ALGORITHM
알고리즘

Lecture 14

스트링 알고리즘 (3)

컴퓨터과학과 | 김진욱 교수

한국방송통신대학교

학습목차

01. 허프만 코딩

허프만 코딩?

Huffman coding

- 스트링에서 각 문자의 빈도 정보를 이용하는 통계적 압축 방법
  - ✓ 빈도수가 큰 문자 → 짧은 코드
  - ✓ 빈도수가 작은 문자 → 긴 코드

⇒ 허프만 트리 이용

문자	코드	빈도
a	11	4
b	0	5
c	101	2
d	100	1

a b a b a b c d b a b c

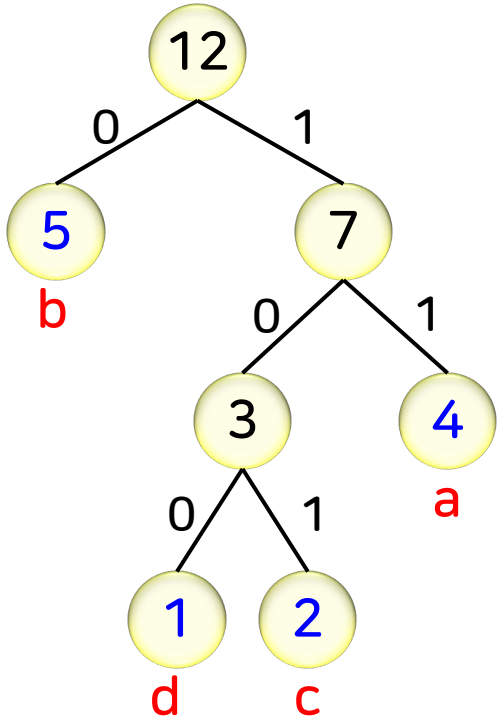
110110110101100011010 1

허프만 트리

각 문자에 이진 코드를 부여하기 위한 전 이진 트리

- 모든 노드는 자식 노드의 개수가 2개 또는 0개(리프 노드)
- 각 리프 노드는 각 문자를 표시

문자	코드	빈도
a	11	4
b	0	5
c	101	2
d	100	1



허프만 트리_생성

욕심쟁이 방법, 상향식

1. 각 문자 → 노드 하나인 트리
   - ✓ 노드의 레이블 → 문자의 빈도수
2. 루트 노드의 레이블이 가장 작은 두 트리 선택 → 한 트리로 합침
   - ✓ 두 루트 노드를 자식 노드로 갖는 새로운 노드 생성
   - ✓ 새로운 노드의 레이블 → 자식 노드의 레이블의 합
   - ✓ 두 간선의 레이블 → 각각 0과 1
3. 남은 트리가 둘 이상 → 2.로 이동

