Compression
정상
좋은 - huffman
안좋은
비정상

Kraft inequality

$$\sum_{i=1} 2^{-l_i} \le 1$$

Introduction to Computer Science & Engineering

Lecture 3.5: Data Compression Theory

Jeonghun Park

KYUNGPOOK NATIONAL UNIVERSITY

General Compression Theory

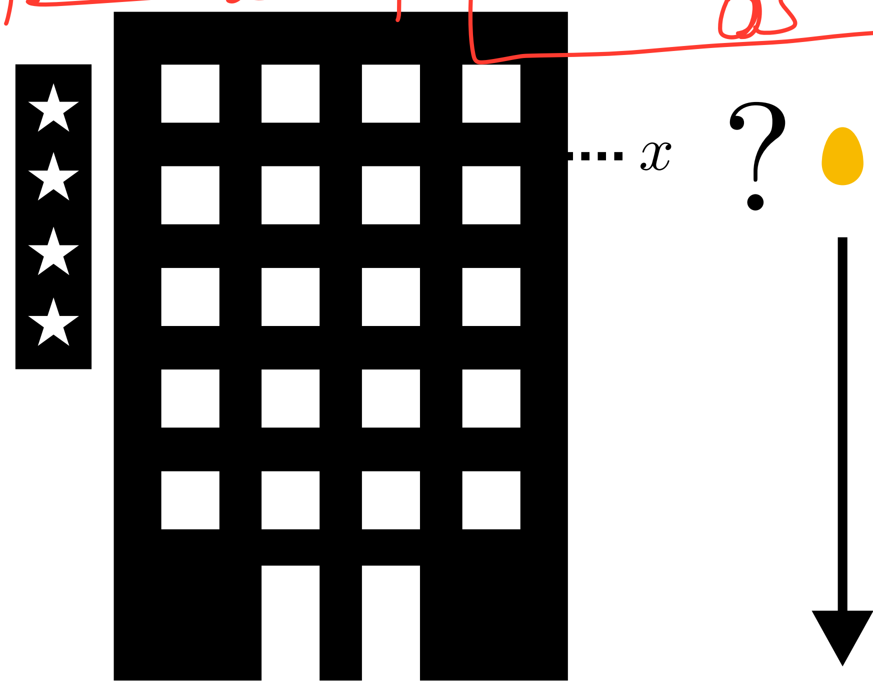
63일 경우

$63/2/2/2/2 \cdots /2 = 1$ 일 때 $\log_2 x$

구하면 빨리 나옴

Worst Case $\log_2 63$

linear 하게 찾으면 worst 63



A dropped egg can have two information: broken vs not broken

Let's consider:
Broken egg: 0
Unbroken egg: 1

So it is equivalent to:
$x \rightarrow \text{bits}$

A basic concept of compression

$\log_2 x$

Binary Search

jeonghun.park@knu.ac.kr

KYUNGPOOK NATIONAL UNIVERSITY

Data Compression

압축

- Definition
  - Represent a general data (let's call ***x***) by using a finite number of ***bits***

x → Compression function → 001001.. $\Longleftrightarrow f_{\text{comp}}(x) = 001001..$

- Target

확실한 Value를 찾아야함

  - No confusion! **(= no duplication!)**

영리하게.

  - Design a compression function **cleverly**, so that the output bits' length is minimized (= decrease the number of eggs)

jeonghun.park@knu.ac.kr

Source Coding Basic 불확실성을 이용한 Compression

- We call the input x as "source," and the output bits (let's write $\boldsymbol{b = (001001..)}$) as "code"

- Let's assume that the input x is as follows:
  - x can be 1,2,3, or 4
  - The probability is: $$\mathbb{P}[x=1]=\frac{1}{2}$$ $$\mathbb{P}[x=2]=\frac{1}{4}$$ $$\mathbb{P}[x=3]=\frac{1}{8}$$ $$\mathbb{P}[x=4]=\frac{1}{8}$$
  - What is an efficient compression function?

