 균형 이진 검색 트리를 이용한 허프만 디코딩 장치를 제공한다.

본 발명의 제 4 목적에 의하면, 균형 이진 검색 트리를 이용하여 허프만 디코딩을 수행하는 디코딩 장치에 있어서, 입력 비트 스트림에서, 상기 균형 이진 검색 트리에 포함된 코드워드 길이 중 최대 코드워드 길이에 해당되는 최대 비트 스트림과 상기 균형 이진 검색 트리의 루트 노드에 저장된 코드워드를 비교하고; 비교 결과, 상기 최대 비트 스트림이 상기 루트 노드에 저장된 코드워드보다 작으면 상기 루트 노드의 왼편 자식(Child) 노드로 진행하여 상기 최대 비트 스트림과 상기 왼편 자식 노드에 저장된 코드워드를 비교하고, 상기 최대 비트 스트림이 상기 루트 노드에 저장된 코드워드보다 크면 상기 루트 노드의 오른편 자식 노드로 진행하여 상기 최대 비트 스트림과 상기 오른편 자식 노드에 저장된 코드워드를 비교하되, 상기 최대 비트 스트림과 일치되는 코드워드가 검색될 때까지 상기 비교를 반복하는 비교부; 및 상기 비교부에서의 비교 결과, 상기 최대 비트 스트림과 일치되는 코드워드가 검색되면, 일치된 코드워드로 디코딩하는 디코딩부를 포함하는 것을 특징으로 하되, 상기 균형 이진 검색 트리는 정렬된 코드워드 리스트의 가장 중간에 위치한 코드워드가 루트로 선택되고, 상기 루트의 왼쪽 리스트의 중간 코드워드가 상기 루트의 왼편 자식(Child) 노드로, 상기 루트의 오른쪽 리스트의 중간 코드워드가 상기 루트의 오른편 자식 노드로 선택되어 구성되는 과정을 통해 생성되되, 상기 코드워드 리스트의 모든 항목이 선택될 때까지 상기 과정이 반복되어 생성되는 것을 특징으로 하고, 상기 비교부에서의 상기 비교시, 비교 대상이 되는 두 코드워드의 길이가 다른 경우에는 짧은 길이의 코드워드까지의 크기를 비교하는 것을 특징으로 하는 균형 이진 검색 트리를 이용하여 허프만 디코딩을 수행하는 디코딩 장치를 제공한다.

이하, 본 발명의 바람직한 실시예를 첨부된 도면들을 참조하여 상세히 설명한다. 우선 각 도면의 구성요소들에 참조부호를 부가함에 있어서, 동일한 구성요소들에 대해서는 비록 다른 도면상에 표시되더라도 가능한 한 동일한 부호를 가지도록 하고 있음에 유의해야 한다. 또한, 본 발명을 설명함에 있어, 관련된 공지 구성 또는 기능에 대한 구체적인 설명이 당업자에게 자명하거나, 본 발명의 요지를 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명은 생략한다.

본 발명의 바람직한 실시예에 따른 균형 이진 검색 트리는 길이가 다른 코드워드 간의 크기 비교를 가능하게 하는 새로운 정의를 사용하여 가능하게 된다. 균형 이진 검색 트리의 구성은 코드워드들을 크기별로 정렬하는 것으로부터 출발하는데, 코드워드들의 정렬을 위하여 다음의 Compare 정의를 사용하였다.

정의 Compare
a. 두 codeword의 길이가 같으면 numerical 값이 비교된다.
Ex) A=1001 B=1100인 경우 B>A
b. 두 codeword의 길이가 다른 경우 짧은 길이까지만 비교된다.
Ex) A=1001, B=110000인 경우 B>A

이를 살펴보면, 길이가 같은 코드워드들의 크기 비교는 코드워드의 수치적(Numerical) 값이 비교되고, 길이가 다른 코드워드들은 짧은 길이까지만 비교하여, 짧은 길이까지의 수치적 값이 큰 쪽이 큰 코드워드가 되고, 작은 쪽이 작은 코드워드가 된다.

알고리즘 1은 본 발명의 바람직한 실시예에 따른 균형 이진 검색 트리를 만드는 알고리즘을 나타낸 것이고, 도 2는 알고리즘 1을 이용하여 본 발명의 바람직한 실시예에 따른 균형 이진 검색 트리의 생성예를 나타낸 것이다.

(알고리즘 1)

```
BuildTree(List)
        if List is empty, return.
        Sort(List);
        let m be the median of List
        root<-m;
        let leftList and rightList contain all elements in the left and right of m
        leftChild(root)<-BuildTree(leftList);
        rightChild(root)<-BuildTree(RightList);
        return address of root
end BuildTree;
```

알고리즘 1은 정렬 과정 후에 수행된다. 도 2의 코드워드 리스트는 본 발명의 바람직한 실시예에 따른 균형 이진 검색 트리를 만들기 위해 표 1의 코드워드들을 크기 순으로 정렬한 것이다. 이렇게 정렬된 리스트의 가장 중간에 위치한 코드워드가 루트로 선택되고, 루트의 왼쪽 리스트의 중간 코드워드가 루트의 왼편 자식(Child) 노드로, 루트의 오른쪽 리스트의 중간 코드워드가 루트의 오른편 자식 노드로 선택된다. 이러한 자식 노드에 대해서도 같은 방식으로 이들의 자식 노드를 선택하여 리스트에 있는 모든 코드워드가 선택될 때까지 반복하면 된다.

도 2에 주어진 코드워드 리스트를 예로 설명하면, 코드워드 리스트 중에서 가운데 있는 01110이 루트 노드로 선택되고 루트의 왼쪽과 오른쪽 리스트의 가운데에 있는 0101과 1100을 다음 노드로 선택되는 것을 반복하여 트리를 만들어 나가면 도 2의 균형 이진 검색 트리가 완성되는 것이다.

앞서 언급한 바와 같이, 도 2의 트리는 표 1의 코드워드들을 가지고 균형 이진 검색 트리를 구성한 예인데, 도 2와 도 1을 비교해 보면 도 2에서는 흰색으로 표시된 비어있는 내부 노드가 없고 트리의 코드워드가 균형 있게 퍼져 있는 것을 볼 수 있다. 본 발명의 바람직한 실시예에 따른 균형 이진 검색 트리는 비어있는 내부 노드가 없고 트리가 한쪽으로 치우치지 않았기 때문에 전체 트리의 높이가 6에서 3으로 줄었는데, 이는 단순한 이진 검색 트리 대신 본 발명의 바람직한 실시예에 따른 균형 이진 검색 트리를 이용하면 검색 횟수가 크게 줄어듦을 보여주는 것이다. 또한, N 개의 심볼을 갖는 세트(Set)에 대해 기존의 이진 검색 트리를 구성하기 위해 저장되어야 하는 최대 노드 수는 2N-1인 반면에, 본 발명의 바람직한 실시예에 따른 균형 이진 검색 트리의 경우는 빈 노드를 저장하지 않아도 되기 때문에 저장되어야 하는 노드의 수를 N으로 줄일 수 있게 된다. 또한, 본 발명의 바람직한 실시예에 따른 균형 이진 검색 트리는 완전 균형 트리로서 메모리 인덱스를 사용하여 검색이 수행되므로 다음 노드로 가는 포인터를 저장할 필요가 없다는 장점도 있다.

도 3은 본 발명의 바람직한 제 1 실시예에 따른 균형 이진 검색 트리를 이용한 허프만 디코딩 장치를 개략적으로 나타낸 블록 구성도이다.

도 3에 도시된 바와 같이, 본 발명의 바람직한 제 1 실시예에 따른 균형 이진 검색 트리를 이용한 허프만 디코딩 장치는 비교부(300) 및 디코딩부(310) 등을 포함할 수 있다.

