ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 18-03-51

j.inst.Korean.electr.electron.eng.Vol.22,No.3,850~853,September 2018
http://dx.doi.org/10.7471/ikeee.2018.22.3.850
329

허프만 코드의 효율적인 복호화에 관한 연구

A Study on Efficient Decoding of Huffman Codes

박 상 호*★

Sangho Park*★

Abstract

In this paper, we propose a decoding method using a balanced binary tree and a canonical Huffman tree for efficient decoding of Huffman codes. The balanced binary tree scheme reduces the number of searches by lowering the height of the tree and binary search. However, constructing a tree based on the value of the code instead of frequency of symbol is a drawback of the balanced binary tree. In order to overcome these drawbacks, a balanced binary tree is reconstructed according to the occurrence probability of symbols at each level of the tree and binary search is performed for each level. We minimize the number of searches using a canonical Huffman tree to find level of code to avoid searching sequentially from the top level to bottom level.

요 약

본 논문에서는 허프만 코드의 효율적인 복호화를 위하여 균형이진 트리와 정규 허프만 트리를 이용한 복호화 방법을 제안하였다. 균형이진 트리 방식은 트리의 높이를 낮추고 이진탐색이 가능하도록 하여 검색횟수를 줄일 수 있었지만 심벌의 발생확률이 아니라 코드의 크기에 트리를 만드는 것이 단점이다. 이러한 단점을 보완하기 위하여 트리의 레벨 마다 심벌의 발생확률에 따른 균형이진 트리를 구성하고 이진탐색을 레벨 별로 수행하였다. 최상위 레벨부터 순차적으로 검색을 하지 않고 코드가 있는 레벨을 찾아 검색하기 위하여 정규 허프만 트리를 사용하여 검색횟수를 최소화하였다.

Key words : Entropy coding, Huffman decoding, Canonical tree, Balanced binary tree, Balanced binary trie

Ⅰ. 서론

허프만 코딩[1]은 엔트로피 코딩으로 심벌의 발생확률에 따라 코드의 길이가 다른 가변 길이 코드를 발생하는 무손실 압축방식이다. 허프만 코딩은 비교적 간단 방법으로 높은 압축률을 얻을 수 있어 여러 분야에서 많이 사용되고 있다. 일반적으로 부호화는 한번 하지만 복호화는 사용할 때마다 수행되므로 부호화 속도보다 복호화 속도가 더 중요하다. 허프만 코딩으로 압축된 데이터에서 원 데이터를 복원하기 위한 효율적인 복호화 방법들이 연구되어 왔다[2],[3]. 허프만 트리의 오버헤드는 압축된 데이터의 크기가 크면 압축률에 영향을 끼치지 않으나 압축된 데이터의 크기가 작으면 압축률에 큰 영향을 줄 수 있어 오버헤드의 크기를 줄이는 것도 중요하다. [4]에서는 허프만 트리에서 이진 코드의 크기에 따라 이진 탐색이 가능한 균형이진 트리(balanced binary tree)를 제안하였다. 균형이진 트

* Dept. of Information and Communication Engineering, Andong National University
★ Corresponding author :
E-mail : spark@anu.ac.kr, Tel : +82-54-820-5748
※ This work was supported by a Research Grant of Andong National University
Manuscript received Jul. 27, 2018; revised Aug. 5, 2018; Accepted Aug. 8, 2018

리는 빈 노드를 포함하지 않으므로 트리의 레벨 수를 감소시킴으로써 코드를 검색하는 횟수를 줄일 수 있었으나 코드의 크기에 따라 이진 탐색을 하므로 길이가 짧은 코드의 검색횟수가 긴 코드보다 더 많아지는 경우가 생겨 검색횟수의 최적화가 이루어지지 않았다. [5]에서는 균형이진 트리[4]를 같은 레벨의 코드마다 따로 적용하여 심벌을 검색하는 균형이진 트라이(balanced binary trie) 구조를 제안하여 균형이진 트리보다 검색횟수를 줄였다. 그러나 균형이진 트리와 같이 코드의 크기에 따라 균형이진 트리를 구성하였고 허프만 트리의 레벨을 차례대로 지나가며 검색하므로 검색횟수가 추가적으로 증가하는 단점이 존재한다.

본 논문에서는 정규 허프만 트리[3], 균형이진 트리[4], 균형이진 트라이[5]의 개념을 사용하여 허프만 코드를 더 효율적으로 복호화 하기위한 방안을 제안하였다. 허프만 코드의 레벨마다 코드의 크기가 아니라 심벌의 발생확률을 이용하여 균형이진 트리를 구성하였고 정규 허프만 트리의 특성을 이용하여 입력 비트스트림이 허프만 코드의 어느 레벨에 있는지 알 수 있도록 하여 최상위 레벨부터 검색하지 않고 코드가 존재하는 레벨의 균형이진 트리만 검색할 수 있도록 하여 효율적인 검색이 가능하도록 하였다.