허프만 트리_생성

a b a b a b c d b a b c

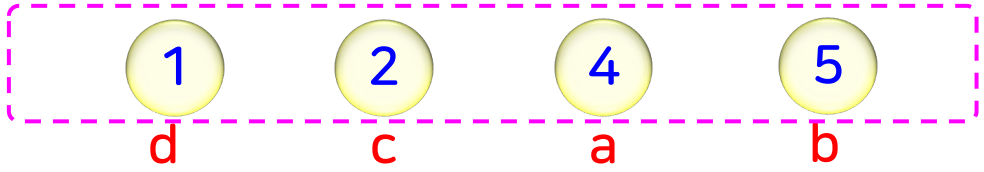
문자	빈도
a	4
b	5
c	2
d	1

허프만 트리_생성

a b a b a b c d b a b c

문자	빈도
a	4
b	5
c	2
d	1

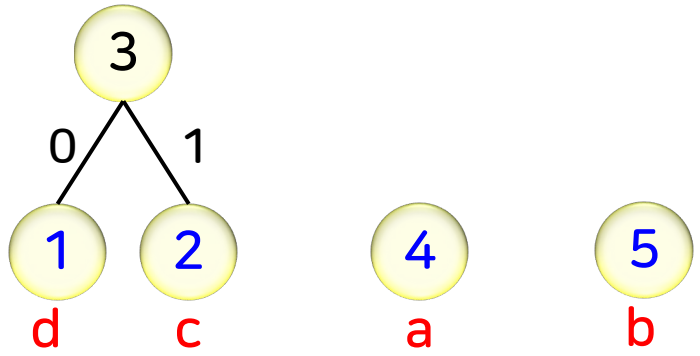
최소 힙



허프만 트리_생성

a b a b a b c d b a b c

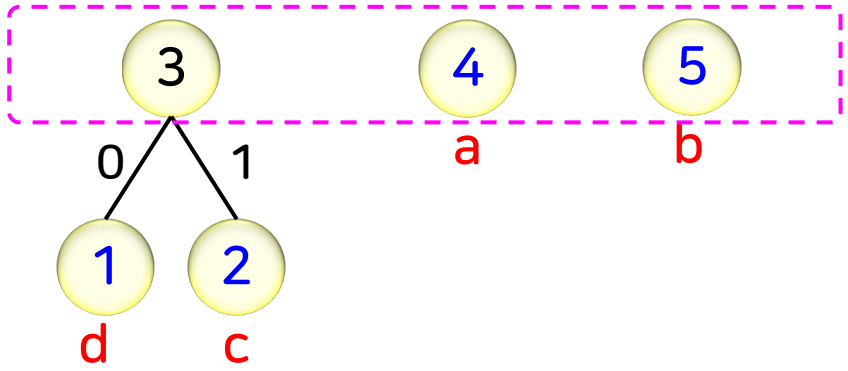
문자	빈도
a	4
b	5
c	2
d	1



허프만 트리_생성

a b a b a b c d b a b c

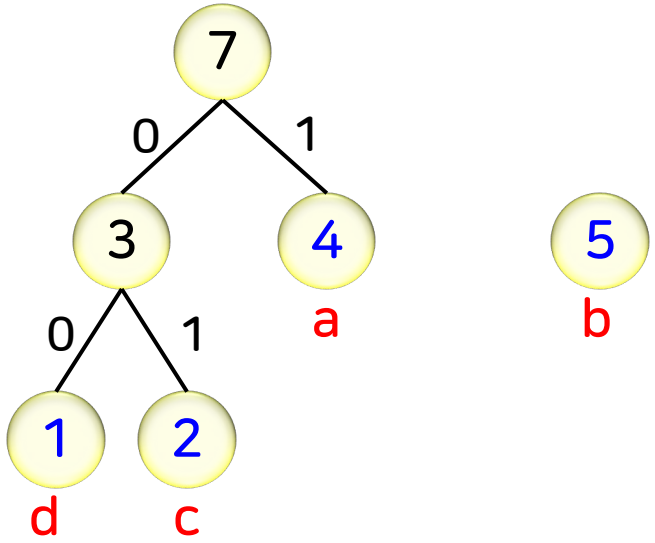
문자	빈도
a	4
b	5
c	2
d	1



허프만 트리_생성

a b a b a b c d b a b c