본 발명의 바람직한 제 1 실시예에 따른 비교부(300)는 입력 비트 스트림에서, 전술한 본 발명의 바람직한 실시예에 따른 균형 이진 검색 트리에 포함된 코드워드 길이 중 최대 코드워드 길이에 해당되는 최대 비트 스트림과 균형 이진 검색 트리의 루트 노드에 저장된 코드워드를 비교한다. 비교 결과, 최대 비트 스트림이 루트 노드에 저장된 코드워드보다 작으면 루트 노드의 왼편 자식 노드로 진행하여 최대 비트 스트림과 왼편 자식 노드에 저장된 코드워드를 비교하고, 최대 비트 스트림이 루트 노드에 저장된 코드워드보다 크면 루트 노드의 오른편 자식 노드로 진행하여 최대 비트 스트림과 오른편 자식 노드에 저장된 코드워드를 비교한다. 본 발명의 바람직한 제 1 실시예에 따른 비교부(300)에서는 최대 비트 스트림과 일치되는 코드워드가 검색될 때까지 이러한 비교를 반복하여 수행한다.

본 발명의 바람직한 제 1 실시예에 따른 디코딩부(310)는 비교부(300)에서의 비교 결과 최대 비트 스트림과 일치되는 코드워드가 검색되면, 일치된 코드워드로 디코딩을 수행한다.

본 발명의 바람직한 제 1 실시예에 따른 균형 이진 검색 트리에서의 코드워드를 검색하는 과정은 알고리즘 2와 같다.

(알고리즘 2)

```
Search(tree, codeword)
        if tree=NIL, return NULL;
        if(codeword<tree(root)),
                symbol<-Search(leftChild(tree), codeword);
        else
                symbol<-Search(rightChild(tree), codeword).
        if codeword matches tree(root),
                symbol<-tree(Node);
                return the remaining part which is not included in symbol
                to the bit stream.
        Return symbol;
end Search;
```

우선, 루트 노드로부터 출발하여 입력된 비트 스트림을 최대 코드워드 길이로 잘라, 이 값과 현재의 노드에 저장된 코드워드 값을 비교한다. 여기서, 입력된 값과 노드에 저장된 코드워드를 비교하는 데 있어서는 전술한 Compare 정의가 사용된다. 즉, 입력된 값이 현재의 노드에 저장된 값보다 작으면 현재의 노드의 왼쪽에 있는 자식 노드로, 크면 오른쪽에 있는 자식 노드로 검색을 진행하여 비교량을 지수적으로 줄이는 검색 방법이다. 입력된 값과 현재의 노드에 저장된 코드워드의 값이 일치한다면 입력된 값에 해당하는 코드워드를 찾은 것이 되므로, 입력값 중 일치된 코드워드의 길이보다 긴 부분은 다시 입력 비트 스트림으로 되돌린 후 검색을 종료한다.

도 4는 본 발명의 바람직한 제 1 실시예에 따른 균형 이진 검색 트리를 이용한 허프만 디코딩 과정을 나타낸 순서도이다.

비트 스트림이 입력되면(S400), 본 발명의 바람직한 제 1 실시예에 따른 비교부(300)에서는 입력 비트 스트림에서 본 발명의 바람직한 실시예에 따른 균형 이진 검색 트리에 포함된 코드워드 길이 중 최대 코드워드 길이에 해당되는 최대 비트 스트림과 균형 이진 검색 트리의 루트 노드에 저장된 코드워드를 비교한다(S402).

단계 S402에서의 비교 결과, 최대 비트 스트림이 루트 노드에 저장된 코드워드보다 작으면(S404), 루트 노드의 왼편 자식 노드로 진행하여 최대 비트 스트림과 왼편 자식 노드에 저장된 코드워드를 비교한다(S406). 또한, 최대 비트 스트림이 루트 노드에 저장된 코드워드보다 크면(S408), 루트 노드의 오른편 자식 노드로 진행하여 최대 비트 스트림과 오른편 자식 노드에 저장된 코드워드를 비교한다(S410).

이러한 비교 과정은 최대 비트 스트림과 일치되는 코드워드가 검색될 때까지 반복되는데, 최대 비트 스트림과 일치되는 코드워드가 검색되면 본 발명의 바람직한 제 1 실시예에 따른 디코딩부(310)에서는 일치된 것으로 검색된 코드워드로 최대 비트 스트림을 디코딩한다(S412).

본 발명의 바람직한 제 1 실시예에 따른 균형 이진 검색 트리를 이용하면 $\log_2 N$ 만큼의 최대 검색 횟수를 갖게 된다. 그러나 입력된 값의 검색 범위를 보다 빨리 좁힐 수 있다면, 검색 성능은 더욱 향상될 수 있게 된다.

이러한 목적을 위하여 본 발명의 바람직한 제 2 실시예에서는 레인지 테이블을 제안한다. 본 발명의 바람직한 제 2 실시예에 따른 레인지 테이블이라 함은 코드워드 중 일부 길이를 먼저 검색하여 다음 디코딩 테이블로 연결시키기 위한 구조로서 레인지 테이블에서 비교하는 코드 길이를 r 이라고 했을 때 레인지 테이블은 2^r 의 크기를 갖게 된다. 허프만 디코딩은 짧은 길이의 코드워드가 빈도 수가 더 높고 그것을 빨리 찾는 것이 더욱 중요하기 때문에 본 발명의 바람직한 제 2 실시예에 따른 레인지 테이블을 이용한다면, 길이가 r보다 작거나 같은 입력이 왔을 때에는 한 번의 메모리 접근으로 디코딩을 가능하게 하여 전체 검색 성능을 더욱 향상시킨다. 한편, 길이가 r보다 긴 코드워드들은 길이가 r인 프리픽스에 따라 깊이가 작은 이진 균형 트리를 구성하게 된다.

도 5는 본 발명의 바람직한 제 2 실시예에 따른 테이블 구조를 나타낸 도면이다.

도 5에 도시된 바와 같이, 본 발명의 바람직한 제 2 실시예에서는 레인지 테이블(500)과 디코딩 테이블(510) 2 개의 테이블을 이용한다. 도 5는 전술한 표 1의 예를 들어 테이블을 구성한 것이다.

먼저, 레인지 테이블의 크기를 2^3이라고 했을 때 본 발명의 바람직한 제 2 실시예에 따른 레인지 테이블(500)은 도 5의 참조부호 500과 같이 구성된다. 도 5에 도시된 바와 같이, 본 발명의 바람직한 제 2 실시예에 따른 레인지 테이블(500)은 포인터 필드(502), 엔트리 수 필드(504) 및 길이 필드(506) 등을 포함한다.

레인지 테이블(500)의 길이인 3 보다 긴 코드워드들을 가리키는 엔트리(인덱스 2, 3, 6)에는, 다음 테이블로 연결되는 포인터와 같은 범위의 프리픽스를 갖는 엔트리가 몇 개인지가 저장되고, 레인지 테이블(500)의 길이인 3보다 작은 코드워드가 가리키는 엔트리(인덱스 0, 1, 4, 5, 7)에는 심볼 자체가 저장된다.

또한, 본 발명의 바람직한 제 2 실시예에 따른 디코딩 테이블(510)은 도 5의 참조부호 510과 같이 구성된다. 도 5에 도시된 바와 같이, 본 발명의 바람직한 제 2 실시예에 따른 디코딩 테이블(510)은 허프만 코드 필드(512), 길이 필드(514) 및 심볼 필드(516) 등을 포함한다.

본 발명의 바람직한 제 2 실시예에 따른 디코딩 테이블(510)에는 레인지 테이블(500)에 저장된 코드워드를 제외한 코드워드와 그 길이, 그리고 심볼이 저장된다.

표 1 및 도 5를 참조하면, 코드워드의 프리픽스가 010인 코드워드는 S_2부터 S_4까지 3개이므로, 레인지 테이블(500)의 세 번째 엔트리의 포인터 필드(502)에는 0이, 엔트리 수 필드(504)에는 3이 저장되고, 디코딩 테이블(510)에는 S_2부터 S_4가 차례로 저장된다.

