허프만부호화 방식에의한 한글데이터의 압축에 관한 비교연구

○
남 상기 정 진 욱

성 균 관 대 학 교 정 보 공 학 과

A Study on the Compression Methods of Hangul Data File by the Huffman Encoding

Sang-Kee Nam , Jin-Wook Chung
Sung Kyun Kwan University, Department of Information Engneering

요약

데이터의 압축은 화일의 저장공간과 전송시간을 줄이는 중요한 이점을 제공한다. 국내에는 많은 경우 데이터 화일에 2 바이트로 구성된 표준한글부호를 포함하고 있다. 본 논문에서는 2 바이트로 부호화 된 한글을 포함한 데이터 화일을 허프만 부호화 방식에의해 압축 할때 한글을 한 바이트 단위로 인식하여 압축하는 경우와 두 바이트 단위로 인식하여 압축하는 경우의 여러가지 압축 특성을 비교 하였다. 아울러 사전에 조사된 한글의 잦기순서에 따라 고정된 압축 부호를 사용하는 경우와 앞에서 제시된 방법들을 비교하였다. 비교 결과 두 바이트 단위로 인식하여 압축하는 방법이 더 좋은 압축율을 보이었다.

I. 서론

데이터의 보관이나 전송시 의미의 손상없이 데이터를 압축하는 것은 큰 이점을 가져온다. 즉 데이터의 저장공간을 절약하므로서 공간이용의 효율화를 가져오며 동일한 대역 폭을 갖는 통신매체를 이용하여 전송을 행할 경우 전송소용시간을 단축 시켜 응답시간의 개선효과를 기대할수 있을 뿐만아니라 통신비용을 절감할수있다. 또한 데이터의 압축은 데이터의 저장이나 전송시 데이터의 비밀성을 높이는 데 기여할수도 있다. 이와같은 이득을 가져오는 데이터압축방법은 샤논의 정보이론 에서 출발하여 Run length encoding ,Diatomic encoding, Pattern encoding ,Bit map encoding,Half-byte encoding, Relative encoding ,Form mode operation,Huffman encoding [1)-13)] 등이 알려져 왔으나 이들 압축 방법 중에는 이론적으로 가장 높은 압축효율을 얻을 수있는 허프만의 방법이 널리 이용되고있다. 그러나 실용적으로 구현되어 이용되고 있는 허프만의 방식은 대부분 ASCII등의 영문 부호를 대상으로 하고 있어 압축할 데이터를 8 비트 단위로 인식하여 그 빈도 수를 조사한 다음 허프만 부호를 부여하여 압축하고 있다.

본 논문에서는 압축할 데이터화일의 대부분이 2 바이트완성형 한글 부호로 구성된 경우 다음과같이 몇가지로 허프만의 방법을 적용하여 구현하고 이들 방법들간의 압축 특성을 비교분석하였다.

첫째, 16 비트인식단위 : 고정빈도수 방식
사전에 한글의 잦기조사를 통하여 허프만 부호를 생성하고 고정된 이 테이블에 의거하여 압축복원을 행하는 방법으로 KS C 5601중 사용빈도 수가 높은 690자만을 압축 대상으로 하는 방식

둘째, 8 비트인식단위 : 가변 빈도수 방식
매번 압축대상화일에 포함된 기호들의 잦기조사를 행하여 허프만부호를 만들고 이 부호표에의거하여 압축과 복원을 행하는 방법으로 압축할 데이터를 8 비트단위로 인식하는 방법

셋째, 16 비트인식단위 : 가변빈도수 방식
둘째의 경우와 같으나 압축 할데이터를 한글인경우 16 비트 ,영문과 숫자인 경우 8 비트로 인식하는 방법

이들중 첫째 방법은 새로 구현 한 것이 아니고 참고문헌 [24)] 에서 사용한 방법을 그대로 적용하되 다른 방법과 비교를 위해 압축대상화일을 동일한 것을 사용하여 그 결과 만을 얻어 냈으며 둘째방법과 셋째방법은 새로이 구현하였다.