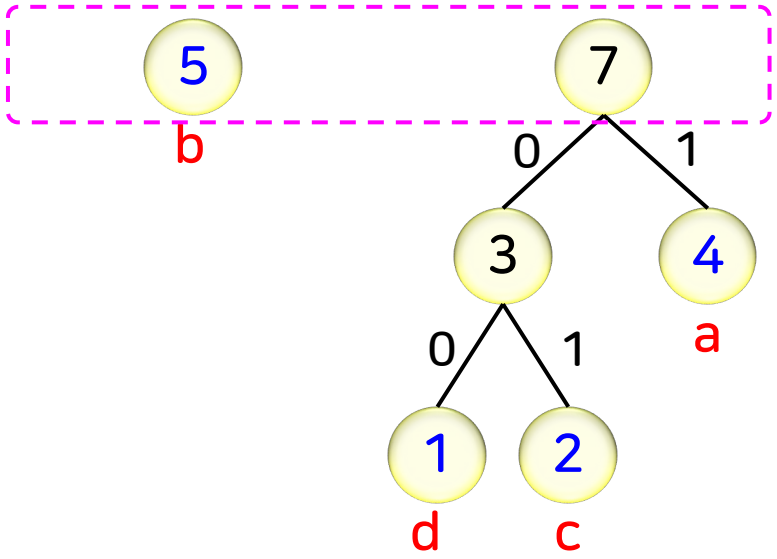
문자	빈도
a	4
b	5
c	2
d	1



허프만 트리_생성

a b a b a b c d b a b c

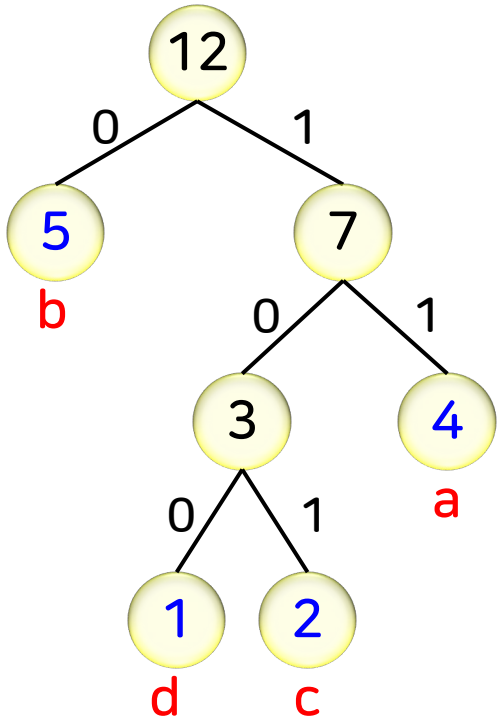
문자	빈도
a	4
b	5
c	2
d	1



허프만 트리_생성

a b a b a b c d b a b c

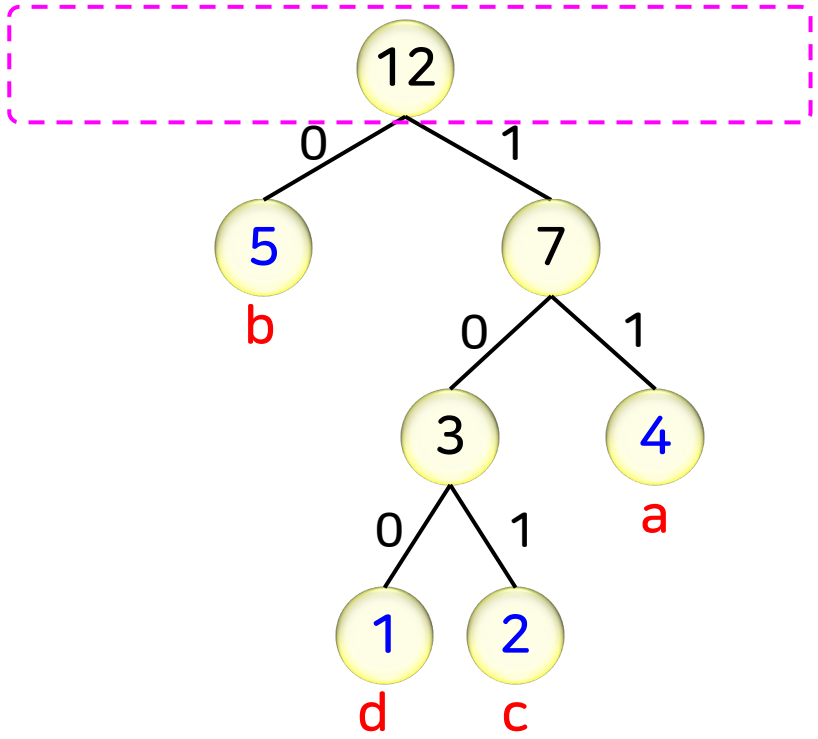
문자	빈도
a	4
b	5
c	2
d	1



허프만 트리_생성

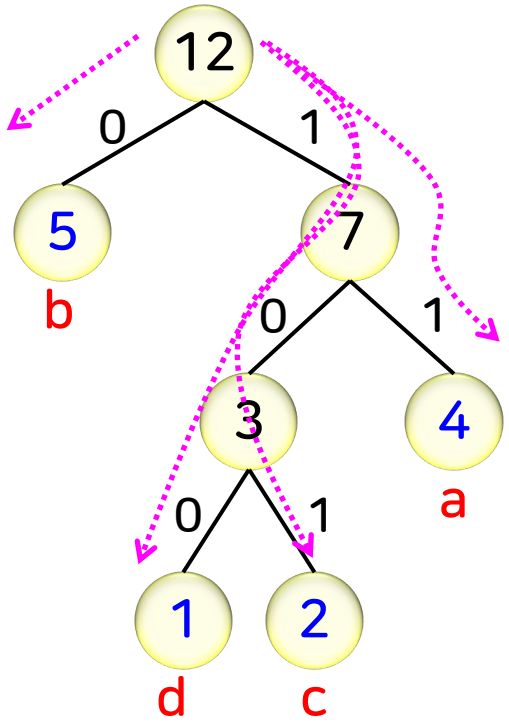
a b a b a b c d b a b c

문자	빈도
a	4
b	5
c	2
d	1



허프만 트리_이진 코드

루트 노드로부터 리프 노드까지의 간선의 레이블



문자	코드
a	11
b	0
c	101
d	100

허프만 코딩_인코딩