한편, 코드워드의 길이가 레인지 테이블(500)의 길이인 r보다 작거나 같은 심볼은 레인지 테이블(500)에만 저장되는데, 이 경우 포인터 필드(502)에 심볼 자체를 저장하고 심볼인지 포인터인지는 엔트리 수 필드(504)가 0인지 아닌지로 구분한다. 도 5를 참조하면, 코드워드의 길이가 레인지 테이블(500)의 길이인 3보다 작은 S_1의 경우, 길이를 3으로 확장하여 두 개의 엔트리에 저장되고 코드워드의 길이가 3보다 작기 때문에 일부의 길이는 다음 코드워드가 될 수 있으므로 길이 필드(506)에 저장된 길이를 제외한 나머지 비트 스트림을 다시 입력으로 되돌려준다.

도 6은 본 발명의 바람직한 제 2 실시예에 따른 균형 이진 검색 트리를 이용한 허프만 디코딩 장치를 개략적으로 나타낸 블록 구성도이다.

도 6에 도시된 바와 같이, 본 발명의 바람직한 제 2 실시예에 따른 균형 이진 검색 트리를 이용한 허프만 디코딩 장치는 메모리(600) 및 디코딩부(620) 등을 포함할 수 있다.

본 발명의 바람직한 제 2 실시예에 따른 메모리(600)는 전술한 레인지 테이블(500) 및 디코딩 테이블(510) 등을 저장한다. 즉, 입력된 코드워드 중 일부 길이를 먼저 검색하게 하기 위해 포인터 필드, 엔트리 수 필드, 길이 필드 등을 포함하는 레인지 테이블(500) 및 레인지 테이블(500)과 연결되어 레인지 테이블(500)에 저장된 코드워드 심볼을 제외한 코드워드 필드, 코드워드의 길이 필드, 심볼 필드 등을 포함하는 디코딩 테이블(510)을 저장한다.

본 발명의 바람직한 제 2 실시예에 따른 디코딩부(620)는 입력 비트 스트림 중 기설정된 레인징 코드 길이의 레인징 비트 스트림을 레인지 테이블(500)의 인덱스로 사용하여, 엔트리 수 필드(504) 값이 0이면 인덱스의 포인터 필드(502)에 저장된 심볼로 디코딩하고, 엔트리 수 필드(504) 값이 0이 아니면 인덱스의 포인터 필드(502)의 값을 인덱스로 가지는 디코딩 테이블(510)의 시작 엔트리부터 인덱스의 엔트리 수 필드(506) 값만큼 떨어진 엔트리 사이에서, 입력 비트 스트림과 코드워드 필드에 저장된 값을 비교하여 이진 검색(Binary Search)을 수행한다. 디코딩부(620)에서는 이러한 이진 검색 수행 결과 일치되는 필드에 저장된 심볼로 디코딩한다.

도 7은 본 발명의 바람직한 제 2 실시예에 따른 균형 이진 검색 트리를 이용한 허프만 디코딩 과정을 나타낸 순서도이다.

본 발명의 바람직한 제 2 실시예에 따른 디코딩부(620)에서는 입력 비트 스트림 중 기설정된 레인징 코드 길이의 레인징 비트 스트림을 상기 레인지 테이블(500)의 인덱스로 사용하여 레인지 테이블(500)에 접근한다(S700).

다음으로, 본 발명의 바람직한 제 2 실시예에 따른 디코딩부(620)에서는 해당 인덱스의 엔트리 수 필드(504) 값을 참조하여(S702) 엔트리 수 필드(504) 값이 0이면 레인지 테이블(500)에서 인덱스의 포인터 필드(502)에 저장된 심볼로 바로 디코딩한다(S704). 레인지 테이블(500)을 참조한 결과, 엔트리 수 필드(504)가 0이라는 것은 포인터 필드(502)에 저장된 것이 심볼 그 자체라는 의미가 되므로 한 번의 메모리 접근으로 입력이 디코딩되는 것이다.

만약, 엔트리 수 필드(504) 값이 0이 아니면, 레인지 테이블(500)에서 인덱스의 포인터 필드(502) 값을 인덱스로 가지는 디코딩 테이블(510)의 시작 엔트리부터 인덱스의 엔트리 수 필드(506) 값만큼 떨어진 엔트리 사이에서, 디코딩부(620)는 입력 비트 스트림과 코드워드 필드에 저장된 값을 비교하는 이진 검색을 수행한다(S706). 다음으로 디코딩부(620)에서는 이러한 이진 검색 결과 일치되는 필드에 저장된 심볼로 디코딩을 수행한다(S708).

도 8은 본 발명의 바람직한 제 2 실시예에 따른 레인지 테이블(500) 및 디코딩 테이블(510)을 사용하였을 때 구성되는 균형 이진 검색 트리의 예를 나타낸 것이다.

예를 들어, 입력으로 011111000… 의 비트 스트림이 들어온다면 먼저 레인지 테이블(500)에 접근하는 코드워드 길이인 011, 즉 십진수 3을 인덱스로 하여 레인지 테이블(500)에 접근하게 된다. 이 엔트리에는 포인터 필드(502)에 3이 엔트리 수 필드(504)에는 5가 저장되어 있으므로, 디코딩 테이블(510)의 인덱스 3부터 7 사이에서 이진 검색이 진행되어야 함을 알 수 있고, 입력 011111이 메모리 인덱스 5에 저장되어 있는 01101과 비교된다. 입력 값이 더 크므로 다음은 디코딩 테이블(510)의 인덱스 7에 저장되어 있는 01111과 비교하고, 이와 일치하므로 S_7가 출력되고 검색이 종료된다. 이미 디코딩

된 5비트를 제외한 다음 비트 스트림을 디코딩하기 위해 다시 3비트인 100, 즉 십진수 4를 인덱스로 하여 레인지 테이블(500)에 접근하면, 이 경우 레인지 테이블(500)의 엔트리 수 필드(504) 값이 0이므로, 포인터 필드(502)에는 심볼 자체가 저장되어 있음을 알 수 있고, 심볼 S_{10} 이 출력된다.

이에 본 출원인은 본 발명의 바람직한 제 2 실시예에 따른 허프만 디코딩 구조의 성능을 평가하기 위해서 Peppers, Lena, Barbara, Zelda 4개의 Gray-Scale 이미지의 차이 영상을 이용하여 실험하였다. 차이 영상 d(x, y)는 I(x, y)를 원영상이라 했을 때, 수학식 2와 같이 표현된다.

$$d(x,y) = I(x,y) - I(x-1,y)$$

수학식 2를 참조하면, 원영상보다 Gray Scale이 특정 값에 집중되므로, 보다 적은 엔트로피를 갖게 되고 따라서 JPEG 등에서 이용하는 허프만 코드와 더 가까운 결과를 얻을 수 있게 된다. 각 차이 영상의 크기는 512 x 512이고 각 픽셀은 256 level의 Gray-Scale로 표현되는데, Gray-Scale의 출현 빈도를 가지고 허프만 인코딩을 수행하면 표 2와 같은 특성을 갖는다.

[표 2]

	Peppers	Lena	Barbara	Zelda
No. of 심볼	256	256	256	246
Tree depth	14	15	15	18
Min. code length	4	4	4	4

도 9는 실험 영상 Peppers를 이용해 레인지 테이블(500)의 엔트리 수에 대한 메모리 사용과 디코딩 시 메모리 접근 횟수를 비교한 그래프이다.

실험 영상의 코드워드는 가장 짧은 길이의 코드워드가 4이므로 레인지 테이블(500)의 크기를 2^4부터 2^{10}까지 다르게 하여 실험하였다. 그래프를 보면 디코딩에 필요한 메모리 접근 횟수는 지수적으로 감소하며, 메모리 사용량은 선형적으로 증가하는 것을 알 수 있다. 레인지 테이블(500)의 크기를 가장 짧은 코드 길이인 2^4에서 한 단계 증가시켜 2^5로 한다면 디코딩 시 필요한 메모리 접근 횟수는 크게 줄어드는 것에 반하여 필요한 메모리 크기는 약간 증가하게 된다. 이러한 실험 결과를 바탕으로 디코딩 시 필요한 메모리 횟수와 저장해야 할 엔트리 수의 트레이드 오프(Trade Off) 관계에서 적합한 지점을 택해 레인지 테이블(500)의 크기를 정하면 된다. 이하에서는 다른 알고리즘과의 성능 비교를 위해 r이 5인 경우의 실험 결과를 이용하였다.