그리고 비교를 위해 둘째방법과 같으나 이미 상용화되어 이용되고 있는 압축프로그램을 같은 압축대상화일에 적용하여 역시 그 결과를 상호 비교하였다.

II. 압축 시스템(Compression System)의 설계

II.1 압축 시스템의 개요

압축 시스템은 그림 1 과같이 빈도조사모듈 ,허프만코드 생성모듈, 복원정보생성모듈,압축모듈(Compression module), 복원모듈(decompression module)등으로 구성되어 있다.

빈도조사모듈에서는 입력데이터에따라 코드빈도테이블을 가변적으로 생성하고 이 테이블을 이용하여 허프만코드생성모듈에서는 허프만코드를 생성하며 그 출력결과인 허프만 코드를 가지고 복원정보생성모듈에서 복원정보를 포함하고있는 유한자동테이블을 생성한다. 또 압축 모듈에서 이코드를 이용하여 데이터를 압축하는 역할을 수행한다. 압축된 형태의 코드 화일을 전송 할 때는 기존의 화일 전송 프로토콜을 이용하게 되며 복원모듈에서는 압축된 형태의 한글 화일을 원래의 상태로 복원하여 출력한다.

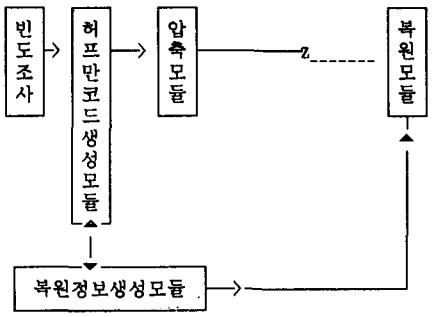


그림 1 압축시스템의 구성

II.2 압축 시스템

압축 시스템은 빈도조사모듈과 허프만코드생성모듈 그리고 복원정보생성 모듈,압축모듈로 구성되었으며 그 흐름은 아래 그림 2 와 같다.

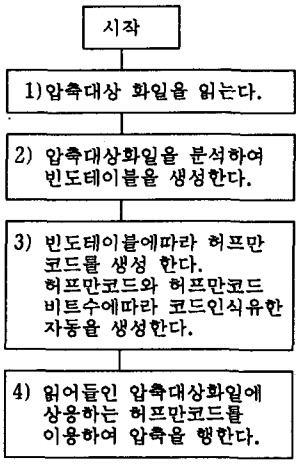


그림 2 압축시스템의 흐름도

II.2.1 빈도조사방법

가] 8 비트단위빈도조사

모든 문자들을 8 비트단위로 인식한다. 따라서 한글인 경우 한글자가 2 바이트로 나뉘어 별도의 기호로 인식된다.

나] 16 비트단위빈도조사

영문 숫자등 ASCII 코드들은 8 비트 단위로 빈도조사를 행하고 한글이나 그래픽문자들은 16 비트단위로 빈도 조사를 하는 방법으로 서론에서 언급한 세번째의 압축방법이 채택하고 있는 방법 이다.

II.2.2 허프만코드생성모듈

허프만코드생성모듈에서 사용되는 자료구조는 그림6 의 코드빈도테이블과 합산과정정보레코드들의 인접 리스트로 구성되고 코드빈도테이블의 구성이 합산과정 정보테이블과 같은 형식의 자료구조의 집합으로 구성되어 있다. 이 합산과정정보테이블은 정보필드와 링크필드로 구성되어 있어 코드테이블에 사용시는 이 정보 필드를 합산치저장소라고하고 합산과정의 진행상태를 나타내는 노드들의 연결리스트에 사용시에는 코드인덱스번호라한다.

생성과정은 다음과 같다.

1] 빈도조사모듈에서 그림 3의 코드빈도테이블을 생성한다.

2] 그림 6의 빈도 테이블에서 최소값을 갖는 두개의 항목을 찾아서 그 두 값을 합산(combine)하여 결과를 둘 중 작은 값이 들어있던 항목의 합산치 저장소필드에 저장한다.