II. 균형이진 트리를 이용한 복호화

표 1에 14개 심벌을 위한 허프만 코드의 예를 보였다[4]. 그림 1은 표 1의 코드북을 허프만 트리로 구성한 것이다. 정규 허프만 트리는 각 레벨의 코드의 크기가 왼쪽 노드에서 오른쪽 노드로 갈수록 큰 이진 트리이다[3]. 그림 2는 그림 1의 허프만 트리를 정규 허프만 트리로 재구성한 것이다. [4]에서는 복호화 시 검색횟수를 줄이기 위하여 이진 트리의 중간노드가 없이 모든 노드에 코드값을 할당한 이진 균형 트리를 제안하였다. 그림 1에서 허프만 코드의 크기가 중앙값인 코드 S_8을 루트 노드로 하고, 루트의 왼쪽에 있는 노드들 $S_1 \sim S_7$은 루트보다 크기가 작은 코드 들인데 루트의 왼쪽에 있는 코드 값 중 중앙값을 갖는 코드 S_4를 왼쪽 자식 노드로 선택한다. 루트의 오른쪽 자식 노드에는 루트의 오른쪽에 있는 코드 값 중 가운데 값을 갖는 코드 S_{12}를 할당한다. 같은 방법으로 각 노드의 왼쪽 자식 노드는 현재 노드의 왼쪽에 있는 코드들 중 가운데 값을 갖는 코드로 할당하고 오른쪽 자식 노드도 같은 방법으로 코드를 할당하며 이러한 과정을 모든 코트가 다 할당될 때까지 반복한다. 일반적으로 N개의 심벌을 허프만 트리로 구성하면 최대 $2N-1$개의 노드가 필요하나 균형이진 트리에서는 N개의 노드만 필요하다. 그림 2에 표1의 허프만 코드에 대한 균형이진 트리를 보였다[4].

Table 1. Huffman codes for fourteen symbols[4]
표 1. 14개의 심벌을 위한 허프만 코드

Symbol	code	Symbol	code
S1	00	S8	01110
S2	01000	S9	01111
S3	01001	S10	100
S4	0101	S11	101
S5	011000	S12	1100
S6	011001	S13	1101
S7	01101	S14	111

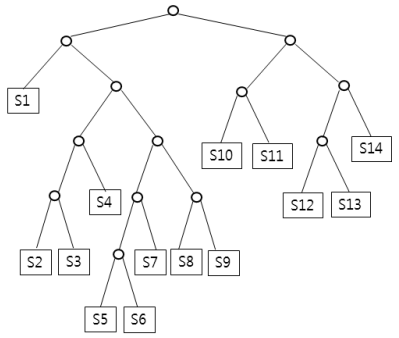

Fig. 1. Huffman tree for fourteen symbols[4].
그림 1. 14개의 심벌을 위한 허프만 트리

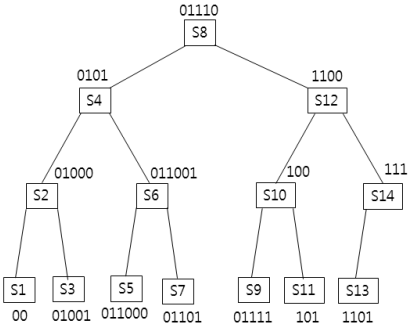

Fig. 2. Binary balanced tree for fourteen symbols[4].
그림 2. 14개의 심벌을 위한 이진 균형 트리

균형이진 트리에서 입력된 코드와 루트에 저장된 코드의 크기를 비교하여 크기가 다르면 아래 레벨에서 계속 검색하는데 입력된 코드가 루트에 있는 코드보다 작으면 왼쪽 자식 노드로 가고 크면 오른쪽 자식 노드로 가서 크기를 비교한다. 크기가 같은 노드가 나올때까지 이진 탐색과정과 동일하게 위 과정을 반복한다. 실험결과 사용된 메모리 크기는 균형이진 트리의 메모리 크기가 허프만 이진 트리의 53% 정도이고 전체 메모리 검색횟수는 이진 트리의 43% 정도라고 보고되어 있다[4].