표 3은 본 발명의 바람직한 제 2 실시예에 따른 구조와 종래 기술에서 설명한 5개의 다른 구조에 대하여 각 구조에서 필요로 하는 메모리 크기를 나타낸 것이다.

[표 3]

	Peppers	Lena	Barbara	Zelda
본 발명	**274**	**275**	**276**	**363**
이진 검색 트리	511	511	511	491
LC-tree	496	496	496	476
First-bit-change	145,145	354,096	428,667	5,139,333
*SGH-tree	6,319	5,744	4,474	3,658
*CHT	267	267	267	260

전술한 바와 같이, 이진 검색 트리는 비어있는 내부 노드를 모두 저장해야 하므로 2N-1 만큼의 엔트리가 필요하다. LC-tree는 이진 검색 트리에서 깊이가 d-1까지의 부분을 없애 (2N-1)-(2^d-1) 의 엔트리를 요구하므로, 표 3에서 보인 값과 같은 결과를 보인다. 전술한 바와 같이 First-Bit-Change 구조는 실험 결과 확장되는 엔트리의 수가 매우 커져 실용적이지 못한 것을 볼 수 있다. 또한, SGH-tree 구조와 CHT 구조는 주어진 영상을 Canonical 형태로 다시 인코딩 한 후 실험한 것으로서, SGH-tree 구조의 경우 First-Bit-Change 구조보다는 메모리 사용량이 줄었으나 여전히 큰 메모리를 요구함을 알 수 있다. CHT 구조는 메모리 사용에 있어 매우 우수한 성질을 가지나, Canonical 형태로 인코딩되어 있는 허프만 코드에만 적용될 수 있다는 단점을 갖는다. 본 발명의 바람직한 제 2 실시예에 따른 구조는 빈 노드를 저장하지 않으므로 메모리 사용량에 있어 Canonical 제약점을 갖지 않는 구조 중에 가장 우수함을 알 수 있다.

표 4는 디코딩 시에 필요한 메모리 접근 횟수를 비교한 것이다.

[표 4]

	Peppers	Lena	Barbara	Zelda
본 발명	**632,323**	**702,100**	**732,563**	**609,088**
이진 검색 트리	1,494,024	1,526,741	1,630,367	1,449,043
LC-tree	707,592	740,309	843,934	662,611
First-bit-change	524,288	524,288	524,288	524,288
*SGH-tree	524,288	524,288	524,288	524,288
*CHT	969,710	1,002,379	1,106,038	924,743

LC-tree는 전술한 바와 같이 빈 노드에 점프(Jump) 값을 미리 계산하여 저장해 두어야 한다는 단점을 갖는다. 이진 검색 트리는 트리의 깊이가 4가 되는 지점에서 처음으로 리프 노드가 나타남에도 심볼이 없는 내부 노드를 다 따라가야 하므로 전체 접근 횟수가 매우 크게 나타난다. First-Bit-Change 구조와 SGH-tree 구조는 심볼 당 항상 두 번의 메모리 접근이 필요하므로, 이미지 크기가 정해지면 필요로 하는 메모리 접근 횟수가 정하여져서 가장 좋은 디코딩 성능을 보이나 앞서 설명했듯이 이 구조는 메모리 크기가 타 구조에 비해 월등히 많이 필요하다는 단점이 있다. 본 발명의 바람직한 제 2 실시예에 따른 레인지 테이블(500)의 크기를 2^7로 하면 디코딩 시 접근 횟수 성능은 위의 두 구조보다 좋아지면서 저장해야 할 엔트리 수는 크게 늘지 않으므로 훨씬 효율적임을 알 수 있다. 본 발명의 바람직한 제 2 실시예에 따른 구조는 영상에서의 Gray-Scale 출현 빈도에 따르는 허프만 코드의 효율성을 매우 잘 반영하고 있으므로 매우 우수한 성능을 보임을 알 수 있다.

표 5는 픽셀 하나를 디코딩하는데 필요한 최소, 최대, 그리고 평균 메모리 접근 횟수를 나타내고 있다.

[표 5]

	Peppers			Lena			Barbara			Zelda		
	Min	Max	Avr	Min	Max	Avr	Min	Max	Avr	Min	Max	Avr
본 발명	**1**	**5**	**2.41**	**1**	**6**	**2.68**	**1**	**5**	**2.79**	**1**	**6**	**2.32**
이진 검색 트리	4	14	7.56	4	15	7.47	4	15	7.31	4	18	7.31
LC-tree	1	10	2.70	1	11	2.82	1	11	3.22	1	15	2.53
First-bit-change	2	2	2	2	2	2	2	2	2	2	2	2
*SGH-tree	2	2	2	2	2	2	2	2	2	2	2	2
*CHT	2	12	3.67	2	12	3.82	2	12	4.22	2	17	3.53

본 발명의 바람직한 제 2 실시예에 따른 구조에서, 코드워드 길이가 d인 심볼은 한 번의 메모리 접근으로 디코딩할 수 있고 그 이상의 길이를 가지는 코드워드는 빈 노드 없는 균형 이진 검색 트리를 구성하게 되므로 최대 5~6번의 메모리 접근으

로 디코딩이 가능하다. 종래의 이진 검색 트리는 루트부터 차례대로 트리를 따라가는 구조이므로 최소, 최대 횟수가 트리의 깊이와 비례하여 나타나게 되고, LC-tree는 이진 검색 트리의 리프 노드가 나타나기 전의 비어있는 내부 노드들을 없앤 것으로 이진 검색 트리에 비해 d 만큼의 접근 횟수가 줄게 되지만, 그 아랫부분에 있는 빈 노드들을 따라가야 하므로 본 발명의 바람직한 제 2 실시예에 따른 구조에 비해 최대 메모리 접근 횟수가 크게 나타났다. 앞서 설명한 바와 같이 First-Bit-Change 구조와 SGH-tree 구조의 메모리 접근 횟수는 항상 2이다. CHT 구조는 코드워드가 가지는 길이의 종류(중간에 없는 길이가 있을 수 있으므로 트리의 깊이가 아님)에 비례하여 순차적 검색 횟수가 증가하고, 심볼이 있는 메모리에 다시 접근해야 하므로 한 번의 추가적인 메모리 접근이 더 필요하게 된다.

이상의 설명은 본 발명을 예시적으로 설명한 것에 불과한 것으로, 본 발명이 속하는 기술분야에서 통상의 지식을 가지는 자라면 본 발명의 본질적인 특성에서 벗어나지 않는 범위에서 다양한 변형이 가능할 것이다. 따라서, 본 명세서에 개시된 실시예들은 본 발명을 한정하기 위한 것이 아니라 설명하기 위한 것이고, 이러한 실시예에 의하여 본 발명의 사상과 범위가 한정되는 것은 아니다. 본 발명의 범위는 아래의 청구범위에 의하여 해석되어야 하며, 그와 동등한 범위 내에 있는 모든 기술은 본 발명의 권리범위에 포함되는 것으로 해석되어야 할 것이다.

발명의 효과

이상에서 설명한 바와 같이 본 발명에 의하면, 허프만 디코딩을 위해 비어 있는 노드를 포함하지 않는 균형 이진 검색 트리 구조를 이용함으로써 비어있는 내부 노드를 포함하지 않게 되므로 메모리 사용량에 있어 매우 우수한 구조를 제공할 뿐만 아니라, 트리의 형태가 완전 균형 구조를 갖기 때문에 디코딩 시간에 있어서도 매우 효율적이라는 장점이 있다.

