論 文

효율적인 한글 코드화에 관한 연구

正會員 金 慶 泰* 正會員 閔 勇 植**

A Study on an Efficient Coding of Hangeul

Kyung Tae KIM*, Yong Sik MIN* *Regular Members*

要 約 본 논문은 한글의 특성을 3상태 변환 그래프를 이용하여 경제적으로 코드화하는 방법을 제시한 것이다. 이 방법을 사용하면, 한글을 표시하는데 코드의 평균 길이는 자모당 약 3.5비트 정도가 필요하게 되는데 이것은 다른 방법에서 제시된 것 보다 약 1비트이상 축약된 것이다. 따라서 정보교환의 필수 조건인 정확성 간결성 신속성 경제성을 향상시킬 수가 있게 된다.

ABSTRACT In this paper, we proposed an economical coding method to be applied for Hangeul character, the Korean letter, by utilizing three state transition graph. With this method, only about 3.5 bits are required in expressing a Hangeul character, which is more than 1 bit shorter than conventional codes so far introduced in order to realize extensive code compression. Hence this method will improve the rapidity and exactitude and economy in processing Hangeul letter.

I. 서 론

현재의 컴퓨터는 데이터 처리에 소요되는 시간보다는 정보의 전달이나 기억을 위하여 많은 시간이 소모되므로 컴퓨터의 효과적인 이용방법은 한정된 기억장소에 더 많은 양의 데이터를 기억시키거나 대량 정보 전송시간을 단축시키는 것이다. 이것을 위해서는 정보 전송시 화일로부터 불필요한 정보나 여분(redundancy)을 제거하거나 정보를 표현하는데 가장 효율적인 코드를 사용하는 방법을 연구할 필요가 있다. 즉 정보 전달시 압축된 최소한의 데이타를 전송하거나 또는 기억시키는 것이다.

특히 한글의 경우에 있어서는 그 특수한 기능

*光云大學校 電子計算學科
Dept. of Computer Science, Kwangwoon University.
**湖西大學校 電子計算學科
Hoseo Univ. Dept. of Computer Science.
論文番號 : 89-61(接受1989. 7. 27)

을 이용하면 재래의 정보처리 장치보다 더 능률적으로 정보를 처리할 수 있는 장치를 구성할 수 있을 수가 있다. 그간 많은 사람들이 한글의 구조 자체및 기계화에 대한 연구를 하였고[6], 현재는 한글 표준화 작업에까지 이르고 있다. 그러나 한글을 표시하기위한 비트수를 더욱 줄일 수 있는 방안을 개발할 필요가 있고, 데이터를 전송하는데 있어서도 더욱 개선할 필요를 느끼고 있다.

본 연구에서는 현재 사용되고 있는 데이타 압축 방법의 단점을 개선, 보안하여 데이타 통신 회선상에서 더욱 대량의 정보를 신속히 전송할 수 있는 한글 코드를 제시하고저 한다. 즉, 동적 huffman 코드를 이용하여 한글의 특성을 분석하여 만든 3상태 변환 그래프를 적용시킴과 동시에 한글을 수학적으로 분석하여 코드의 평균비트수를 최소로 줄일 수 있는 경제적인 한글 코드를 제시하고저 한다. 여기서 제안된 효율적인 한글 코드를 이용한다면 대량의 정보를 처리 함에 있어서 더 경제적이고도 효과적으로 기억시키거나 전송시킬 수 있으리라고 기대된다.

Ⅱ. 한글의 특성분석[1,2,5,6,7,8]

보내야할 데이타 비트수를 감소시킴으로써 데이타의 전송속도를 개선시키는데 이용되는 것이 데이타 압축인데 대체로 다음과 같은 것이 있다.

첫째, 발생되는 source symbol에 고정 길이를 부여시켜서 전송하거나 발생시키는 EBCDIC나 ASCII방법이 있다. 둘째로 source symbol 가운데서 공백이 많이 발생하는 경우 공백을 0, 아닌 경우를 1로하여 대응시켜 bit-mapping이라는 1바이트를 실행시켜 전송하는 bit-mapping방법이 있다. 그리고 bit-mapping 방법에서 발생되는 빈도 수에 따라서 코드길이를 부여시키는 방법인 이진 compact코드방식이 있다.