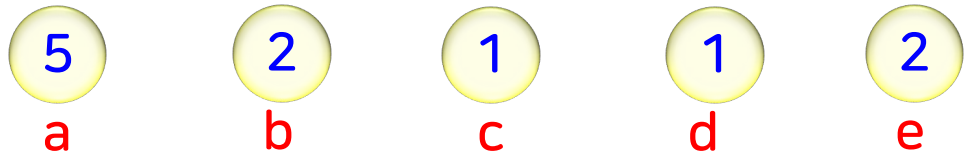
① 주어진 스트링에서 각 문자의 출현 빈도수 계산

② 각 문자의 빈도수를 이용하여 허프만 트리를 만들어 각 문자에 이진 코드 부여

③ 주어진 스트링의 각 문자를 이진 코드로 변환하여 압축된 스트링 생성

a b e a c a d a b e a

문자	코드	빈도
a		5
b		2
c		1
d		1
e		2



허프만 코딩_인코딩

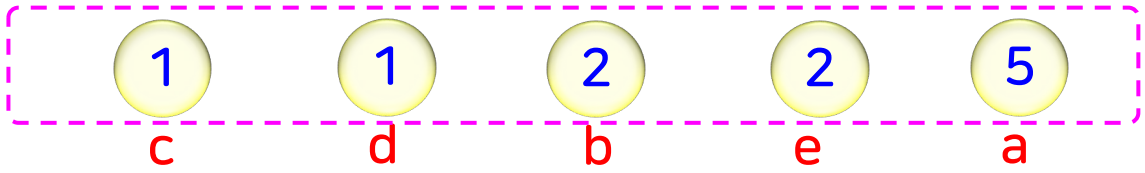
① 주어진 스트링에서 각 문자의 출현 빈도수 계산

② 각 문자의 빈도수를 이용하여 허프만 트리를 만들어 각 문자에 이진 코드 부여

③ 주어진 스트링의 각 문자를 이진 코드로 변환하여 압축된 스트링 생성

a b e a c a d a b e a

문자	코드	빈도
a		5
b		2
c		1
d		1
e		2



허프만 코딩_인코딩