즉, 본 발명에 의한 균형 이진 검색 트리 구조는 비어있는 내부 노드가 없고 트리가 한쪽으로 치우치지 않기 때문에 전체 트리의 높이가 줄어들므로 단순한 이진 검색 트리와 비교할 때 검색 횟수가 크게 줄어든다. 또한, N 개의 심볼을 갖는 세트(Set)에 대해 기존의 이진 검색 트리를 구성하기 위해 저장되어야 하는 최대 노드 수는 2N-1인 반면에, 본 발명의 바람직한 실시예에 따른 균형 이진 검색 트리의 경우는 빈 노드를 저장하지 않아도 되기 때문에 저장되어야 하는 노드의 수를 N으로 줄일 수 있게 된다. 또한, 본 발명의 바람직한 실시예에 따른 균형 이진 검색 트리는 완전 균형 트리로서 메모리 인덱스를 사용하여 검색이 수행되므로 다음 노드로 가는 포인터를 저장할 필요가 없다는 장점도 있다.

또한, 본 발명에 의하면 코드워드 중 일부 길이를 먼저 검색할 수 있게 하는 레인지 테이블을 이용함으로써 길이가 레인지 테이블 크기보다 작거나 같은 입력이 왔을 때 한 번에 디코딩을 가능하게 하므로 전체 검색 성능을 향상시키는 장점이 있다.

(57) 청구의 범위

청구항 1.

코드워드(Codeword) 중 일부 길이를 먼저 검색하게 하기 위해 포인터 필드, 엔트리 수 필드 및 길이 필드를 포함하는 레인지 테이블; 상기 레인지 테이블과 연결되어 상기 레인지 테이블에 저장된 코드워드 심볼을 제외한 코드워드 필드, 상기 코드워드의 길이 필드 및 심볼 필드를 포함하는 디코딩 테이블에서 균형 이진 검색 트리(Balanced Search Tree)를 이용하여 허프만 디코딩을 수행하는 방법에 있어서,

(a) 입력 비트 스트림 중 기설정된 레인징 코드 길이의 레인징 비트 스트림을 상기 레인지 테이블의 인덱스로 사용하여 상기 레인지 테이블의 엔트리에 접근하는 단계;

(b) 상기 인덱스의 상기 엔트리 수 필드 값이 0이면 상기 인덱스의 상기 포인터 필드에 저장된 심볼로 디코딩하는 단계; 및

(c) 상기 인덱스의 상기 엔트리 수 필드 값이 0이 아니면 상기 인덱스의 상기 포인터 필드 값을 인덱스로 가지는 상기 디코딩 테이블의 시작 엔트리부터 상기 인덱스의 상기 엔트리 수 필드 값만큼 떨어진 엔트리 사이에서, 상기 입력 비트 스트림과 상기 코드워드 필드에 저장된 값을 비교하여 이진 검색(Binary Search)을 수행하고, 이진 검색 수행 결과 일치되는 필드에 저장된 심볼로 디코딩하는 단계

를 포함하는 것을 특징으로 하는 균형 이진 검색 트리를 이용한 허프만 디코딩 방법.

청구항 2.

제 1 항에 있어서,

상기 레인지 테이블과 상기 디코딩 테이블은 상기 균형 이진 검색 트리 형태를 가지되, 상기 균형 이진 검색 트리는 정렬된 코드워드 리스트의 가장 중간에 위치한 코드워드가 루트로 선택되고, 상기 루트의 왼쪽 리스트의 중간 코드워드가 상기 루트의 왼편 자식(Child) 노드로, 상기 루트의 오른쪽 리스트의 중간 코드워드가 상기 루트의 오른편 자식 노드로 선택되어 구성되는 과정을 통해 생성되되, 상기 코드워드 리스트의 모든 항목이 선택될 때까지 상기 과정이 반복되어 생성되는 것을 특징으로 하는 균형 이진 검색 트리를 이용한 허프만 디코딩 방법.

청구항 3.

제 1 항에 있어서,

상기 단계 (c)에서,

상기 입력 비트 스트림과 상기 코드워드 필드에 저장된 값의 비교시, 비교 대상이 되는 상기 입력 비트 스트림과 상기 코드워드 필드에 저장된 값의 길이가 다른 경우에는 짧은 길이의 코드워드까지의 크기를 비교하는 것을 특징으로 하는 균형 이진 검색 트리를 이용한 허프만 디코딩 방법.

청구항 4.

제 1 항에 있어서,

상기 레인지 테이블에서, 상기 레인지 테이블의 길이인 기설정된 상기 레인징 코드 길이보다 긴 길이의 코드워드를 가리키는 엔트리에 있어서, 상기 포인트 필드에는 상기 디코딩 테이블로 연결되는 포인터가 저장되고 상기 엔트리 수 필드에는 상기 레인징 코드값이 같은 엔트리의 개수가 저장되는 것을 특징으로 하는 균형 이진 검색 트리를 이용한 허프만 디코딩 방법.

청구항 5.

제 1 항에 있어서,

상기 레인지 테이블에서, 상기 레인지 테이블의 길이인 기설정된 상기 레인징 코드 길이보다 짧거나 같은 길이의 코드워드를 가리키는 엔트리에 있어서, 상기 포인터 필드에는 상기 심볼 자체가 저장되고 상기 엔트리 수 필드에는 0이 저장되며 상기 길이 필드에는 상기 심볼의 길이가 저장되는 것을 특징으로 하는 균형 이진 검색 트리를 이용한 허프만 디코딩 방법.

청구항 6.

제 1 항에 있어서,

상기 단계 (c) 수행 후, 상기 입력 비트 스트림 중 일치된 코드워드를 제외한 나머지 비트 스트림을 상기 입력 비트 스트림으로 하여 상기 단계 (a) 내지 단계 (c)를 반복 수행하는 것을 특징으로 하는 균형 이진 검색 트리를 이용한 허프만 디코딩 방법.

청구항 7.

제 1 항에 있어서,

상기 단계 (b)에서,

상기 인덱스의 상기 길이 필드에 저장된 길이를 제외한 나머지 비트 스트림을 입력으로 되돌려주는 것을 특징으로 하는 균형 이진 검색 트리를 이용한 허프만 디코딩 방법.

청구항 8.

균형 이진 검색 트리를 이용하여 허프만 디코딩을 수행하는 방법에 있어서,

(a) 입력 비트 스트림에서, 상기 균형 이진 검색 트리에 포함된 코드워드 길이 중 최대 코드워드 길이에 해당되는 최대 비트 스트림과 상기 균형 이진 검색 트리의 루트 노드에 저장된 코드워드를 비교하는 단계;

(b) 상기 단계 (a)에서의 비교 결과, 상기 최대 비트 스트림이 상기 루트 노드에 저장된 코드워드보다 작으면 상기 루트 노드의 왼편 자식(Child) 노드로 진행하여 상기 최대 비트 스트림과 상기 왼편 자식 노드에 저장된 코드워드를 비교하고, 상기 최대 비트 스트림이 상기 루트 노드에 저장된 코드워드보다 크면 상기 루트 노드의 오른편 자식 노드로 진행하여 상기 최대 비트 스트림과 상기 오른편 자식 노드에 저장된 코드워드를 비교하되, 상기 최대 비트 스트림과 일치되는 코드워드가 검색될 때까지 상기 비교를 반복하는 단계; 및

(c) 상기 단계 (b)에서의 비교 결과, 상기 최대 비트 스트림과 일치되는 코드워드가 검색되면, 일치된 코드워드로 디코딩하는 단계

를 포함하는 것을 특징으로 하되,

상기 균형 이진 검색 트리는 정렬된 코드워드 리스트의 가장 중간에 위치한 코드워드가 루트로 선택되고, 상기 루트의 왼쪽 리스트의 중간 코드워드가 상기 루트의 왼편 자식(Child) 노드로, 상기 루트의 오른쪽 리스트의 중간 코드워드가 상기 루트의 오른편 자식 노드로 선택되어 구성되는 과정을 통해 생성되되, 상기 코드워드 리스트의 모든 항목이 선택될 때까지 상기 과정이 반복되어 생성되는 것을 특징으로 하고,

상기 단계 (a) 및 단계 (b)에서의 상기 비교시, 비교 대상이 되는 두 코드워드의 길이가 다른 경우에는 짧은 길이의 코드워드까지의 크기를 비교하는 것을 특징으로 하는 균형 이진 검색 트리를 이용하여 허프만 디코딩을 수행하는 방법.