3] 둘 중 다른 한 항목의 합산치저장소 필드는 영으로 한다.

4] 허프만 코드 생성시 그 전에 합산되어온 과정을 알기위해 합산과정정보레코드를 리스트로 유지해야하는데 이를 위하여 그림 6에서와 같은 링크드리스트(linked list)를 둔다. 여기에 위에서 구한 최소값들 중 큰 값의 인덱스를 코드인덱스번호 필드 에 저장하고 이 노드의 주소를 최소값중 작은 값을 갖는 인덱스의 코드빈도 테이블의 링크 필 드에 저장한다.

5] 합산에 참여한 두개의 심볼의 허프만코드는 적은쪽에 1 큰쪽에 0을 부여하고 그림 4의 허프만코드테이블에 보관한다.

6] 허프만 코드의 비트 수를 기억하기 위하여 그림 5과 같은 허프만코드비트수테이블을 두고 새로운 비트의 추가시 1씩 증가시킨다.

7] 위와 같은 과정을 반복 수행하는 데 두 번째 합산과정 이후에는 합산에 참여한 두 항목의 합산과정정보레코드링크필드의 값을 확인하여 합산과정리스트에 포함된 항목들에게 허프만코드의 1 비트를 추가 부여한다. 위와 같은 과정을 코드빈도테이블 합산치저장소 필드의 값이 모두 영이 될때까지 반복 하여 허프만코드테이블을 완성한다. 최종적으로 생성된 허프만 코드의 형태는 우측단(rightmost)에서부터 좌측으로 한비트 씩 이동되어 허프만 비트 수 만큼 저장되어 있다. 최종적인 결과인 허프만코드는 그림 4에 그리고 각 코드의 비트길이는 그림 5에 남게되고 나머지 비트는 영으로 채워져있다.

II.2.3 복호화정보생성모듈

여기서는 그림 8의 유한자동 테이블을 생성하며 그 내용은 아래와같다.

1] 허프만코드를 한 비트씩 이동시키면서 완전 이진트리의 자식노드 번호를 생성한다.

2] 1]에서 생성된 노드들이 터미날이 되면 그 노드 번호를 상태테이블에 기억 시킨다.

3] 모든 허프만코드에대한 터미날노드가 상태테이블에 저장되면 오름차순으로 정렬한다.

4] 상태테이블의 인덱스를 유한자동테이블의 첫단과 두번째단에 번갈아가며 저장한다.

5] 허프만코드의 터미날노드의 위치에 -1을 저장하고 그 뒤에따라는 유한자동테이블은 한줄씩아래로 밀어낸다. 이과정을 끝까지 반복한다.

II.2.4 압축모듈

1] 압축해야할 화일을 16 비트인식단위의 경우는 두 바이트단위로 가져와서 코드빈도테이블을 탐색하고 8 비트인식단위의 경우 는 한 바이트단위로가져 와 탐색한다.

2] 여기서 발견되면 그림 5에서의 허프만 코드의 비트수만큼 이동하면서 압축버퍼를 허프만 코드로 채운다.

3] 압축버퍼가 다 차게되면 16 비트인식단위는 1 바이트씩 두번 출력 하고 8 비트인식단위일 경우는 한번 출력한다.

이와 같은 과정을 반복하여 압축을 행한다.

(코드 , 빈도) (허프만코드) (비트수)

0	혜	1	0	00011	0	5
1	로	2	1	10011	1	5
2	와	4	2	1011	2	4
3	과	7	3	111	3	3
4	에	9	4	00	4	2
5	의	11	5	10	5	2
6	공백	13	6	01	6	2