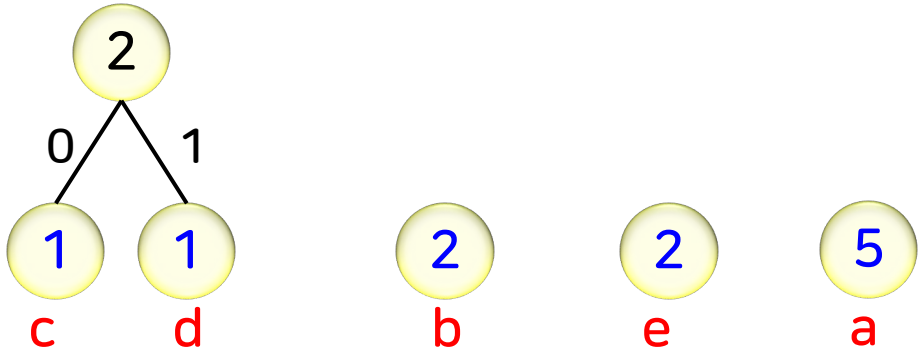
① 주어진 스트링에서 각 문자의 출현 빈도수 계산

② 각 문자의 빈도수를 이용하여 허프만 트리를 만들어 각 문자에 이진 코드 부여

③ 주어진 스트링의 각 문자를 이진 코드로 변환하여 압축된 스트링 생성

a b e a c a d a b e a

문자	코드	빈도
a		5
b		2
c		1
d		1
e		2



허프만 코딩_인코딩

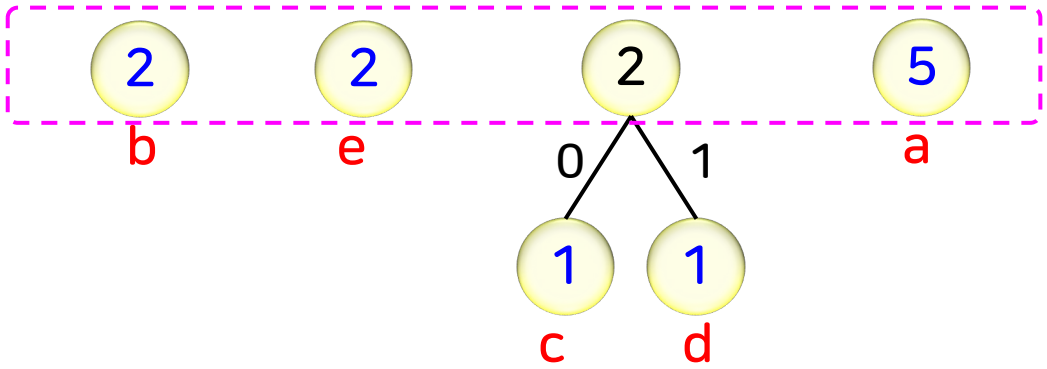
① 주어진 스트링에서 각 문자의 출현 빈도수 계산

② 각 문자의 빈도수를 이용하여 허프만 트리를 만들어 각 문자에 이진 코드 부여

③ 주어진 스트링의 각 문자를 이진 코드로 변환하여 압축된 스트링 생성

a b e a c a d a b e a

문자	코드	빈도
a		5
b		2
c		1
d		1
e		2



허프만 코딩_인코딩

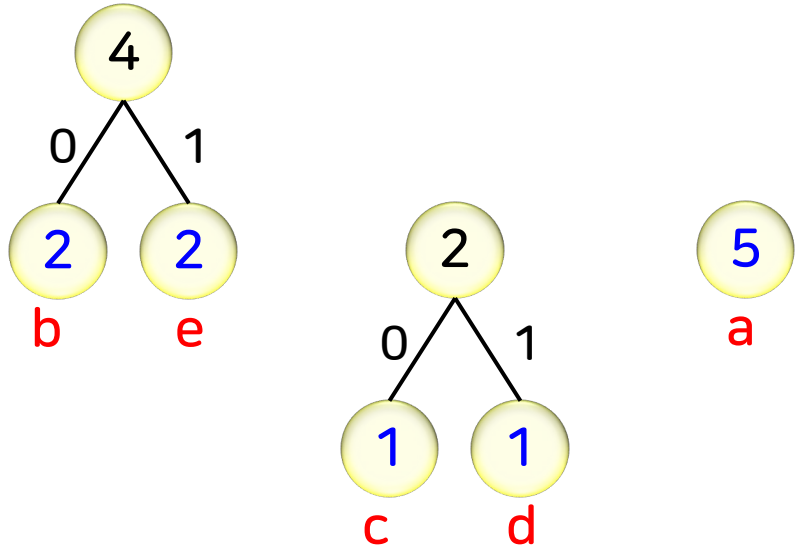