청구항 9.

제 8 항에 있어서,

상기 단계 (c) 수행 후, 상기 입력 비트 스트림 중 상기 일치된 코드워드를 제외한 나머지 비트 스트림을 상기 입력 비트 스트림으로 하여 상기 단계 (a) 내지 단계 (c)를 반복 수행하는 것을 특징으로 하는 균형 이진 검색 트리를 이용하여 허프만 디코딩을 수행하는 방법.

청구항 10.

균형 이진 검색 트리를 이용하여 허프만 디코딩을 수행하는 디코딩 장치에 있어서,

코드워드(Codeword) 중 일부 길이를 먼저 검색하게 하기 위해 포인터 필드, 엔트리 수 필드 및 길이 필드를 포함하는 레인지 테이블; 상기 레인지 테이블과 연결되어 상기 레인지 테이블에 저장된 코드워드 심볼을 제외한 코드워드 필드, 상기 코드워드의 길이 필드 및 심볼 필드를 포함하는 디코딩 테이블을 저장하는 메모리; 및

입력 비트 스트림 중 기설정된 레인징 코드 길이의 레인징 비트 스트림을 상기 레인지 테이블의 인덱스로 사용하여, 상기 인덱스의 상기 엔트리 수 필드 값이 0이면 상기 인덱스의 상기 포인터 필드에 저장된 심볼로 디코딩하고, 상기 인덱스의 상기 엔트리 수 필드 값이 0이 아니면 상기 인덱스의 상기 포인터 필드 값을 인덱스로 가지는 상기 디코딩 테이블의 시작 엔트리부터 상기 인덱스의 상기 엔트리 수 필드 값만큼 떨어진 엔트리 사이에서, 상기 입력 비트 스트림과 상기 코드워드 필드에 저장된 값을 비교하여 이진 검색(Binary Search)을 수행하고, 이진 검색 수행 결과 일치되는 필드에 저장된 심볼로 디코딩하는 디코딩부

를 포함하는 것을 특징으로 하는 균형 이진 검색 트리를 이용한 허프만 디코딩 장치.

청구항 11.

제 10 항에 있어서,

상기 레인지 테이블과 상기 디코딩 테이블은 상기 균형 이진 검색 트리 형태를 가지되, 상기 균형 이진 검색 트리는 정렬된 코드워드 리스트의 가장 중간에 위치한 코드워드가 루트로 선택되고, 상기 루트의 왼쪽 리스트의 중간 코드워드가 상기 루트의 왼편 자식(Child) 노드로, 상기 루트의 오른쪽 리스트의 중간 코드워드가 상기 루트의 오른편 자식 노드로 선택되어 구성되는 과정을 통해 생성되되, 상기 코드워드 리스트의 모든 항목이 선택될 때까지 상기 과정이 반복되어 생성되는 것을 특징으로 하는 균형 이진 검색 트리를 이용한 허프만 디코딩 장치.

청구항 12.

제 10 항에 있어서,

상기 디코딩부에서 상기 입력 비트 스트림과 상기 코드워드 필드에 저장된 값을 비교할 때, 비교 대상이 되는 상기 입력 비트 스트림과 상기 코드워드 필드에 저장된 값의 길이가 다른 경우에는 짧은 길이의 코드워드까지의 크기를 비교하는 것을 특징으로 하는 균형 이진 검색 트리를 이용한 허프만 디코딩 장치.

청구항 13.

제 10 항에 있어서,

상기 레인지 테이블에서, 상기 레인지 테이블의 길이인 기설정된 상기 레인징 코드 길이보다 긴 길이의 코드워드를 가리키는 엔트리에 있어서, 상기 포인트 필드에는 상기 디코딩 테이블로 연결되는 포인터가 저장되고 상기 엔트리 수 필드에는 상기 레인징 코드값이 같은 엔트리의 개수가 저장되는 것을 특징으로 하는 균형 이진 검색 트리를 이용한 허프만 디코딩 장치.

청구항 14.

제 10 항에 있어서,

상기 레인지 테이블에서, 상기 레인지 테이블의 길이인 기설정된 상기 레인징 코드 길이보다 짧거나 같은 길이의 코드워드를 가리키는 엔트리에 있어서, 상기 포인터 필드에는 상기 심볼 자체가 저장되고 상기 엔트리 수 필드에는 0이 저장되며 상기 길이 필드에는 상기 심볼의 길이가 저장되는 것을 특징으로 하는 균형 이진 검색 트리를 이용한 허프만 디코딩 장치.

청구항 15.

균형 이진 검색 트리를 이용하여 허프만 디코딩을 수행하는 디코딩 장치에 있어서,

입력 비트 스트림에서, 상기 균형 이진 검색 트리에 포함된 코드워드 길이 중 최대 코드워드 길이에 해당되는 최대 비트 스트림과 상기 균형 이진 검색 트리의 루트 노드에 저장된 코드워드를 비교하고; 비교 결과, 상기 최대 비트 스트림이 상기 루트 노드에 저장된 코드워드보다 작으면 상기 루트 노드의 왼편 자식(Child) 노드로 진행하여 상기 최대 비트 스트림과 상기 왼편 자식 노드에 저장된 코드워드를 비교하고, 상기 최대 비트 스트림이 상기 루트 노드에 저장된 코드워드보다 크면 상기 루트 노드의 오른편 자식 노드로 진행하여 상기 최대 비트 스트림과 상기 오른편 자식 노드에 저장된 코드워드를 비교하되, 상기 최대 비트 스트림과 일치되는 코드워드가 검색될 때까지 상기 비교를 반복하는 비교부; 및

상기 비교부에서의 비교 결과, 상기 최대 비트 스트림과 일치되는 코드워드가 검색되면, 일치된 코드워드로 디코딩하는 디코딩부

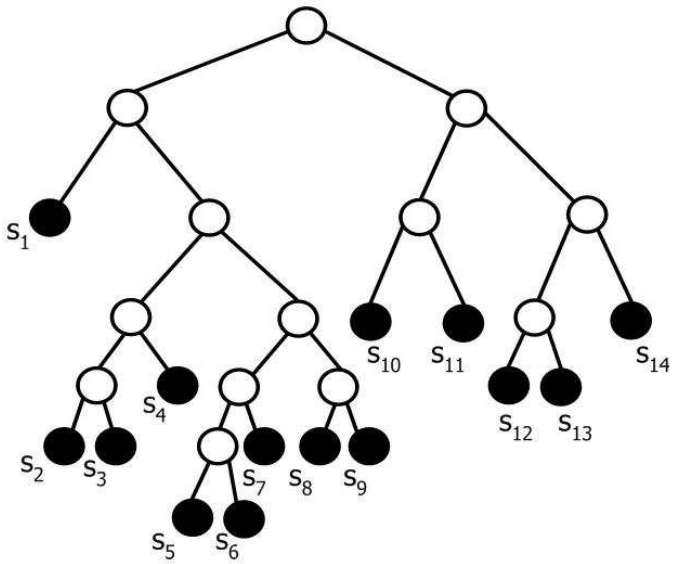
를 포함하는 것을 특징으로 하되,

상기 균형 이진 검색 트리는 정렬된 코드워드 리스트의 가장 중간에 위치한 코드워드가 루트로 선택되고, 상기 루트의 왼쪽 리스트의 중간 코드워드가 상기 루트의 왼편 자식(Child) 노드로, 상기 루트의 오른쪽 리스트의 중간 코드워드가 상기 루트의 오른편 자식 노드로 선택되어 구성되는 과정을 통해 생성되되, 상기 코드워드 리스트의 모든 항목이 선택될 때까지 상기 과정이 반복되어 생성되는 것을 특징으로 하고,