현재 컴퓨터 시스템에서는 첫번째 방법인 EBCDIC 나 ASCII와 같은 고정 길이의 코드가 많이 사용하지만, 이 방법 보다는 가변길이로 하는 것이 보다 경제적인 경우가 많다.

이때 source symbol을 가변 길이의 코드로 표현한 경우에는 각 symbol의 코드 길이가 일정치 않는데 발생하는 코드의 평균길이는 다음 식으로 표시된다.[2]

$$L=\sum_{i=1}^{q} P(Si)\ I(Si)$$

(여기서, P(Si)는 symbol Si의 확률
I(Si)는 symbol Si를 표시하는 코드 길이)

이 식에서 계산한 것과 같이, 발생 빈도가 높은 문자에 길이가 짧은 코드를 할당하고, 낮은 것에 긴 코드를 할당하는 가변 길이 코드를 채택하면 코드의 평균길이가 축소될 것이며 효율적인 코드화가 가능해진다.

이러한 목적을 위해서 한글의 사용 빈도에 관한 통계 자료를 수집하여 코드의 평균 길이가 다른 compact 코드로 만든다. 한글의 경우 24개의 기본문자가 사용되므로 최소한의 경우인 24가지의 상태를 고려해야 하는데 이 경우 코드의 실용성이 없어진다. 따라서 초성 쌍자음, 중모음, 종성 중자음을 하나의 symbol로 취급하는 3상태의 조건부 확률(conditional probability)를 고려하면 그림 1과 같은 형태의 그래프를 만들 수 있는데, 이것을 이용하면 보다 경제적인 코드화를 달성시킬 수가 있다.[6]

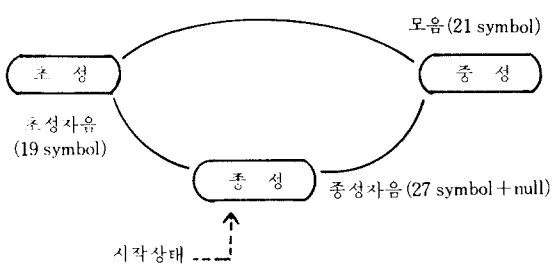


그림 1. 3 상태 변환 그래프
Three state transition graph

Ⅲ. 동적 compact 코드 알고리즘

각 문자에 대한 빈도수를 조사하여 정보전송시

확률이 높은 문자에 코드 길이를 짧게 그리고 확률이 낮은 문자에 코드길이를 길게 부여한 것이 이진 compact 코드방식이다.

각 symbol이 발생되는 사용빈도에 따라서 이진 compact 코드로서 코드 트리를 형성 시킬 수가 있다. 그러면 이때 leaf에 해당되는 각 symbol 의 사용 빈도합인 weight 값과 internal 노드에 해당되는 symbol들의 weight의 합을 비교한 경우에 있어서 같은 값을 지닌 경우가 발생이된다. 이때 이러한 internal 노드의 weight와 leaf에 해당되는 symbol의 weight값이 같은 경우에 그 위치를 교환시켜서 형성시키는 방법이 동적 compact코드 방법[3]이라 한다. 즉, 각 symbol 들의 발생빈도에 따라서 문장이 달라지는 경우에 그것에 따라 코드 길이를 부여시키고자 하는 방식임을 알 수가 있다.

예로서, source symbol S={ABABCDCADB-BADA)}인 경우의 동적 compact 코드를 부여시키기 위하여 weight 값이 발생된 트리를 살펴보면 그림 2와 같다. 이때 이 트리의 전체 weight 값 즉, 발생된 source symbol의 수 X level 수는 5X1+4X2+2X3+3X3=28이 된다. 그리고 weight값이 같은 leaf 노드와 internal 노드에 해당되는 위치를 교환시켜보면 다음 그림3과 같다. 이때의 전체 weight 값은 2X2+3X2+4X2+5X2=28이 되므로 그림 2의 전체 weight 값과 같음을 알 수가 있다. 그러면 이때 어느 트리로서 코드길이를 부여시키는가 하는 경우는 상황에 따라서 장·단점이 존재하나 평균적으로 그림 3의 형태로 표현한 것이 더 좋다.[3]

