+0?

데이터 압축하기

기초100제+0?

2016.09

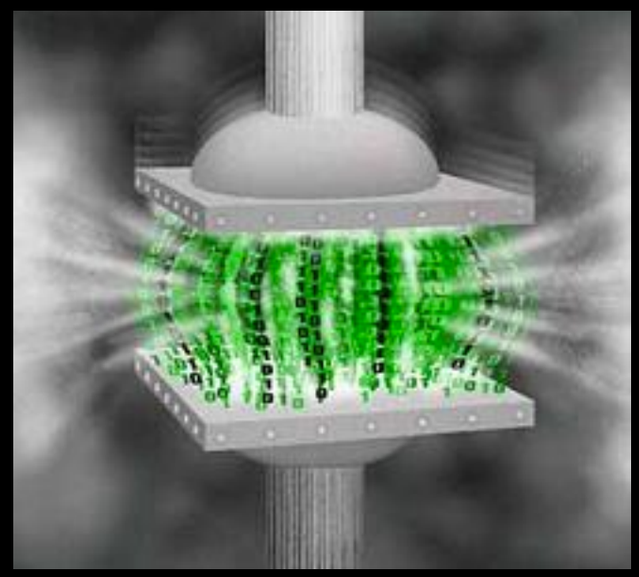
과학영재학교 경기과학고등학교

(v0.0) 전현석 melongist@gmail.com

- 데이터 압축?이란 무엇일까?

- 원래의 정보를 보다 적은 양의 데이터로 표현할 수 있다면?

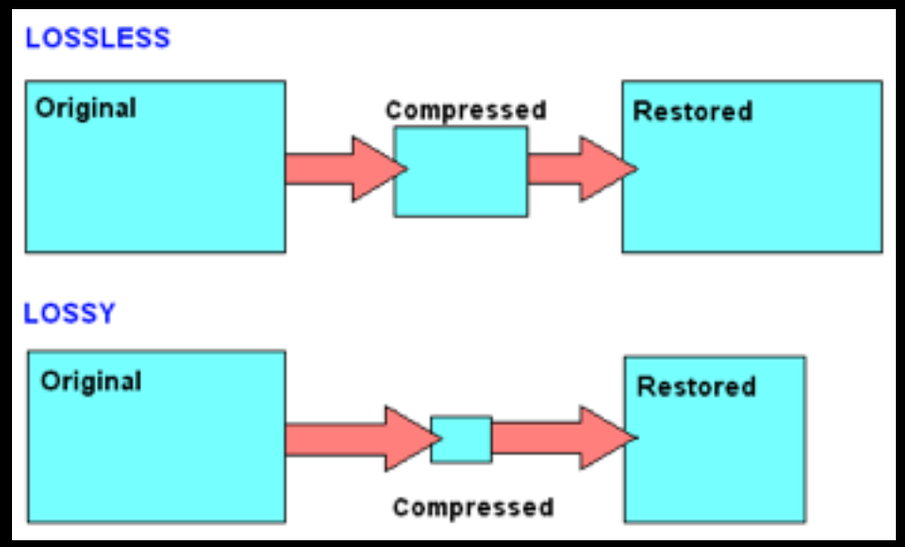
- 데이터를 저장하기 위한 저장공간과 데이터를 전송하기 위한 시간을 줄일 수 있다.



- 데이터 압축은 압축 과정에서의 데이터 손실 여부에 따라 2가지로 나눌 수 있다.