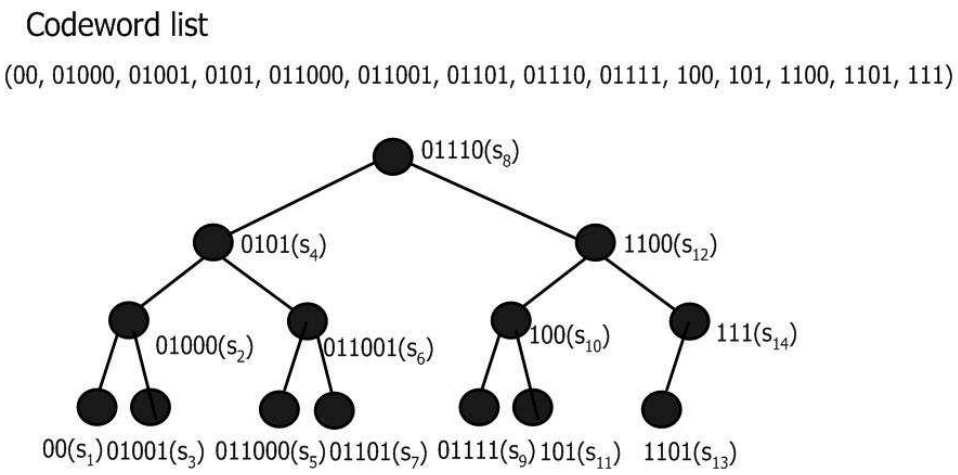
상기 비교부에서의 상기 비교시, 비교 대상이 되는 두 코드워드의 길이가 다른 경우에는 짧은 길이의 코드워드까지의 크기를 비교하는 것을 특징으로 하는 균형 이진 검색 트리를 이용하여 허프만 디코딩을 수행하는 디코딩 장치.

도면

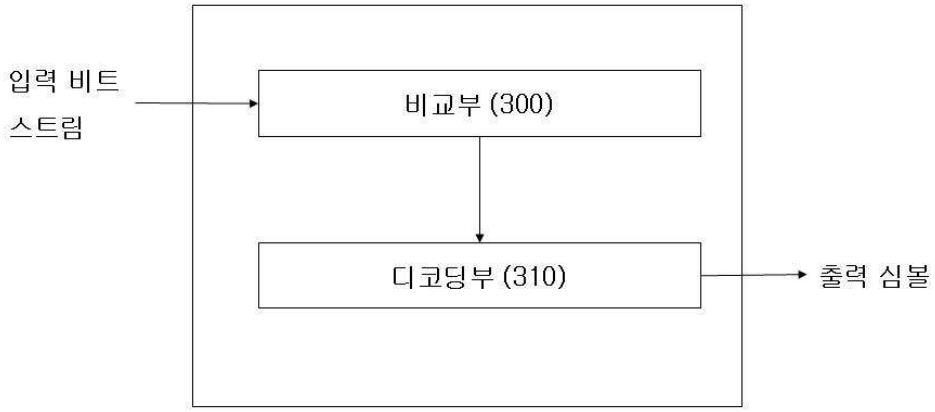
도면1



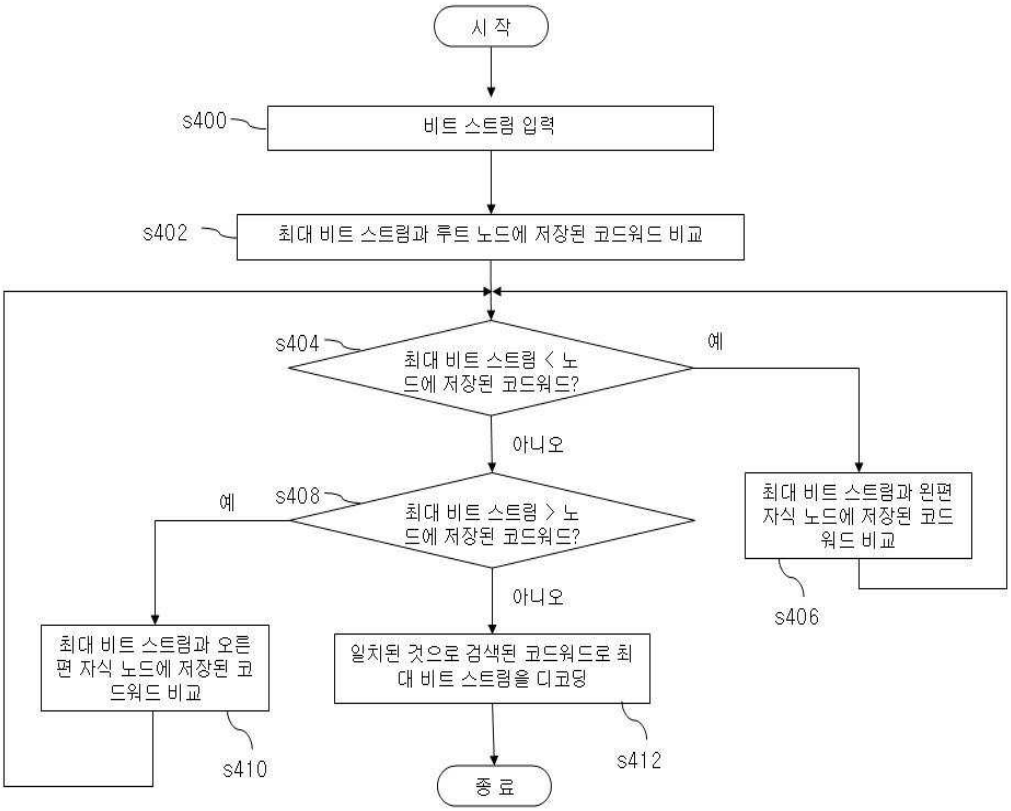
도면2


Codeword list
(00, 01000, 01001, 0101, 011000, 011001, 01101, 01110, 01111, 100, 101, 1100, 1101, 111)
01110(s8)
0101(s4)
1100(s12)
01000(s2)
011001(s6)
100(s10)
111(s14)
00(s1) 01001(s3) 011000(s5) 01101(s7) 01111(s9) 101(s11) 1101(s13)