균형이진 트리는 코드값이 중간값부터 검색을 시작하여 이진 탐색 알고리즘을 수행하므로 발생확률이 가장 높은 심벌의 코드가 작은 값으로 할당되어 발생확률이 높음에도 불구하고 검색횟수가 많아지므로 메모리 접근횟수를 감소시키지 못하게 되며. 심벌 중 최대 발생확률을 갖는 심벌의 확률이 높으면 클수록 메모리 접근횟수를 증가시킨다. 이러한 단점을 극복하기 위하여 [5]에서는 허프만 트리의 레벨마다 균형이진 트리를 구성하여 상위 레벨에서 심벌을 검색하고 없으면 다음 레벨의 코드를 이용하여 균형이진 트리를 구성하고 심별을 검색하는 균형이진 트라이(balanced binary trie) 구조를 제안하였다. 허프만 트리는 발생확률이 높은 심벌은 상위 레벨에 발생확률이 낮은 심벌은 하위 레벨에 위치하게 되므로 균형이진 트라이 구조를 이용하면 균형이진 트리보다 검색횟수를 줄일 수 있으나 균형이진 트라이도 코드의 크기를 이용하여 이진 탐색을 하므로 코드값이 작은 코드 즉 심벌의 발생확률이 높은 코드가 검색횟수가 확률이 낮은 심벌보다 길어지는 경우가 생기며, 상위 레벨부터 하위 레벨로 트리를 순차적으로 검색할 때 중간 레벨에서 추가적인 검색이 필요하므로 검색횟수를 최적화하지 못한다. 균형이진 트라이 구조를 이용한 경우 균형이진 트리를 이용한 복호화보다 메모리 검색횟수가 1/3로 줄었다고 보고되어 있다[5].

III 정규 허프만 트리를 이용한 균형이진 트라이 검색

균형이진 트리[4]는 허프만 이진 트리에 비해 검색횟수를 줄일 수 있었으나 심벌을 검색하는 과정이 코드의 가운데 값부터 시작하여 발생확률이 높은 심벌이 확률이 낮은 심벌보다 검색횟수가 많아지는 경우가 생기며, 균형이진 트라이[5]는 허프만 트리의 레벨마다 균형이진 트리를 구성하여 심벌을 검색하여 균형이진 트리 복호화[4]보다 검색횟수를 줄일 수 있었으나 균형이진 트라이도 코드의 크기에 따라 이진 탐색을 하였고 최상위 레벨부터 심벌이 있는 레벨까지 순차적으로 검색하므로 추가적인 검색이 발생하는 단점들을 개선하기 위하여 본 논문에서는 심벌의 발생확률에 따라 검색하는 균형이진 트리와 코드가 위치하고있는 레벨에서 바로 검색하기 위하여 정규 허프만 트리를 이용한 검색방법을 사용하였다.

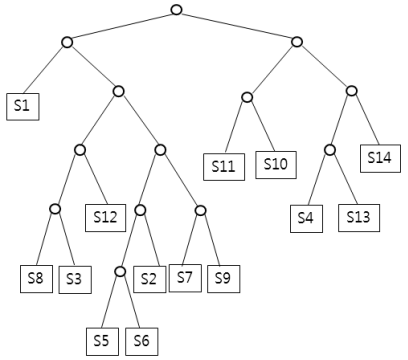

Fig. 3. Reconstructed Huffman tree based on probability.
그림 3. 발생확률에 따라 재구성된 허프만 트리

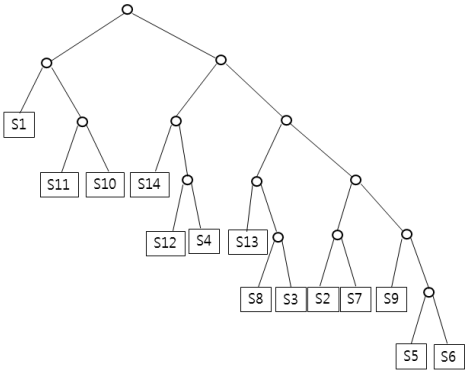

Fig. 4. Canonical Huffman tree of Fig. 3.
그림 4. 그림 3의 정규 허프만 트리

그림 1의 허프만 트리에서 같은 레벨에 있는 심벌들은 허프만 코드의 길이가 같다. 따라서 같은 레벨에 있는 심벌들은 서로 코드를 바꾸어 사용할 수 있다. 이진 트리에서 같은 레벨에 있는 코드들을 코드의 크기별로 정렬한 다음 가운데 있는 코드를 그 레벨에서 발생확률이 가장 큰 심벌에게 할당하고 두 번째로 발생확률이 높은 심벌에게는 가운

데의 왼쪽에 있는 코드들 중 가운데 있는 코드를 할당하고 세 번째로 발생확률이 높은 심벌은 오른쪽 가운데 코드를 할당한다. 동일한 방법으로 동일한 레벨에 있는 모든 심벌에 코드가 할당될 때까지 반복하며 이 과정을 모든 레벨에 적용한다. 그림 3에 각 노드들을 심벌의 발생확률에 따라 이진탐색이 가능하도록 재 구성한 트리를 나타내었다.