jeonghun.park@knu.ac.kr

Source Coding Basic

- Let's consider this
  - Okay, the number of the sources is 4 (1,2,3,4)
  - Then, we may design a compression function as:

length

$$
\begin{aligned}
f_{\text{comp}}(x=1) &= 00 \quad = \tfrac{1}{4}\\
f_{\text{comp}}(x=2) &= 01 \quad = \tfrac{1}{4}\\
f_{\text{comp}}(x=3) &= 10 \quad = \tfrac{1}{4}\\
f_{\text{comp}}(x=4) &= 11 \quad = \tfrac{1}{4}
\end{aligned}
$$

Uniform 한 경우 length

$\left(\frac{1}{4}\times 2\right) + \cdots + \cdots + \cdots = 2$

(bits length)

  - Check point: There is no confusion
  - We just complete to design **our first compression function**

jeonghun.park@knu.ac.kr

length

0	1
0 1	2
1 0	2
1 00	3

Efficiency

- What is the expected number of bits length of our first design?
  - This is easy. The bits length is $\boxed{2}$
- Can we do better?
  - Let's consider the following:

$$f_{\text{comp}}(x = 1) = 0$$
$$f_{\text{comp}}(x = 2) = 10$$
$$f_{\text{comp}}(x = 3) = 110$$
$$f_{\text{comp}}(x = 4) = 111$$

general 한게 답은 아니다.

(평균 length 줄일려면 자주사용하는거에 bit를 줄이면 됨.)

  - The expected length is $\underset{l_1}{1} \times \frac{1}{2} + \underset{l_2}{2} \times \frac{1}{4} + \underset{l_3}{3} \times \frac{1}{8} + \underset{l_4}{3} \times \frac{1}{8} = \boxed{1.75}$

jeonghun.park@knu.ac.kr

Fundamentals

- The insights behind this technique is:
    - Give a few bits to sources that **<u>less appear</u>**
- One may wonder..
    - Is there the optimal bits length for a given source?
    - And if so, how to design that?
- The above questions are fundamental questions where we try to find the answers

jeonghun.park@knu.ac.kr

헷갈리면 안된다
겹치지 말라 해서
나온것

Kraft Inequality

↳ 무조건 만족해야됨

- Assume that we have m number of sources. Then also assume that our codes' length is $\ell_1, \ell_2, ..., \ell_m$.
  - Then the following should be satisfied

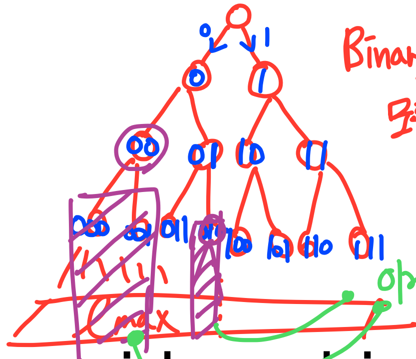
$$\sum_{i=1}^{m} 2^{-\ell_i} \leq 1$$

$l_1=1$ 일때 $\sum_{i=1}^{m} 2^{-l_i}$ 하면
$l_2=2$
$l_3=3$
$l_4=3$

$2^{-1}+2^{-2}+2^{-3}+2^{-3}$
$=\frac{1}{2}+\frac{1}{4}+\frac{1}{8}+\frac{1}{8}=1$

정확하게 1이 나와서 Best Case.

jeonghun.park@knu.ac.kr



Binary tree로 모든 수 표현

Proof

ex) 00 쓰면 000이나 001은 절대 못씀
⇒ Prefix Condition

이거 두개(못쓰는 후손) 교집합 없음

- Consider a binary tree, where each branch represents each code. Then, by the prefix condition (for unique decodability), no code is an ancestor of any other code on this tree. Hence, each code eliminates its descendants as possible codes.

유니크 해야함.