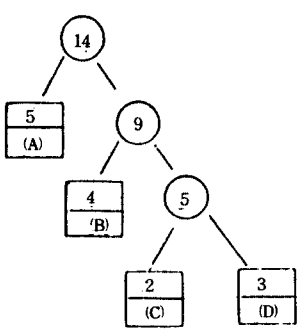


그림 2. Huffman방식 Huffmar method

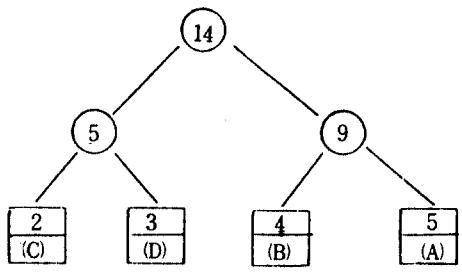


그림 3. 동적 Huffman방식 Dynamic Huffman method

따라서, 한글의 초성, 중성, 종성의 3가지 상태별로 각각의 source symbol들의 사용 빈도를 구하여 그 빈도에 의거하여 이진 코드 트리를 형성시킨 후에 위의 동적 compact 코드로서 weight값이 서로 같은 위치의 내용을 교환시켜서 위의 그림 3의 형태로 하면 최소의 비트수를 지닌 코드를 가짐을 알 수가 있다.

즉, 임의의 symbol Si에 대한 코드의 길이[2] 는

$$I(Si)=\log_2 \frac{1}{P(Si)} \quad (\text{bits})$$

가 되도록 코드길이를 부여하면 코드의 평균길이[2] 는

$$H(S)=\sum_{i=1}^{q} P(Si)\ I(Si)=\sum_{i=1}^{q} \log_2 \frac{1}{P(Si)} \quad (\text{bits})$$

와 같이 되어 평균길이가 가장짧은 코드가 될 것이다. 그러나 I(Si)는 반드시 정수이어야 하므로 해서 임의 symbol Si에 대한 평균코드 길이[2] 는

$$H(S) \leq I(Si) < H(S)+1$$

이 된다.

이와같은 특성을 지닌 동적 compact 코드의 알고리즘은 다음과 같은 순서로서 해결된다.

1. 각 source symbol들의 사용빈도를 고려하여

다음과 같은 과정을 통해 이진코드 트리를 형성한다.

a. 사용빈도의 크기순으로 symbol들을 다시 순서적으로 배열한다.

b. 사용빈도가 가장 낮은 symbol과 다음으로 낮은 symbol에 대해서 각각 1과 0의 코드 알파벳을 부여한다.

c. 위의 2개 symbol을 합성하여 하나의 새로운 symbol로 간주한다.

d. source symbol의 수효가 점차로 감소하여 1개로 될때까지 위의 방법을 반복한다.

2. 이같이 생성된 이진 코드 트리에서 leaf 노드와 internal 노드가 같은 weight값을 가지고 있는 노드이 부분트리를 교환시킨다. 이와같은 과정을 더이상 교환이 없을때까지 반복 수행한다.

3. 교환이 된 이진 코드 트리에 실제 코드 알파벳을 부여시켜서 각 symbol의 실제 코드를 작성해 나가면 우리가 요구하는 각 symbol의 코드가 된다.

Ⅳ. 실험 결과

그림 1의 3상태 변환 그래프를 적절히 활용하기 위해서는 먼저 각 상태의 symbol 들의 출현 확률을 파악할 필요가 있다. 이를 위해 본 논문에서는 symbol조사는 약 1150자 정도의 문자를 지닌 문장을 임의로 선정하여 실험에 이용하였다. 다른 통계자료를 이용한 경우에도 비슷한 효과를 나타내리라고 생각이 든다. 이것은 m차 Markov source가 가질 수 있는 장점이기 때문이다.[6]

알고리즘 순서에 입각하여서 각 문자 즉 초성,중성,종성에 대한 사용 빈도에 따라서 이진 코드 트리를 형성시킨 결과가 그림 4,5,6에 제시되어져 있다. 그리고 같은 weight값을 지닌 leaf 노드와 internal 노드를 교환 시켜서 새로이 형성된 동적 compact 코드 트리로 형성된 것이 그림 7,8,9에 나타나 있다. 그림 7,8,9에 표시된 이진 코드 트리는 본 연구에서 보이는 한개의 예에 불과하며 수학적인 특징이 같은 서로 다른 형태의 compact 코드도 존재할 수 있다.[6] 표 1,2,3은 이 작업의 결과를 종합한 것이다.