그림 3의 허프만 트리의 정규 허프만 트리를 그림 4에 나타내었다. 정규 허프만 트리는 레벨마다 코드의 크기가 1씩 증가하며 값의 범위가 정해져 있다. 예를 들면 그림 4에서 최대 비트 수는 6비트이므로 레벨 3의 코드 뒤에 3개의 0을 추가하면 코드값이 각각 16, 24, 32이고 뒤에 3개의 111을 추가하면 코드값이 각각 23, 31, 39이므로 레벨 3의 코드 값의 범위는 16~39이다. 레벨 4, 5, 6의 코드 값의 범위는 각각 40~51, 52~61, 62~63이다. 이러한 성질을 이용하여 입력 비트에서 6개 비트로 코드를 만들어 크기를 계산하면 어느 코드가 어느 레벨에 속하는지 알 수가 있다. 코드가 어느 레벨에 있는지 알려지면 그 레벨에 구성된 심벌의 발생확률에 따라 만들어진 균형이진 트리로 코드를 검색하여 검색횟수를 획기적으로 줄일 수 있다. 제안한 복호화 방법을 로마자 알파벳[6]에 적용하여 성능을 평가하였다. 주어진 알파벳 A부터 Z까지의 허프만 코드의 길이는 가장 짧은 코드가 3이고 가장 긴 코드가 10이고 허프만 트리의 레벨이 10이다. [5]에서 균형이진 트라이가 [4]의 방법보다 메모리 접근횟수가 1/3 정도로 줄어들었음을 보였으므로 [5]의 방법과 검색횟수를 비교하였다. 제안된 방법에서 정규 허프만 트리를 사용하지 않고 심벌의 발생확률에 따라 구성한 균형이진 트리만 사용한 경우 두 방법의 평균 검색횟수는 [5]가 4.84이고 제안한 방법이 4.74로 제안한 방법이 3% 정도 적었다. 이는 제안된 방법은 각 레벨에서 만드는 이진 트리가 발생횟수가 높을 수로 상위 레벨에 위치하게 되어 검색횟수가 더 적게 나옴을 알 수 있다. 본 논문에서 제안한 허프만 트리를 정규 허프만 트리로 재구성한 후 균형이진 트리를 만들어 복호화한 결과 평균 검색횟수가 2.16315로 [5]의 45% 정도로 줄어들었다.

V. 결론

본 논문에서는 허프만 코드를 균형이진 트라이를 이용하여 복호화하는 방법에 정규 허프만 트리의 특성을 적용하여 트리의 상위 레벨부터 하위 레벨로 순차적으로 검색하지 않고 코드가 위치하고있는 레벨의 정보를 알아내어 그 레벨의 균형이진 트리를 이용하여 심벌을 검색하는 알고리즘을 제안하고 실험을 통하여 기존의 방법보다 검색횟수를 줄일 수 있음을 보였다.

References

[1] D. Huffman, “A method for the construction of minimum redundancy codes,” *Proc. of the IRE,* vol. 40, no. 90, pp. 1098-1101, 1952. DOI:110.1007/BF02839372

[2] L. Larmore and D. Hirschberg, “Efficient decoding of prefix codes,” *Comm. of the ACM,* vol. 33, pp. 449-459, 1990. DOI:10.1145/77556.77566

[3] S. Park, “Efficient Huffman decoding using canonical Huffman tree,” *J. of KSCI,* vol.12, no.4, pp. 111-117, 2007. (DOI없음)

[4] H. Kim, Y. Jung, C. Yim, and H. Lim, “A balanced binary, search tree for Huffman decoding,” *J. of KICS,* vol. 30, no. 5C, pp. 382-390, 2005. (DOI없음)

[5] I. Park, H. Lee, and J. Yi, “Optimizing the Huffman Decoding for Embedded System,” *J. of KIISE: Computing Practices and Letters,* vol. 18, no 12, pp.881-885, 2012. (논문 검색이 안됨)

[6] Y. Takishima, M. Wada, and H. Murakami, “Reversible variable length codes,” *IEEE Trans. Comm,* vol. 43, pp. 158-162, 1995. DOI:10.1109/26.380026