- Let $\ell_{\max}$ be the length of the longest codes. A code at level ℓ_i has $2^{\ell_{\max}-\ell_i}$ descendants at level $\ell_{\max}$

↳ 못쓰는 region

- Since each of the above mentioned descendant must be disjoint, which implies that $\underline{\sum_{i=1}^{m} 2^{\ell_{\max}-\ell_i}} \leq \underline{2^{\ell_{\max}}}$

ℓ_{max}는 뭐라도 상관 X

ℓ_{max} 등 전체

못쓰는 ℓ_{max} 등 합계

jeonghun.park@knu.ac.kr

KYUNGPOOK NATIONAL UNIVERSITY

Some Notes

- Kraft inequality is a clue of the optimal compression function
- If there is a set of codes, and this code length satisfies Kraft inequality, this is the optimal code!

$$\boxed{\sum_{i=1}^{m} 2^{-\ell_i} = 1}$$

jeonghun.park@knu.ac.kr

Optimal Codes

- The expected code length is calculated as

$$L = \sum_{i=1}^{m} p_i \ell_i$$

Probability of the i-th source

- And from Kraft inequality, we have $\sum_{i=1}^{m} 2^{-\ell_i} \leq 1$

- Jointly considering the above two conditions, we obtain the optimal code length is $(\ell_i)^{\star} = \log_2 p_i^{-1}$

Also fits our intuition!

직관

jeonghun.park@knu.ac.kr

Achievability

- Okay, now we know the optimal code length.
- But how to achieve..? This is a different story.

jeonghun.park@knu.ac.kr

Huffman Coding

⇒ 파일 용량압축에 사용

(jpg, png)

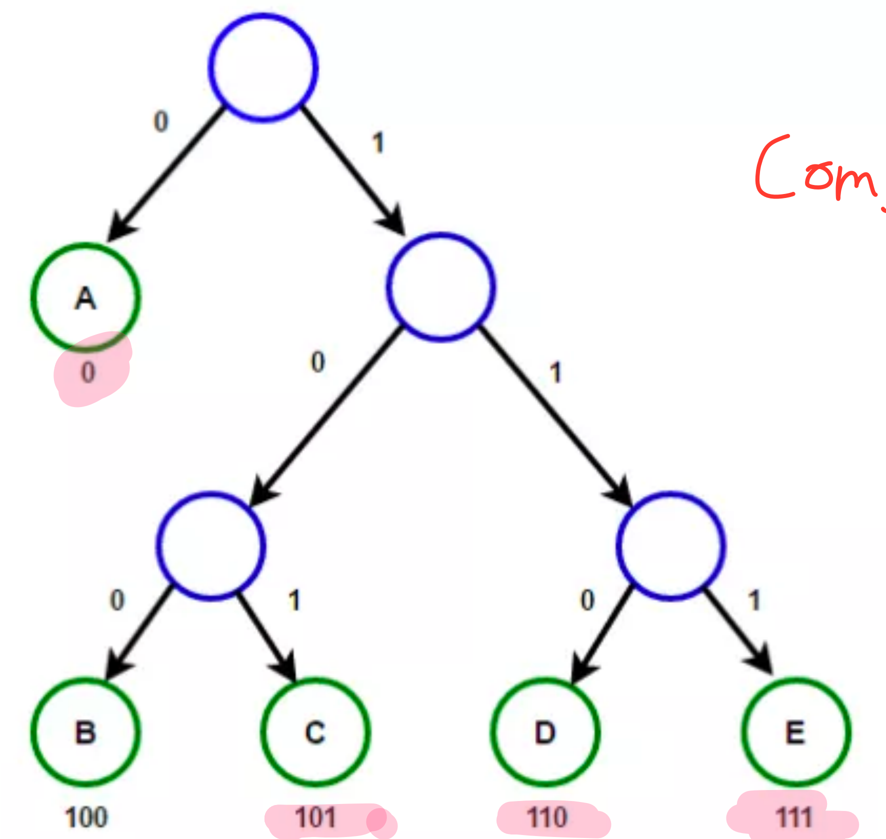
- David Huffman’s student project result
    - During his Ph.D. course!
    - This achieves the optimal code length