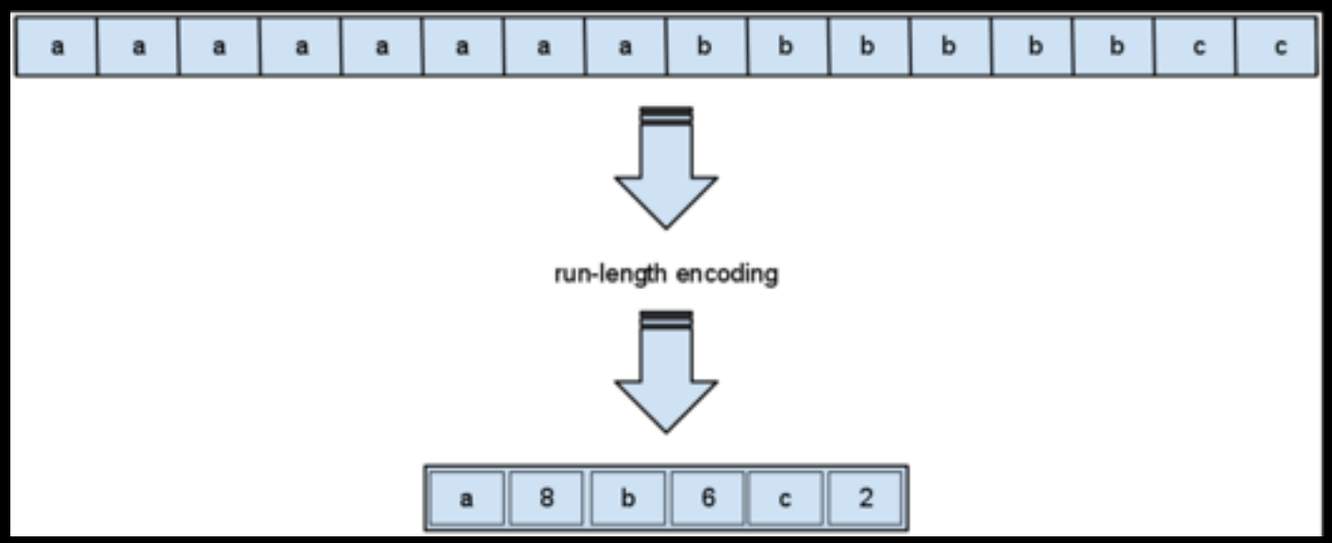
- 압축과정에서 데이터 손실이 발생되지 않는 무손실(lossless) 압축

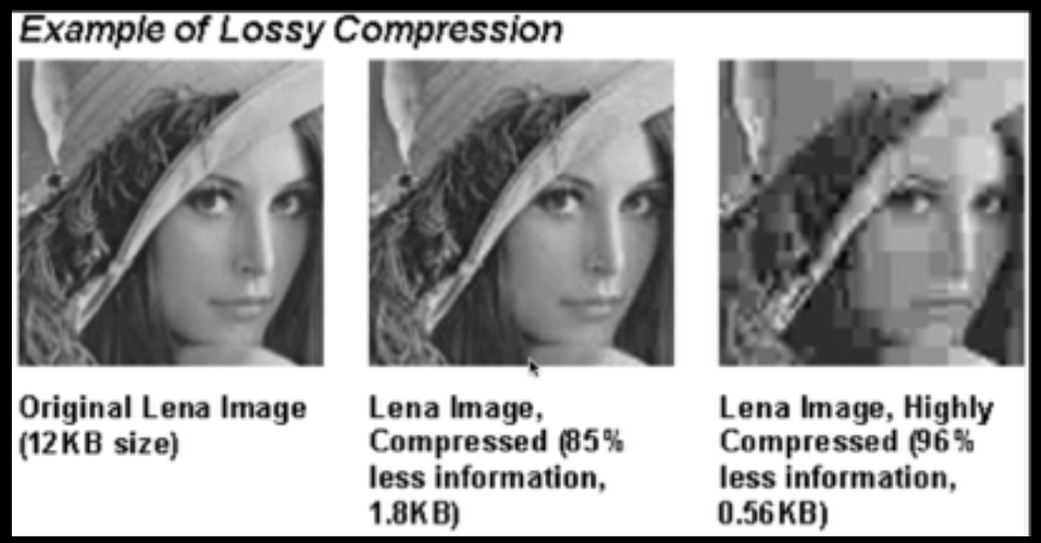
- 압축과정에서 데이터 손실이 발생되는 손실(lossy) 압축



- 무손실 압축은…
  데이터가 손실 되지 않아야 하는 데이터파일, 실행파일의 압축에 주로 사용되고,

- 손실 압축은…
  데이터가 손실이 크게 지장이 없는 미디어(사운드/이미지) 압축에 주로 사용된다.





Run Length Encoding

RLE(run length encoding)

- RLE 는 가장 간단한 압축 알고리즘의 하나로서,

  - 연속적으로 반복되는 횟수를 함께 기록하여 데이터의 양을 줄이는 비손실 압축이다.

- 데이터가 연속적으로 반복되는 경우가 많은 경우 효과적으로 사용될 수 있다.