그림 3 그림 4 그림 5

그림 3 코드빈도테이블, 그림 4 허프만코드테이블
그림 5 허프만코드비트수

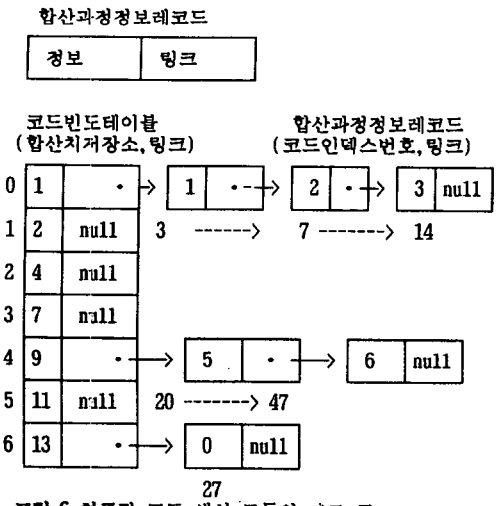


그림 6 허프만 코드 생성 모듈의 자료 구조

II.3 복원 모듈 (Decompression Module)

압축과정중 복호화과정은 복호화정보생성모듈에서 만들어진 그림 7의 허프만트리를 의미하는 그림 8의 유한자동테이블을 이용하여 복원을 수행 한다.

복원 알고리즘은 다음과 같은 과정을 거치게 된다.

1] 압축되어있는 데이터를 모두 끝까지 읽어들인다.

2] 읽어들인 데이타를 한 비트씩 이동하는 데 유한자동테이블(finite automata table)에서 다음상태의 결정를 현재의 상태와 한 비트 이동된 값의 1 ,0에의한다. 초기의 상태는 0 상태에서 시작되며 이동된 값이 0 이면 첫 단의값을 1 이면 두 번째단의 값을 다음 상태로 선택 한다.

3] 선택된 상태값이 -1 이되면 터미날로 인식하여 두 번째단의 인덱스 값을 취한다.

4] 코드테이블에서 인덱스 값에 해당하는 코드를 출력하고 상태를 초기화한다.

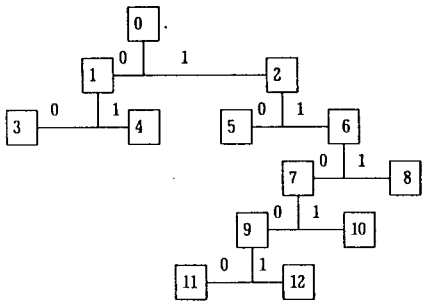


그림 7 허프만 트리

입력의 한 비트 이동된 값 (0,1)

상태	0	1
0	1	2
1	3 터	4 터
2	5 터	6
3	-1	인덱스 6
4	-1	인덱스 5
5	-1	인덱스 4
6	7	8 터
7	9	10터
8	- 1	인덱스 3
9	11 터	12 터
10	-1	인덱스 2
11	-1	인덱스 0
12	-1	인덱스 1

코드테이블

인덱스	
0	혜
1	로
2	와
3	과
4	에
5	의
6	공백

그림 8 유한자동테이블 과 코드 테이블

III. 구현 및 비교분석

III.1 구현

본 압축 시스템을 구현하기 위하여 사용한 컴퓨터는 IBM PC AT급으로 운영체제로서 Xenix를 사용하고 주기억장치의 용량은 1 MB 이며 보조기억장치의 용량 은 40MB 이다.[14)-16)] 압축 알고리즘을 구현하는 데는 XENIX

하에서 C 언어를 사용하였으며 각 모듈의 크기는 표 1-1~3과같으며[21),-23)] 생성된허프만코드표는 표 2-1 표 2-2, 표 2-3과같다.