① 주어진 스트링에서 각 문자의 출현 빈도수 계산

② 각 문자의 빈도수를 이용하여 허프만 트리를 만들어 각 문자에 이진 코드 부여

③ 주어진 스트링의 각 문자를 이진 코드로 변환하여 압축된 스트링 생성

a b e a c a d a b e a

문자	코드	빈도
a		5
b		2
c		1
d		1
e		2



허프만 코딩_인코딩

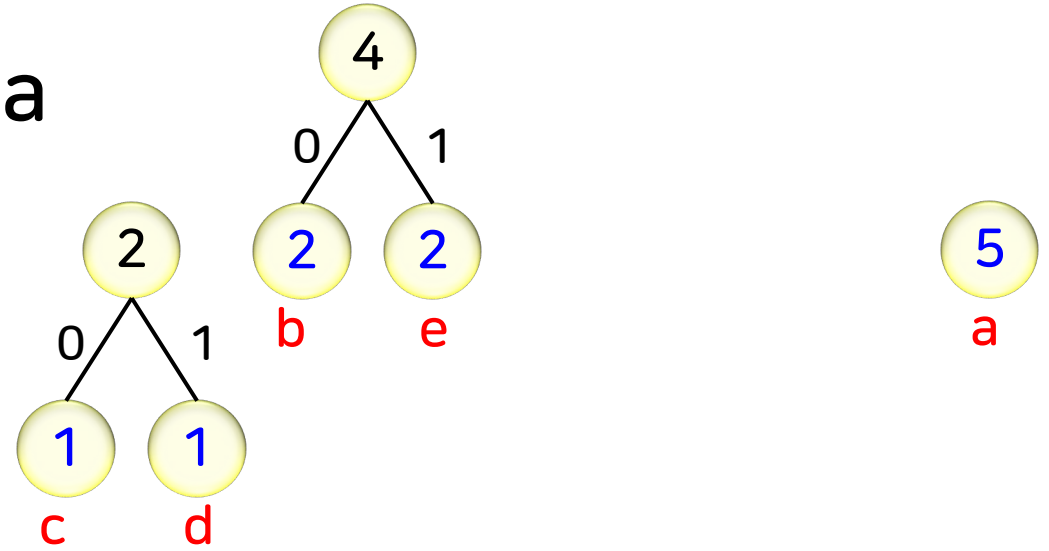
① 주어진 스트링에서 각 문자의 출현 빈도수 계산

② 각 문자의 빈도수를 이용하여 허프만 트리를 만들어 각 문자에 이진 코드 부여

③ 주어진 스트링의 각 문자를 이진 코드로 변환하여 압축된 스트링 생성

a b e a c a d a b e a

문자	코드	빈도
a		5
b		2
c		1
d		1
e		2



허프만 코딩_인코딩

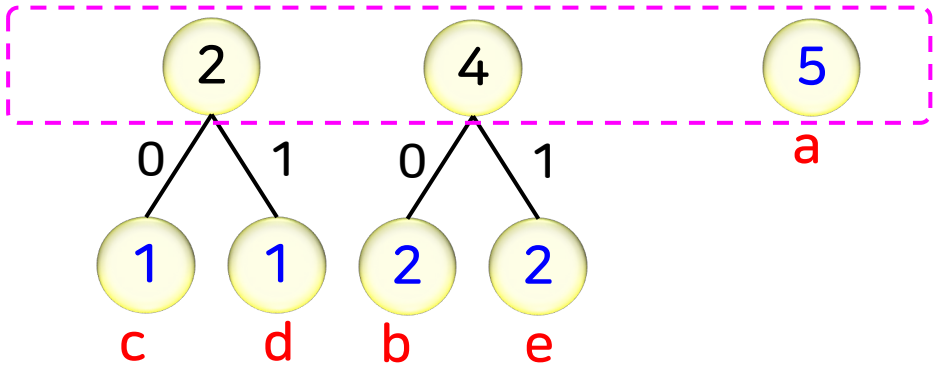