- https://en.wikipedia.org/wiki/Run-length_encoding

RLE(run length encoding)

- 흰(W) 화면에 검정(B) 글자를 나타내기 위한 67개의 문자가 아래와 같다고 한다면?

WWWWWWWWWWWBWWWWWWWWWWWWBBBWWWWWWWWWWWWWWWWWWWWWBWWWWWWWWWWWWWW

- [연속된개수,문자] 의 쌍으로 반복되는 구간을 표현해서

12W1B12W3B24W1B14W

와 같이 18개의 문자로 재표현 할 수 있다.

- 연속적으로 반복되는 횟수를 이용하는 다양한 RLE 방법을 만들 수 있다.

RLE(run length encoding)

- 10000 개의 문자('W' or 'B') 로 구성된 랜덤 텍스트 파일 생성

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int n=10000;  // # of data
char str[32768];

int main()
{
  FILE *out=fopen("rletext.txt", "w");

  for(int i=1; i<=n; i++)
    str[i]='W';

  srand((int)time(NULL));
  for(int i=1; i<=n/10; i++)
  {
    int c=rand()%n;
    str[c]=(str[c]=='W'?'B':'W');
  }

  for(int i=1; i<=n; i++)
    fprintf(out, "%c", str[i]);
}
```

RLE(run length encoding)

- rletext.txt 파일 RLE 압축 인코딩

```c
#include <stdio.h>

int n, nc;
char str[32768];

int main()
{
  FILE *in=fopen("rletext.txt", "r");
  FILE *out=fopen("rleencoded.txt", "w");

  while(!feof(in))
    fscanf(in, "%c", &str[n++]);
  n--;

  for(int i=0; i<n; i++)
  {
    int cnt=1;
    while(str[i]==str[i+1] && i+1<n)
    {
      cnt++;
      i++;
    }
    fprintf(out, "%d%c", cnt, str[i]);
  }
}
```

RLE(run length encoding)

- rleencoded.txt 파일 RLE 압축 해제 디코딩

```c
#include <stdio.h>

int n, nc;
char str[32768];

int main()
{
  FILE *in=fopen("rleencoded.txt", "r");
  FILE *out=fopen("rledecoded.txt", "w");

  while(!feof(in))
    fscanf(in, "%c", &str[n++]);
  n--;

  for(int i=0; i<n; i++)
  {
    int cnt=str[i]-'0';
    while(str[i+1]>='0' && str[i+1]<='9' && i+1<n)
    {
      cnt *= 10;
      cnt += (str[i+1]-'0');
      i++;
    }

    for(int j=1; j<=cnt; j++)
      fprintf(out, "%c", str[i+1]);
    i++;  //skip character
  }
}
```

Lempel-Ziv-Welch

Lempel-Ziv-Welch

- LZW 는 매우 효율이 좋은 압축 알고리즘의 하나로서,

  - 단어 사전을 만들어 사용하는 비손실 압축이다.

- 데이터의 양이 많은 경우 효과적으로 사용될 수 있다.

- https://en.wikipedia.org/wiki/Lempel-Ziv-Welch

Lempel-Ziv-Welch

- 24개의 문자가 아래와 같다고 한다면?

TOBEORNOTTOBEORTOBEORNOT

- 다음과 같은 단어 사전을 만들고

```
 1:T   2:O   3:B   4: E   5:R   6:N    7:TO   8:OB   9: BE   10: EO
11:OR 12:RN 13:NO 14: OT 15:TT 16:TOB 17:BEO 18:ORT 19: TOBE 20: EOR
21 RNO
```

1 2 3 4 2 5 6 2 1 7 9 11 16 10 12 14

와 같이 16개의 값으로 재표현 할 수 있다.(원래 데이터는 24개의 값이 필요하다.)

- 한 개의 값으로 여러 문자를 한 번에 표현할 수 있는데…
  - 단어 사전을 함께 저장해야하는 단점이 있지만,
  - 데이터의 양이 많으면 많을수록 압축율이 좋아진다. ZIP 알고리즘의 원형?

Lempel-Ziv-Welch

- 다음과 같은 데이터에 대한 단어 사전을 만들어내는 알고리즘은?

ABABBABCABABBA

- 1. 압축할 데이터집합의 각 문자에 대한 기본 문자 사전을 만든다.
- 2. 사전을 이용해 코드를 변환한다.
- 3. 더 만들어질 수 있는 단어들을 사전에 추가해 간다.
- 4. 데이터의 마지막까지 읽어 코드와 사전을 완성한다.

Lempel-Ziv-Welch

- 기본 문자 사전 만들기

ABABBABCABABBA

압축할 문자열	사전	
	번호	내용
ABABBABCABABBA	1	A
	2	B
	3	C

Lempel-Ziv-Welch

- 압축 코드생성 및 단어 (사전에) 추가하기

ABABBABCABABBA