표1. 압축시스템의 각 모듈의 크기

표1-1 고정빈도수방식 2 바이트인 경우

	모듈명칭	모듈크기(byte)
1	입력데이터생성모듈	17517
2	허프만코드생성모듈	9556
3	압축 모듈	9413
4	복원모듈	11165
5	출력데이터생성모듈	위의 1에포함됨
합계	30134 (1,5의 17517바이트는 생략)	

표 1-2 가변빈도수 방식 1 바이트인 경우

	모듈명칭	모듈크기(byte)
1	빈도조사모듈	35828
2	허프만코드생성모듈	77982
3	복원정보생성모듈	166275
4	압축모듈	32424
5	복원모듈	32050

표 1-3 가변빈도수방식 2 바이트인 경우

	모듈명칭	모듈크기(byte)
1	빈도조사모듈	25792
2	허프만코드생성모듈	67883
3	복원정보생성모듈	136292
4	압축모듈	22889
5	복원모듈	22061

고정빈도수방식에의한 허프만 코드의 일부를 표2-1에 보이었고 가변빈도수방식 8 비트 인식단위의 허프만 코드는 표 2-2에 ,가변빈도수방식 16 비트인식 단위는 표2-3에 보이 었다.

표2-1 한글의 허프만코드와 비트수

한글	허프만코드	비트수
의	0 1 0	3
에	1 0 1 1	4
한	0 0 0 0 1	5
・ ・ ・	・ ・ ・	・ ・ ・
팩	01111001011101	14
혜	01111001011100	14

표 2-2 가변빈도수방식 1 바이트인 경우

코드	허프만코드	비트수
0X20	01110	5
0XC0	01111	5
0X0A	10000	5
・ ・ ・	・ ・ ・	・ ・ ・
0X7C	00001101011100	14
0XFE	00001101011101	14

표 2-3 가변빈도수방식 2 바이트인 경우

코드	허프만코드	비트수
0X20	1 0	2
0X0A	11010	5
0XC0CC	000011	6
・ ・ ・	・ ・ ・	・ ・ ・
0X05	11100101001000	14
0X5C	11100101001001	14

III.2 비교분석

실험을 위해사용한 압축대상화일은 전체 357150 바이트이고 이중 한글과 순수영문을 제외한 일반적인 코드 80.95% 이고 순수영문이 19.05% 로 구성되어있다.

표 3과 그림 10-1 ~ 2 에서 알 수있듯이 16 비트 단위로 인식하는 경우가 8 비트단위로 인식하는 경우보다 코드의 집중도가 높은 것을 알 수있다. 이것은 결국 압축율에 있어서 16비트 인식의 경우 더 높은 것을 의미한다.

또, 가변빈도수방식 8 비트인식 방식의 짧은 허프만 코드의 크기는 5비트로 " 0xc0 "이고 가장긴 비트는 14 비트인데 반하여 가변빈도수방식 16 비트 인식 방식의 가장 짧은 허프만 코드 는 2 비트의 공백문자이고 가장긴 코드는 16진수로 0X5C인 14 비트코드이다. 샘플로 입력데이타의 크기가 35170 바이트 인 경우 허프만 코드의 평균 부호장과 엔트로피 및 코드효율 ,중복탐 (redundancy) 은 그림 9의 식 1 ~4]에의해서 구하여보면 그 결과가 표 4와 같다.

이와 같은 결과로 볼때 고정식 방법보다는 가변빈도수 방식이 그리고 8 비트 인식단위보다는 16 비트 인식단위가 더 압축율이 좋게 나타나며 평균부호장도 짧음을 알 수있다.

각 방법의 장단점을 살펴보면

• 고정빈도수방식 16 비트인식단위일경우

테이블이 고정되어있으므로 매번 복원에 필요한 유한자동테이블의 전송오버헤드가없어 압축데이터의 전송 시에 유리하나 이 허프만코드가 고정되어 있으므로 실제상황에 따른 허프만코드가

생성되지않아 압축효율이 가변빈도수방식 방식보다 떨어지며 압축된 형태의 코드와 그렇지않은 코드를 구별하는데 따른 오버헤드가 있다.

- 가변빈도수방식 8 비트인식단위인 경우

매입력 데이터에 따라 생성된 빈도 테이블에 의하여 허프만코드를 작성하므로 실제상황이 가장잘반영되지만 한글과 같은 2 바이트데이터도 1 바이트단위로 인식하므로 빈도들의 확률분산이 커져 부호장이 긴 비트들이 할당되어 2 바이트단위 때보다 효율이 좋지 못하다.