① 주어진 스트링에서 각 문자의 출현 빈도수 계산

② 각 문자의 빈도수를 이용하여 허프만 트리를 만들어 각 문자에 이진 코드 부여

③ 주어진 스트링의 각 문자를 이진 코드로 변환하여 압축된 스트링 생성

a b e a c a d a b e a

문자	코드	빈도
a		5
b		2
c		1
d		1
e		2



허프만 코딩_인코딩

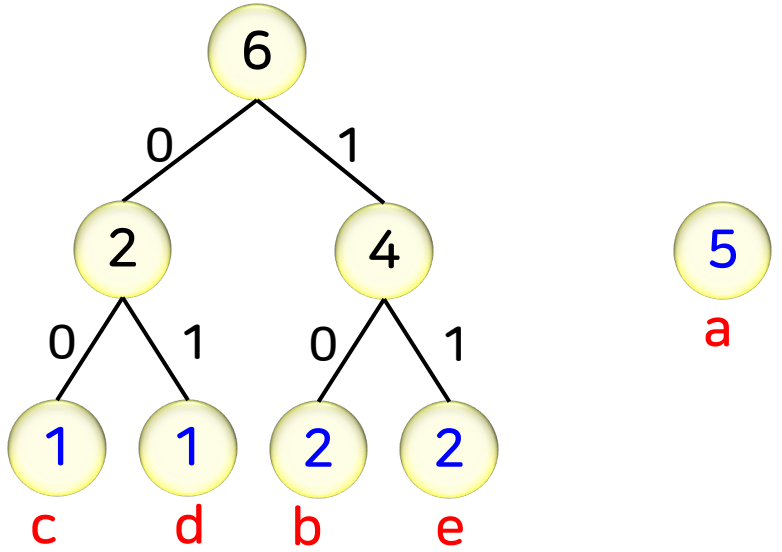
① 주어진 스트링에서 각 문자의 출현 빈도수 계산

② 각 문자의 빈도수를 이용하여 허프만 트리를 만들어 각 문자에 이진 코드 부여

③ 주어진 스트링의 각 문자를 이진 코드로 변환하여 압축된 스트링 생성

a b e a c a d a b e a

문자	코드	빈도
a		5
b		2
c		1
d		1
e		2



허프만 코딩_인코딩

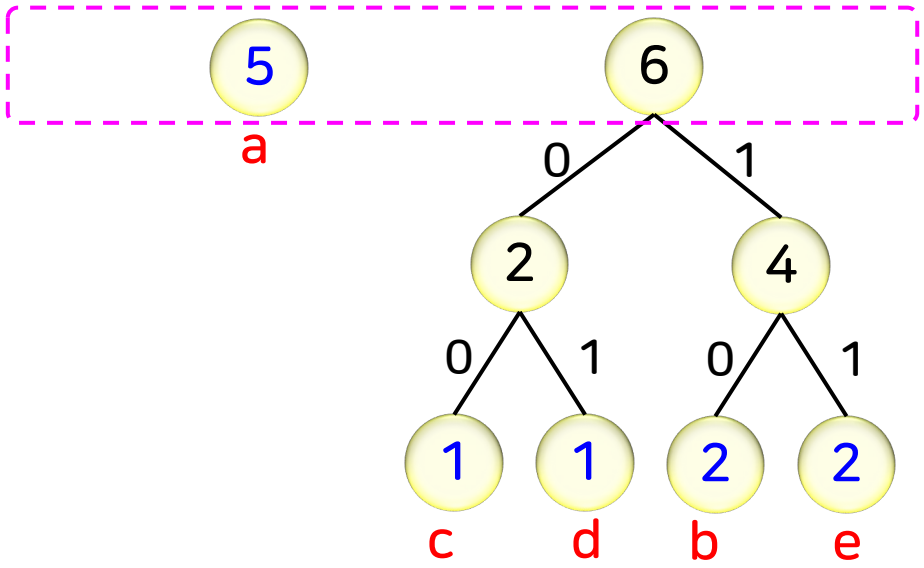