```
s = 입력되는 문자;
while(문자열의 마지막까지)
{
  c = 그 다음에 입력되는 문자;
  if("s+c" 가 사전에 있으면)
  {
    s = "s+c";
  }
  else
  {
    s 문자의 코드 출력;
    "s+c"를 새로운 코드로 사전에 추가;
    s = c;
  }
}
s 문자의 코드 출력;
```

Lempel-Ziv-Welch

- 압축 코드생성 및 단어 (사전에) 추가하기

ABABBABCABABBA

압축할 문자열	사전	
ABABBABCABABBA	번호	내용
	1	A
출력된 LZW 코드	2	B
1	3	C
	4	AB

처음에 A가 입력되고, 그 다음 B가 입력된다.

AB가 사전에 없으므로, A의 코드인 1을 출력하고, AB를 사전에 새롭게 추가한다.
A다음의 B로 이동해 같은 방법으로 진행한다.

압축할 문자열	사전	
ABABBABCABABBA	번호	내용
	1	A
출력된 LZW 코드	2	B
12	3	C
	4	AB
	5	BA

다음에 B가 입력되고, 그 다음 A가 입력된다.

BA가 사전에 없으므로, B의 코드인 2를 출력하고, BA를 사전에 새롭게 추가한다.
B다음의 A로 이동해 같은 방법으로 진행한다.

Lempel-Ziv-Welch

- 압축 코드생성 및 단어 (사전에) 추가하기

ABABBABCABABBA

압축할 문자열	사전	
ABABBABCABABBA	번호	내용
	1	A
출력된 LZW 코드	2	B
12	3	C
	4	AB
	5	BA

다음에 A가 입력되고, 그 다음 B가 입력된다.
AB가 사전에 있으므로, 다음 입력 단어를 AB로 바꾼다.

이전과 같은 방법으로 진행한다.

압축할 문자열	사전	
ABABBABCABABBA	번호	내용
	1	A
출력된 LZW 코드	2	B
124	3	C
	4	AB
	5	BA
	6	ABB

다음에 AB가 입력되고, 그 다음 B가 입력된다.
ABB가 사전에 없으므로, AB의 코드인 4를 출력하고, ABB를 사전에 새롭게 추가한다.

AB다음의 B로 이동해 같은 방법으로 진행한다.

Lempel-Ziv-Welch

- 압축 코드생성 및 단어 (사전에) 추가하기

ABABBABCABABBA

압축할 문자열	사전	
ABABBABCABABBA	번호	내용
	1	A
출력된 LZW 코드	2	B
124	3	C
	4	AB
	5	BA
	6	ABB

다음에 B가 입력되고, 그 다음 A가 입력된다.
BA가 사전에 있으므로, 다음 입력 단어를 BA로 바꾼다.

이전과 같은 방법으로 진행한다.

압축할 문자열	사전	
ABABBABCABABBA	번호	내용
	1	A
출력된 LZW 코드	2	B
1245	3	C
	4	AB
	5	BA
	6	ABB
	7	BAB

다음에 BA가 입력되고, 그 다음 B가 입력된다.
BAB가 사전에 없으므로, BA의 코드인 5를 출력하고, BAB를 사전에 새롭게 추가한다.

BA다음의 B로 이동해 같은 방법으로 진행한다.

Lempel-Ziv-Welch

- 압축 코드생성 및 단어 (사전에) 추가하기

ABABBABCABABBA

압축할 문자열	사전	
ABABBABCABABBA	번호	내용
	1	A
출력된 LZW 코드	2	B
12452	3	C
	4	AB
	5	BA
	6	ABB
	7	BAB
	8	BC

다음에 B가 입력되고, 그 다음 C가 입력된다.
BC가 사전에 없으므로, B의 코드인 2를 출력하고, BC를 사전에 새롭게 추가한다.

B다음의 C로 이동해 같은 방법으로 진행한다.

압축할 문자열	사전	
ABABBABCABABBA	번호	내용
	1	A
출력된 LZW 코드	2	B
124523	3	C
	4	AB
	5	BA
	6	ABB
	7	BAB
	8	BC
	9	CA

다음에 C가 입력되고, 그 다음 A가 입력된다.
CA가 사전에 없으므로, C의 코드인 3를 출력하고, CA를 사전에 새롭게 추가한다.

C다음의 A로 이동해 같은 방법으로 진행한다.

Lempel-Ziv-Welch

- 압축 코드생성 및 단어 (사전에) 추가하기

ABABBABCABABBA

압축할 문자열	사전	
ABABBABCABABBA	번호	내용
	1	A
출력된 LZW 코드	2	B
124523	3	C