표 1,2,3에서 Σ 확률(probability) X길이(length) 의 전체 합은 그림 7,8,9의 각 상태에서 한개의 symbol을 표시하는데 필요한 평균 코드 길이 L을 의미한다. 또 한글의 한 음절을 표시하기 위한 평균 코드 길이는 초성, 중성, 종성에 대한 Σ 확률 X길이를 가산하면 3.48238 + 3.398339 + 2.40714= 9.28791 비트가 된다.

여기서 한글의 경제적 부호의 가장 좋은 효과를 가진다고 제시한 이진compact코드[6]로서 계산해 보면 3.6286 + 4.33931 + 2.40714 =10.37505 비트가 필요케 한다. 즉, 본 논문에서 제시된 이진 compact 코드 방식은 적어도 한글 한자당 1.09 비트가 축약됨을 알수가 있다.

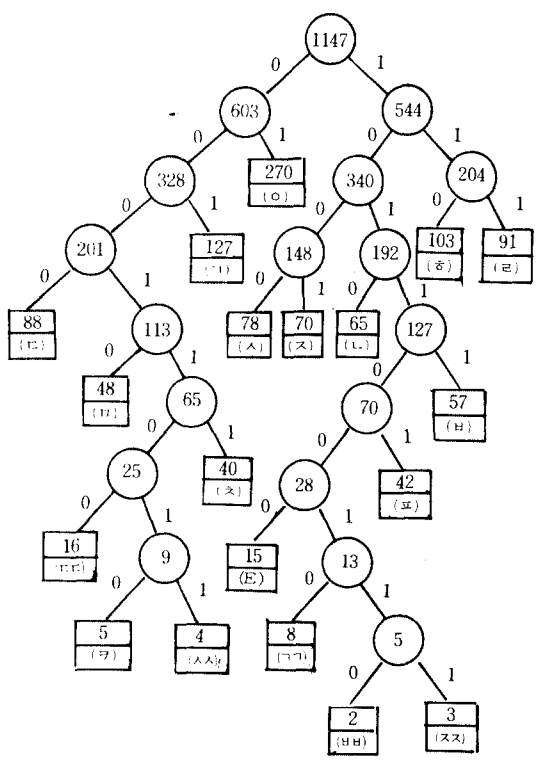


그림 4. 초성에 대한 이진 코드 트리
Binary code tree for Chosong

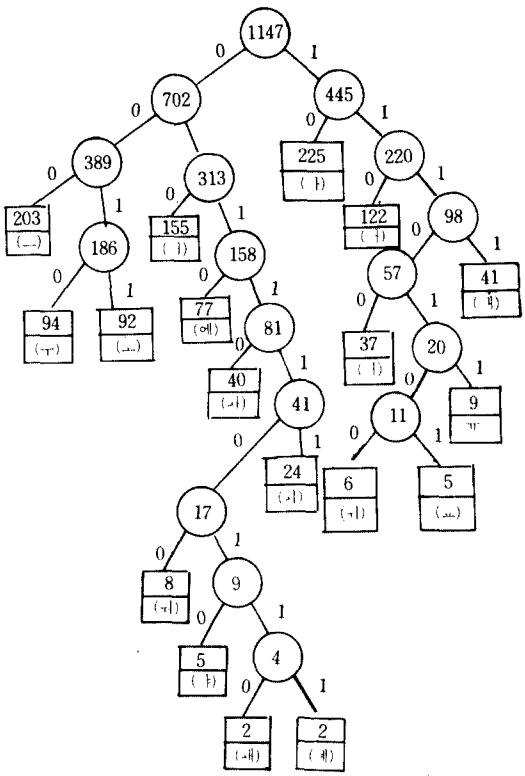


그림 5. 중성에 대한 이진 코드 트리
Binary code tree for Jungsong

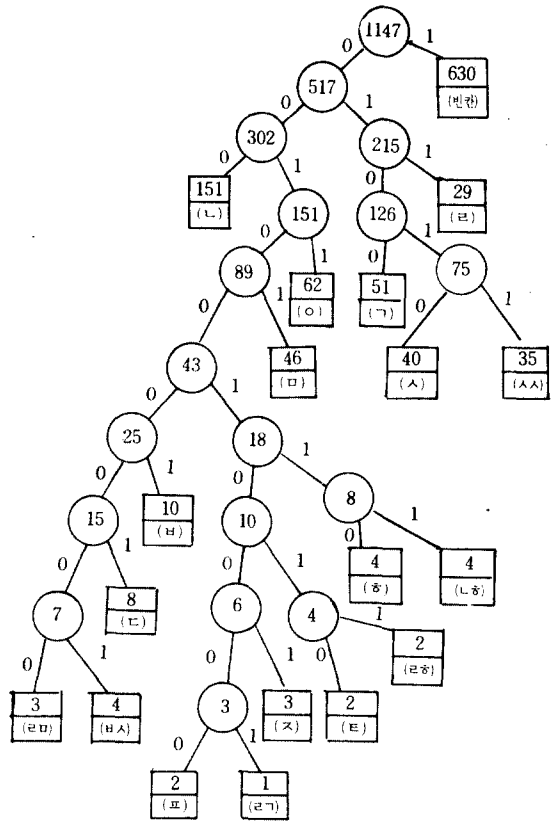


그림 6. 종성에 대한 이진 코드 트리
Binary code tree for Jongsong

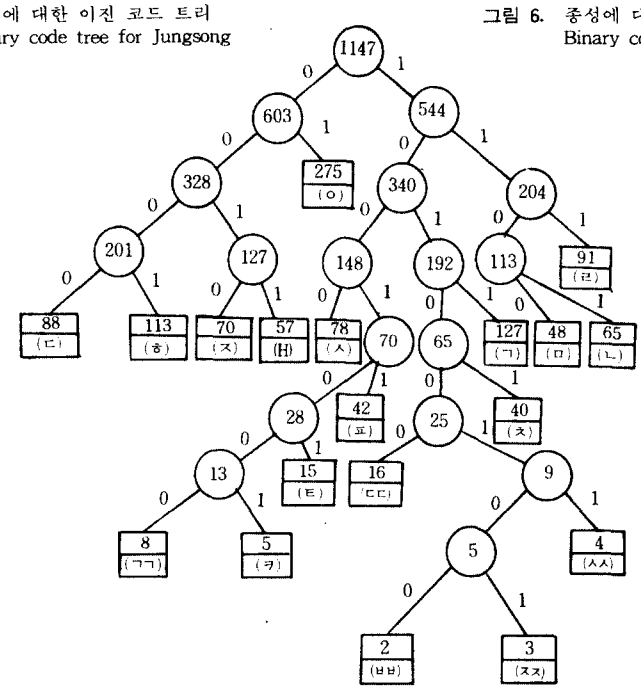


그림 7. 초성에 대한 동적 Huffman 코드 트리