- 가변빈도수방식 16 비트 인식단위

가변빈도수방식 8 비트 에서와 같으나 한글과 같은 2 바이트단위를 하나의 심볼로 인식해야 하는 상황에서는 좋은 압축효율을 보인다.

그러나 복원시 필요한 유한자동테이블의 크기가 크며 압축 복원소요시간이 크다.

평균정보량(entropy) $H(s) = \sum_{i=1}^{q} p(s_i)\log\frac{1}{p(s_i)}$ 식 1]

s_i : 정보원 알파벳
$p(s_i)$: s_i 가 나타날 확률

평균부호장 $L = \sum_{i=1}^{q} p_i l_i$ 식 2]

p_i : i 번째부호가 나타날 확률
l_i : i 번째 부호의 부호장

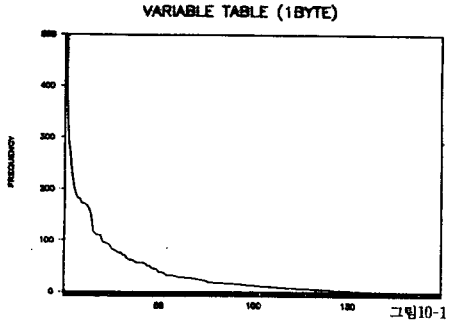
코드효율 $N = \frac{H(s)}{L}$ 식 3]

$redundancy = 1 - N = \frac{L - H(s)}{L}$ 식 4]

그림 9 압축관련 공식

표 3 입력데이터분석

코드	가변테이블 8	코드	가변 테이블 16
0xc0	0.029557	공백	0.488660
0xb0	0.029557	0x0a	0.050841
공백문자	0.029439	0xc0cc	0.041140
.	.	.	.
.	.	.	.
.	.	.	.
0x7c	0.000117	0x05	0.000127
0xfe	0.000117	0x5c	0.000127



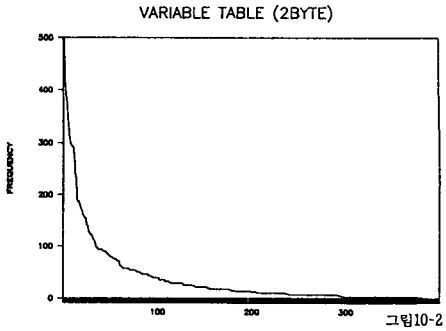


그림 10정렬된 빈도를 표현하는 코드값(입력 정보원 심볼을 빈도순서로 정렬하여 정보원심볼의 확률분포를 알수 있다.)

표 4 여러가지 압축방법의 비교테이블

	고정빈도수방식(2바이트)	가변빈도수방식		
		1 바이트 (기존)	1바이트	2 바이트
입력 35170 시 압축후 크기(byte)	225442	234462	165886	144725
압축율	1.56	1.50	2.12	2.38
엔트로피	7.007	5.30	3.27	5.36
평균부호장	8.79	-	3.28	6.38
코드효율	0.7971	-	0.9969	0.8401
redundancy	0.2028	-	0.0031	0.1599
수행시간	압축시 소요시간 (초)			
	8.0	4.1	12.01	33.13
	복원시소요시간 (초)			
	8.2	5.1	8.45	7.93
오버헤드 (사전정보)	없음	217 개 (195)	344 개 (173)	774 개 (388)

여기서압축시 소요시간은 빈도조사시간 ,허프만코드생성시간, 복원정보생성시간,압축시간등을 합산한 시간을 의미하며 사전정보에해당하는 오버헤드는 유한자동테이블크기를 나타내는데 위 표 4에서보면 195,172,388등은 서로다른 입력의 가짓 수를 의미하며 217,344,744는 상태테이블의 수를의미한다. 여기서 344를보면 가변빈도수방식 8 비트인식 방법이므로 344*8*2 비트가 복원정보 오버헤드가 된다.