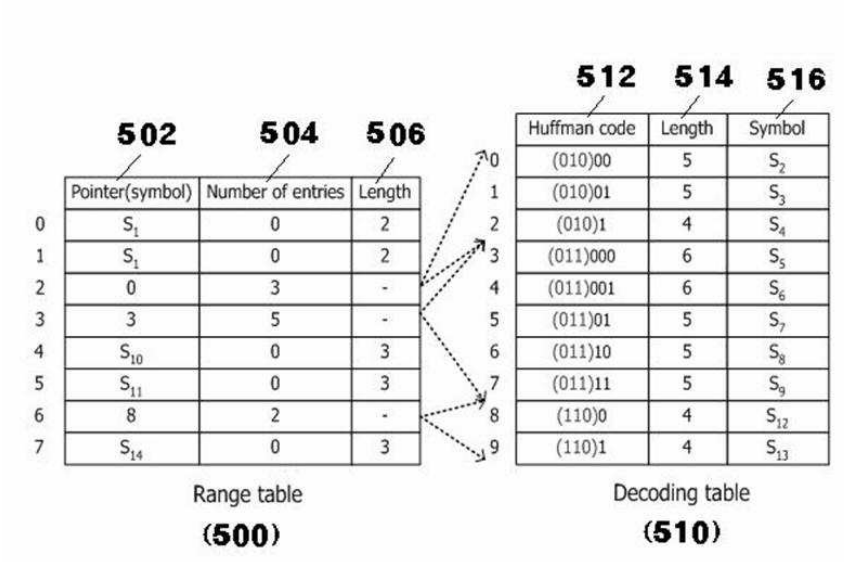
도면3


입력 비트
스트림
비교부 (300)
디코딩부 (310)
출력 심볼

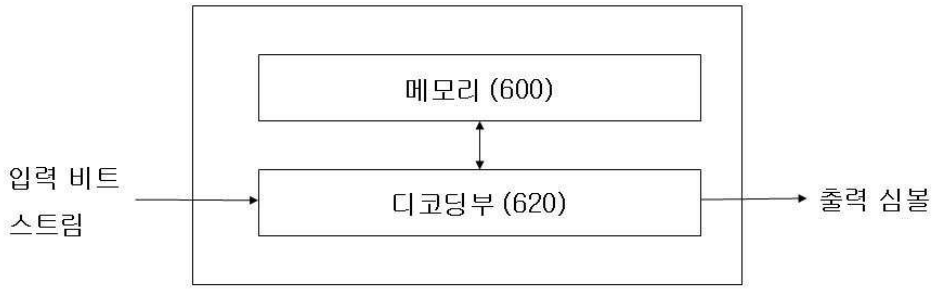
도면4


시작
s400
비트 스트림 입력
s402
최대 비트 스트림과 루트 노드에 저장된 코드워드 비교
s404
최대 비트 스트림 < 노드에 저장된 코드워드?
예
아니오
최대 비트 스트림과 왼편 자식 노드에 저장된 코드워드 비교
s406
s408
최대 비트 스트림 > 노드에 저장된 코드워드?
예
아니오
최대 비트 스트림과 오른편 자식 노드에 저장된 코드워드 비교
s410
일치된 것으로 검색된 코드워드로 최대 비트 스트림을 디코딩
s412
종료

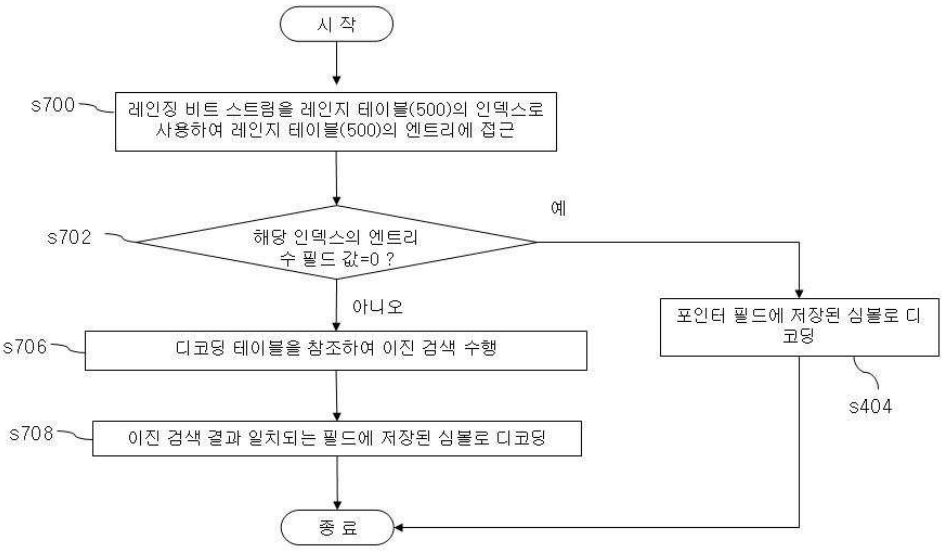
도면5


502
504
506
Pointer(symbol)
Number of entries
Length
0 S1 0 2
1 S1 0 2
2 0 3 -
3 3 5 -
4 S10 0 3
5 S11 0 3
6 8 2 -
7 S14 0 3
Range table
(500)
512
514
516
Huffman code
Length
Symbol
0 (010)00 5 S2
1 (010)01 5 S3
2 (010)1 4 S4
3 (011)000 6 S5
4 (011)001 6 S6
5 (011)01 5 S7
6 (011)10 5 S8
7 (011)11 5 S9
8 (110)0 4 S12
9 (110)1 4 S13
Decoding table
(510)

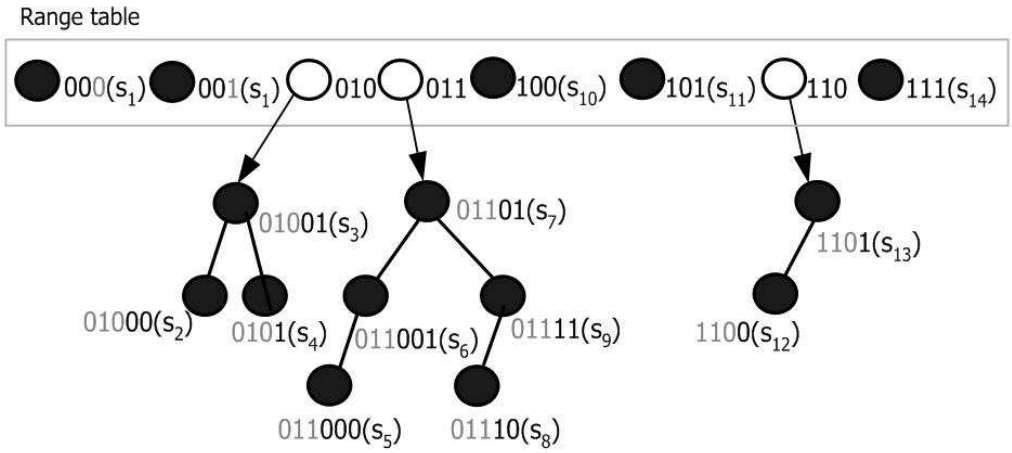
도면6


메모리 (600)
입력 비트
스트림
디코딩부 (620)
출력 심볼

도면7


시 작
s700
레인징 비트 스트림을 레인지 테이블(500)의 인덱스로 사용하여 레인지 테이블(500)의 엔트리에 접근
s702
해당 인덱스의 엔트리 수 필드 값=0 ?
예
아니오
포인터 필드에 저장된 심볼로 디코딩
s404
s706
디코딩 테이블을 참조하여 이진 검색 수행
s708
이진 검색 결과 일치되는 필드에 저장된 심볼로 디코딩
종 료

도면8


Range table
000(s1)
001(s1)
010
011
100(s10)
101(s11)
110
111(s14)
01001(s3)
01101(s7)
1101(s13)
01000(s2)
0101(s4)
011001(s6)
01111(s9)
1100(s12)
011000(s5)
01110(s8)

도면9

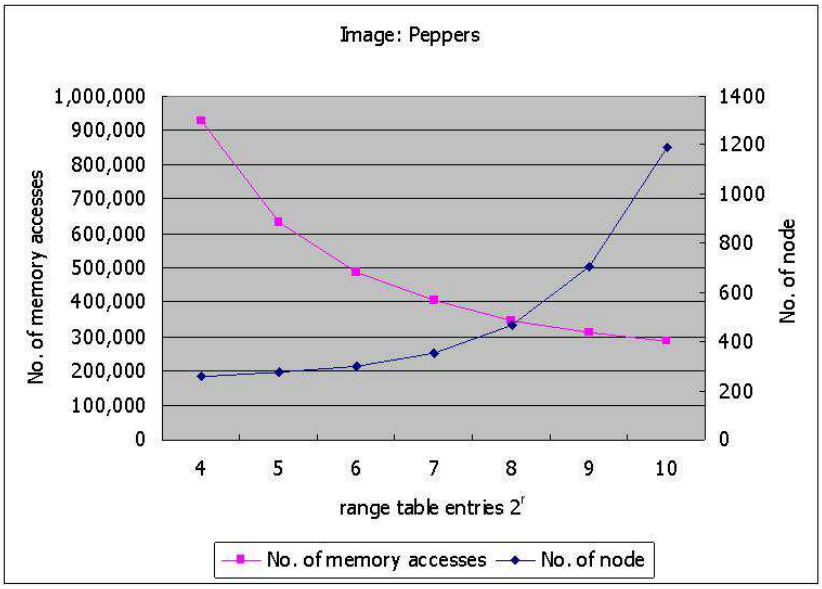


Image: Peppers
No. of memory accesses
No. of node
1,000,000
900,000
800,000
700,000
600,000
500,000
400,000
300,000
200,000
100,000
0
1400
1200
1000
800
600
400
200
0
4
5
6
7
8
9
10
range table entries 2^r
No. of memory accesses
No. of node