Dynamic Huffman code tree for Chosong

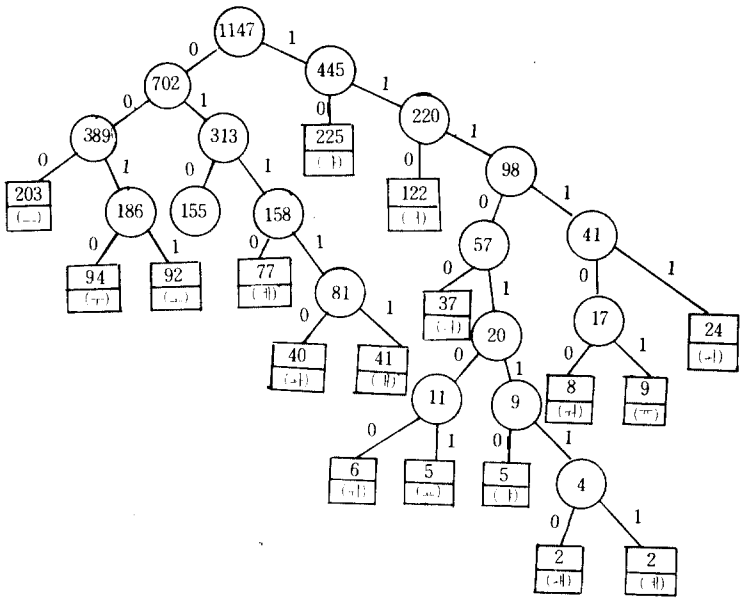


그림 8. 중성에 대한 동적 Huffman 코드 트리
Dynamic Huffman code tree for Jungsong

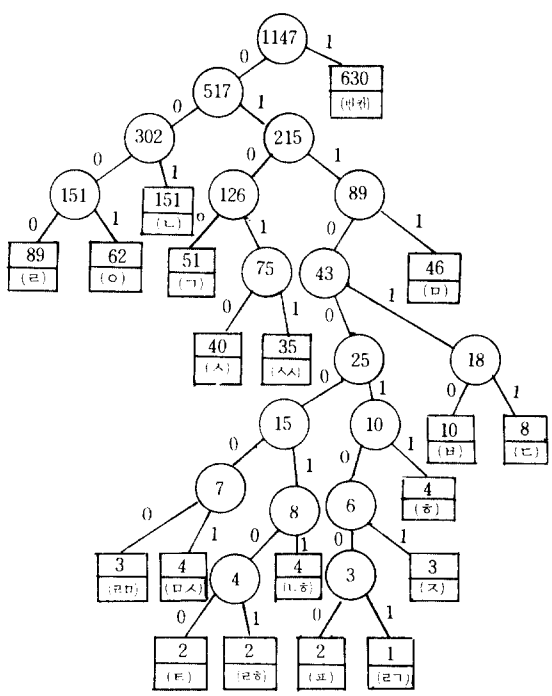


그림 9. 중성에 대한 동적 Huffman 코드 트리
Dynamic Huffman code tree for Jungsong

표 1. 초성을 위한 표
Table for Chosong

symbol	freguency	probability	code	length	prob.XL
ㄱ	127	0.11072	111	3	0.33216
ㄲ	8	0.00679	1001000	7	0.04879
ㄴ	65	0.05667	1101	4	0.22668
ㄷ	88	0.07672	0000	4	0.30688
ㄸ	16	0.01359	101000	6	0.0837
ㄹ	91	0.07933	111	3	0.23799
ㅁ	48	0.04185	1100	4	0.1674
ㅂ	57	0.04969	0011	4	0.19876
ㅃ	2	0.00174	10100100	8	0.01392
ㅅ	78	0.06801	1000	4	0.27204
ㅆ	4	0.00349	1010011	7	0.0238
ㅇ	275	0.23976	01	2	0.47952
ㅈ	70	0.06103	0010	4	0.24412
ㅉ	3	0.00262	10100101	8	0.02096
ㅊ	40	0.03487	1101	4	0.13948