jeonghun.park@knu.ac.kr

KNU KYUNGPOOK NATIONAL UNIVERSITY

Huffman Coding

- Huffman coding's key is Huffman tree



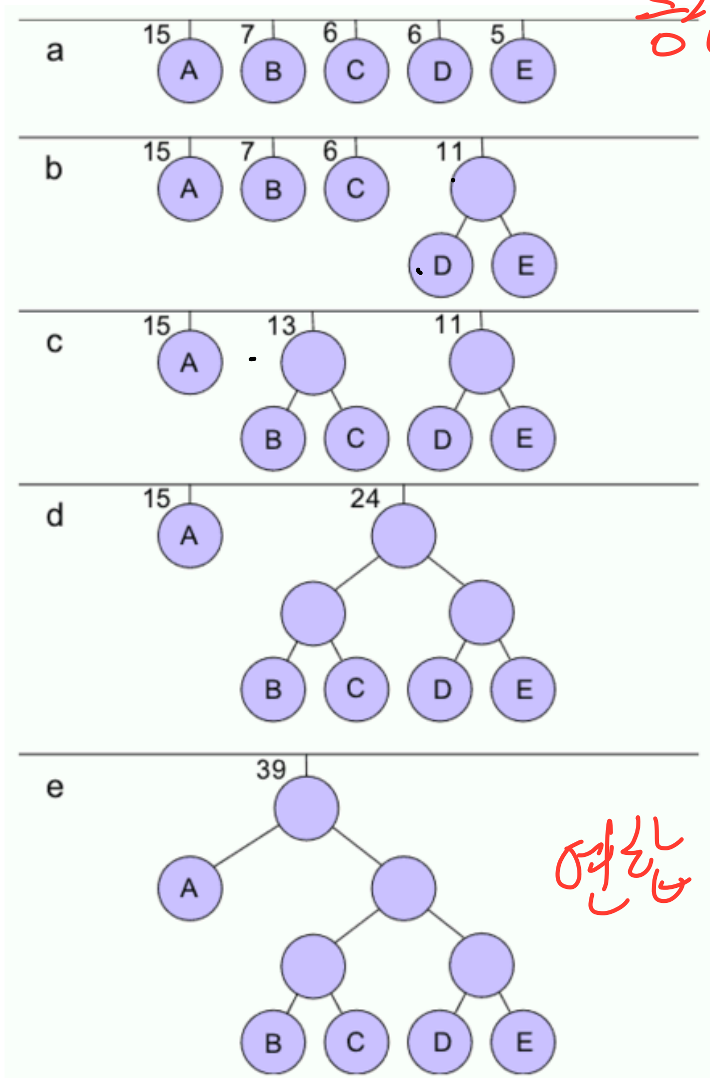
Combined 된 연합체엔 부여 X

jeonghun.park@knu.ac.kr

Huffman Coding

더하고 ordering 다시
... 반복

등장확률이 적은것을
더해감



연합 에 코드부여 안함

jeonghun.park@knu.ac.kr
KNU KYUNGPOOK NATIONAL UNIVERSITY

How to Construct Huffman Tree

- Algorithm description
  1. Order the sources by their appear probability
  2. Add the most less appearing sources
     - They will make a branch
  3. Go to step 1 and repeat

0,1 뿐만 아니라

0,1,2,3 도 가능

(2진법이 optimal 한 방법)

Codeword Length	Codeword	X	Probability				
2	01	1	0.25	0.3	0.45	0.55	1
2	10	2	0.25	0.25	0.3	0.45	
2	11	3	0.2	0.25	0.25		
3	000	4	0.15	0.2			
3	001	5	0.15				

jeonghun.park@knu.ac.kr