	4	AB
	5	BA
	6	ABB
	7	BAB
	8	BC
	9	CA

다음에 A가 입력되고, 그 다음 B가 입력된다.
AB가 사전에 있으므로, 다음 입력 단어를 AB로 바꾼다.

이전과 같은 방법으로 진행한다.

압축할 문자열	사전	
ABABBABCABABBA	번호	내용
	1	A
출력된 LZW 코드	2	B
1245234	3	C
	4	AB
	5	BA
	6	ABB
	7	BAB
	8	BC
	9	CA
	10	ABA

다음에 AB가 입력되고, 그 다음 A가 입력된다.
ABA가 사전에 없으므로, AB의 코드인 4를 출력하고, ABA를 사전에 새롭게 추가한다.

AB다음의 A로 이동해 같은 방법으로 진행한다.

Lempel-Ziv-Welch

- 압축 코드생성 및 단어 (사전에) 추가하기

ABABBABCABABBA

압축할 문자열	사전	
ABABBABCABABBA	번호	내용
	1	A
출력된 LZW 코드	2	B
	3	C
	4	AB
	5	BA
1245234	6	ABB
	7	BAB
	8	BC
	9	CA
	10	ABA

다음에 A가 입력되고, 그 다음 B가 입력된다.
AB가 사전에 있으므로, 다음 입력 단어를 AB로 바꾼다.

이전과 같은 방법으로 진행한다.

압축할 문자열	사전	
ABABBABCABABBA	번호	내용
	1	A
출력된 LZW 코드	2	B
	3	C
	4	AB
	5	BA
1245234	6	ABB
	7	BAB
	8	BC
	9	CA
	10	ABA

다음에 AB가 입력되고, 그 다음 B가 입력된다.
ABB가 사전에 있으므로, 다음 입력 단어를 ABB로 바꾼다.

이전과 같은 방법으로 진행한다.

Lempel-Ziv-Welch

- 압축 코드생성 및 단어 (사전에) 추가하기

ABABBABCABABBA

압축할 문자열	사전	
ABABBABCABABBA	번호	내용
	1	A
출력된 LZW 코드	2	B
	3	C
	4	AB
	5	BA
	6	ABB
12452346	7	BAB
	8	BC
	9	CA
	10	ABA
	11	ABBA

다음에 ABB가 입력되고, 그 다음 A가 입력된다.
ABBA가 사전에 없으므로, ABB의 코드인 6을
출력하고, ABBA를 사전에 새롭게 추가한다.

ABB다음의 A로 이동해 같은 방법으로 진행한다.

압축할 문자열	사전	
ABABBABCABABBA	번호	내용
	1	A
출력된 LZW 코드	2	B
	3	C
	4	AB
	5	BA
	6	ABB
124523461	7	BAB
	8	BC
	9	CA
	10	ABA
	11	ABBA

다음에 A입력되고, 그 다음 입력이 없다.

A의 코드를 출력하고 압축이 종료된다.

Lempel-Ziv-Welch

- 압축 코드생성 및 단어 (사전에) 추가하기

ABABBABCABABBA

LZW 코드	LZW 코드 사전	
124523461	번호	내용
	1	A
복원한 데이터	2	B
ABABBABCABABBA	3	C
	4	AB
	5	BA
	6	ABB

함께 저장되어있는 사전을 이용해 LZW 코드를 풀어내면 원래의 데이터가 복원된다.

Lempel-Ziv-Welch

- 파일 입출력 준비
- 단어 사전을 만들어야 하는데… C++ 의 자료구조 map 을 이용하자!
- map 구조는 {“인덱스값”, “저장값”} 을 묶어 저장하고,
  “인덱스값”을 이용해 “저장값”을 알아낼 수 있다.
- 이왕 C++ 사용하는 김에 C++의 파일 입출력 방법도 사용해 보자. ^_^;

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <map>
using namespace std;

map<string, int> endic;  //for encoding
map<int, string> dedic;  //for decoding
int dcnt;                //dictionary count
int enc[10000], ecnt;    //encoding array

int main()
{
  ifstream in("lzwtext.txt");    //input file stream
  ofstream out("lzwencoded.txt");//output file stream
```