IV. 결 론

허프만 방식에 근거한 가변빈도수 1 바이트 인식단위와 2 바이트 인식단위의 알고리즘을 설계하고 구현하였다. 그리고 구현된 두가지방식과 기존의 1 바이트인식단위의 알고리즘과 고정빈도수방식등 네가지 방식에대해 결과를 비교하였다. 네가지방식중 새로 구현된 가변빈도수 2 바이트인식방법이 가장좋은 압축율을 보이고 있으나 압축소요시간이 길고 복원에 필요한 정보량이 많아지므로 단점이 있다. 그러나 압축소요시간이 큰 관심의 대상이되지않는 뱃치환경에서는 이의 활용이 큰 문제가되지 않으며 전송의 경우에도 화일의 크기가 큰

경우 복원 정보량의 상대적 비율이 작아 지므로 활용이 가능하다. 추후 알고리즘의 개선으로 압축 소요시간을 단축 할 수있을 것으로 생각된다.

〈 참 고 문 헌 〉

1. Huffman, D.A.,"A Method for the Construction of Minimum Redundancy Codes" ,proc. IRE, vol. 40, no. 10, pp. 1098-1101, Sept. 1952.
2. Norman Abramson, "Information theory and Coding," McGRAW-Hill, 1963.
3. Synder, M. and Hunt, B,"The Myriad Virtues of Text Compression" Datamation, 1, pp. 36-40. Dec. 1970.
4. Ruth, s.s and Kreuzer,P.J.,"Data compression for large business files," Datamation, 18, 9, pp.62-66. Sept. 1972.
5. Rovert A.Wagner, "Common Phrases and Minimum-Space Text Storage," comm. CACM, vol. 16,No. 3, pp.148-151. Martch. 1973.
6. Bruce Hahn, "A Few Technique for Compression and Storage of Data," comm. Acm, Vol. 17, No. 8, pp. 434-436 Aug. 1974.
7. Ian J. Barton, and Susan E. Creasey, "An Information Theoretic Approach to Text Searching in Direct Access Systems," Comm. ACM, Vol. 17, No. 6, Jun 1974.
8. Frank Rubin, "Experiments in Text File Compression," Comm. ACM, Vol. 19, No. 11,pp. 617-623 Nov. 1976.
9. James L. Peterson, James R.Bitener, and John H.Howard, "The Selection of Optimal TOB Setting," Comm. ACM, Vol. 21, No. 12, Dec. 1978.
10. Timothy Mc Cullough, "Data Compression in High-Spreed Digitial Facsimile Tellcommunications," July 1977.
11. Usko Moilanen, "Information Processing Codes Compress Binary Picture Data," Computer Design, Nov. 1978.
12. Gilbert Held, "Data compression," John Willey & Sons, Inc., 1983.
13. V. Cappellin, "Data Compression and Error Contror Techniques with Applications," Academic Press, 1985.
14. The Santa Cruz Operation,Inc, "The XENIX V operating system user's manual I,II"
15. The Santa Cruz Operation,Inc, "The XENIX V development system programmer's guide I,II"
16. The Santa Cruz Operation,Inc, "The XENIX V development system programmer's reference"
17. "IBM Technical manual Telecommunications Fundamentals," INFOTEL SYSTEM CORP., 1985.
18. "과학기술처 , 한글 한자정기조사," 1986.
19. "정보 교환용 부호 (한글한자) KS C 5601," 1989.
20. rechard W.Hamming, "Coding and information Theory. second edition," New Jersey : Prentice-Hall, 1986.
21. Herbert Schildt, "C: The complete Reference," Osborne:McGRAW-Hill, 1987.
22. Joe Campbell, "C Programmer's Guide to Serial Communications," HAWARD W.SAMS & COMPANY, 1987.
23. James F. Korsh, and Leonard J. Garrett, "Data structure and algorithm using in C," Boston:PWS- KENT Publishing Company, 1988.
24. 정 진욱, "화일전송효율화를 위한 compression Algorithm연구 ", 성균관대학교 논문집 과학기술편 제 40 집 NO.1 1989.9.