ㅋ	5	0.00436	1001001	7	0.03052
ㅌ	15	0.01308	100101	6	0.07848
ㅍ	42	0.03662	10011	5	0.1831
ㅎ	113	0.09852	0001	4	0.39408
합 계	1147	1.00000			3.48238

표 2. 중성을 위한 표
Table for Jungsong

symbol	frequency	probability	code	length	prob.XL
ㅏ	225	0.19616	01	2	0.39232
ㅐ	41	0.03575	01111	5	0.17875
ㅑ	5	0.00436	1110110	7	0.03052
ㅓ	122	0.10637	110	3	0.31911
ㅔ	77	0.06713	0110	4	0.26852
ㅖ	2	0.00174	11101111	8	0.01392
ㅗ	92	0.08021	0011	4	0.32084
ㅘ	24	0.02092	11111	5	0.1046
ㅙ	40	0.03487	01110	5	0.17435
ㅚ	2	0.00174	11101110	8	0.01392
ㅛ	5	0.00436	1110101	7	0.03052
ㅜ	94	0.08195	0010	4	0.3278
ㅝ	6	0.00523	1110100	7	0.03661
ㅞ	9	0.00785	111101	6	0.0471
ㅟ	37	0.03226	11100	5	0.1613
ㅠ	155	0.13514	010	3	0.40542
ㅡ	203	0.17699	000	3	0.53097
ㅢ	8	0.00697	111100	6	0.04182
합 계	1147	1.00000			3.39839