Lempel-Ziv-Welch

- 기본 문자 사전({문자, 값} 맵) 만들기
- 코드 복원을 위해 역방향({값, 문자} 맵) 도 같이 만든다.

- .find(“인덱스값”) 을 실행하면, “저장값”을 리턴 된다.
  “ 인덱스값 ” 으로 저장되어있는 “ 저장값 ” 이 없다면? .end()값이 리턴 된다.
- .insert({“인덱스값”, “저장값”}) 을 실행하면, 맵에 추가된다.

```
dcnt=1;
char t;
while(in.get(t))  //make dictionary for character
{
  string s="";
  s+=t;

  if(endic.find(s)!=endic.end())
    continue;

  endic.insert({s, dcnt});
  dedic.insert({dcnt, s});
  dcnt++;
}
in.close();
```

Lempel-Ziv-Welch

- 파일을 다시 읽어 들이면서…
- LZW 코드를 생성하고, 새로운 단어를 사전에 추가한다.

- 맵[“인덱스값”] 으로 “저장값”을 가져올 수 있다.

```
in.open("lzwtext.txt");
ecnt=1;
string s="";
while(in.get(t))  //make dictionary + encoding
{
  if(endic.find(s+t)!=endic.end())
    s+=t;
  else
  {
    enc[ecnt]=endic[s];
    ecnt++;

    endic.insert({s+t, dcnt});
    dedic.insert({dcnt, s+t});
    dcnt++;

    s=t;
  }
}
enc[ecnt]=endic[s];
ecnt++;
```

Lempel-Ziv-Welch

- 만들어진 사전 정보와….
- LZW 코드를 파일로 출력한다.
  (원래는 2진 코드로 바꿔 출력해야 하지만, 알고리즘의 이해를 위해 텍스트로 출력)

```
  dcnt--;              //dictionary out
  cout<<dcnt<<endl;
  out<<dcnt<<endl;
  for(int i=1; i<=dcnt; i++)
  {
    cout<<i<<" "<<dedic[i]<<endl;
    out<<i<<" "<<dedic[i]<<endl;
  }

  ecnt--;              //encoding out
  for(int i=1; i<=ecnt; i++)
  {
    cout<<enc[i]<<" ";
    out<<enc[i]<<" ";
  }
  cout<<endl;
  out<<endl;
}
```

Lempel-Ziv-Welch

- ABABBABCABABBA 데이터가 들어있는 lzwtext.txt 파일

Lempel-Ziv-Welch

- ABABBABCABABBA 데이터에 대한
- 생성된 LZW 사전과 압축코드
- 압축 알고리즘만 텍스트 모드로 출력했다!!! 압축 파일의 용량이 더 클 것이다?
- 실제 압축은 2진 모드로 출력해야 한다.

```
ABABBABCABABBA
```

```
11
1 A
2 B
3 C
4 AB
5 BA
6 ABB
7 BAB
8 BC
9 CA
10 ABA
11 ABBA
1 2 4 5 2 3 4 6 1
```

마치며...

- 데이터의 양을 줄이는 것을 데이터 압축이라고 한다.

- 압축 과정에서 데이터의 손실 여부에 따라, 손실/비손실 압축으로 나눌 수 있다.

- 가장 간단한 압축 방법으로는 연속적으로 반복되는 길이를 이용하는 RLE가 있다.

- 단어 사전을 만들어 압축하는 LZW 방법도 있다.

- 문자의 빈도에 따라 다른 비트길이를 사용하는 허프만 코딩 알고리즘도 있다.
  https://en.wikipedia.org/wiki/Huffman_coding