표 3. 종성을 위한 표
Table for Jongsong

symbol	frequency	probability	code	length	prob.XL
ㄱ	51	0.44446	0100	4	0.17784
ㄴ	151	0.13156	001	3	0.39495
ㄷ	8	0.00679	011011	6	0.04182
ㄹ	89	0.07759	0000	4	0.31036
ㄺ	1	0.00087	011001001	9	0.00783
ㄻ	3	0.00262	01100000	8	0.02096
ㄼ	2	0.00174	011000101	9	0.01556
ㅁ	46	0.0401	0111	4	0.1604
ㅂ	10	0.00872	011010	6	0.05232
ㅃ	4	0.00349	01100001	8	0.02792
ㅅ	40	0.03487	01010	5	0.17435
ㅆ	35	0.03052	01011	5	0.1526
ㅇ	62	0.05406	0001	4	0.21624
ㅈ	3	0.00262	01100101	8	0.02096
ㅌ	2	0.00174	011000100	9	0.01566
ㅍ	2	0.00174	011001000	9	0.01566
ㅎ	4	0.00349	0110011	7	0.02443
(빈 칸)	630	0.54926	1	1	0.54926
나중	4	0.00349	01100011	8	0.02792
합 계	1147	1.00000			2.40714

V. 결 론

초성, 중성, 종성에 따라 각각 다른 3개의 코드표를 사용함으로서 한글 한 음절당 평균 9.28 비트로서 표시가 가능한 방법을 개발하였다. 한글 한 음절을 구성하는 평균 자모수가 2.64라는 통계값을 고려한다면 본 연구에서 제안한 코드의 길이는 자모당 약 3.5 비트정도가 된다. 이것은 지금껏 한글표시에 가장 좋다고 제시된 이진 compact 코드보다 약 1.09비트가 축약된 것이다.

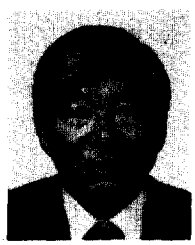
그러나, 이와 같은 경제적인 코드에도 불구하고 노드 수 만큼의 검색이 필요함과 동시에 서로 같은 weight값을 지닌 것들끼리 교환을 위해 처리되는 시간이 필요로 하게 되는 단점이 있다.

이같은 단점에도 불구하고 전송로상에 모든 정보를 송/수신하는데 있어서는 적은 용량을 가지고도 많은 양의 정보를 처리할 수가 있어서 효율이 다른 방식보다 우수함을 알수가 있다.

參 考 文 獻

1. Cobin, Harlod, "An introduction to data compression", Byte publication, pp.218-250, April, 1981.
2. Abramson, Norman, "Information theory and coding", McGraw-Hill, 1963.
3. Donald. E.Knuth, "Dynamic Huffman coding", Journal of algorithm, vol. 6. No. 2, pp. 163-180, June, 1985.
4. R.Tropper, "Binary-coded text, a text compression method", Byte publication, pp. 398-413, April, 1982.
5. Reingold, Nevergelt, Deo, "Combinational Algorithm: theory and practice", Prentice--Hall, 1977.
6. 김경태, 이균하, "한글 정보의 경제적 처리를 위한 부호화에 관한 연구", 한국정보과학회지, vol. 5, No.2, pp.99-104, Dec. 1978.
7. 김경태, 민용식, "OCM 방법을 이용한 Compact code 구성에 관한 연구", 한국통신학회

지, vol. 9, No. 3,pp.103-107, Oct. 1984.
8. 문경헤, "효율적인 자료 전송을 위한 compression 방식에 관한 연구", 광운대학교 대학원 석사학위 청구 논문, 1984.



金 慶 泰(Kyung Tae KIM) 正會員
1929年 5月15日生
1956年 2月：延世大學校 數學科 卒業
1958年 2月：延世大學校 大學院 數學科 卒業(理學碩士)
1972年：캐나다Dalhousie 大學校 大學院 應用數學科 卒業
1974年～現在：光云大學校 電子計算 學科 教授
1982年～1984年：韓國情報科學會副會長
1983年～1984年：캐나다 Carleton 大學 客員教授
1989年～現在：光云大學校 理科大學長

閔 勇 植(Yong Sik MIN) 終身會員
1958年 1月19日生
1981年 2月：光云大學校電子計算學科 卒業
1983年 2月：光云大學校 大學院 電子計算學科 卒業(理學碩士)
1984年～1987年：松源實業專門大學電子計算學科專任講師
1986年～現在：光云大學大學院電子計算學科 博士課程 修了
1987年～現在：湖西大學校 電子計算學科 助教授