① 주어진 스트링에서 각 문자의 출현 빈도수 계산

② 각 문자의 빈도수를 이용하여 허프만 트리를 만들어 각 문자에 이진 코드 부여

③ 주어진 스트링의 각 문자를 이진 코드로 변환하여 압축된 스트링 생성

a b e a c a d a b e a

문자	코드	빈도
a		5
b		2
c		1
d		1
e		2



허프만 코딩_인코딩

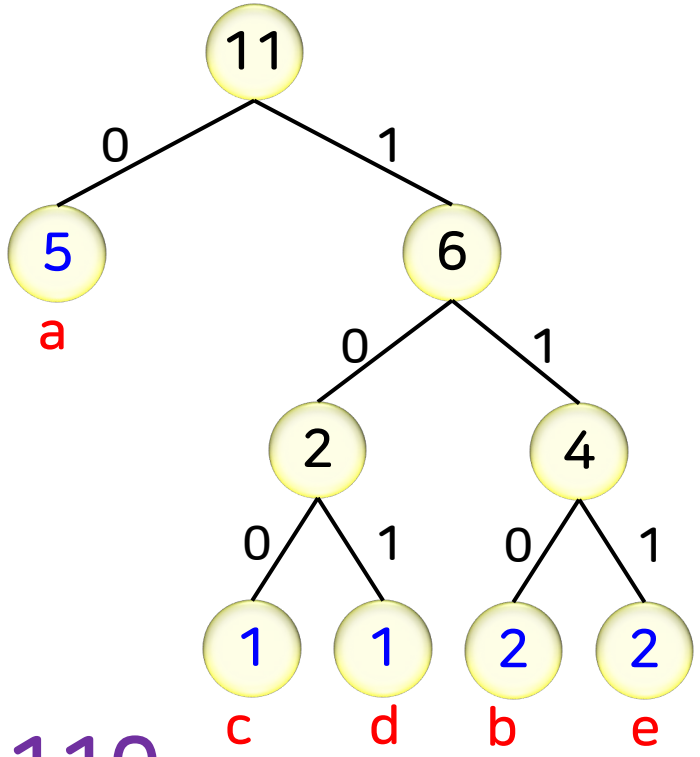
① 주어진 스트링에서 각 문자의 출현 빈도수 계산

② 각 문자의 빈도수를 이용하여 허프만 트리를 만들어 각 문자에 이진 코드 부여

③ 주어진 스트링의 각 문자를 이진 코드로 변환하여 압축된 스트링 생성

a b e a c a d a b e a

문자	코드	빈도
a	0	5
b	110	2
c	100	1
d	101	1
e	111	2



01101111010001010111011110

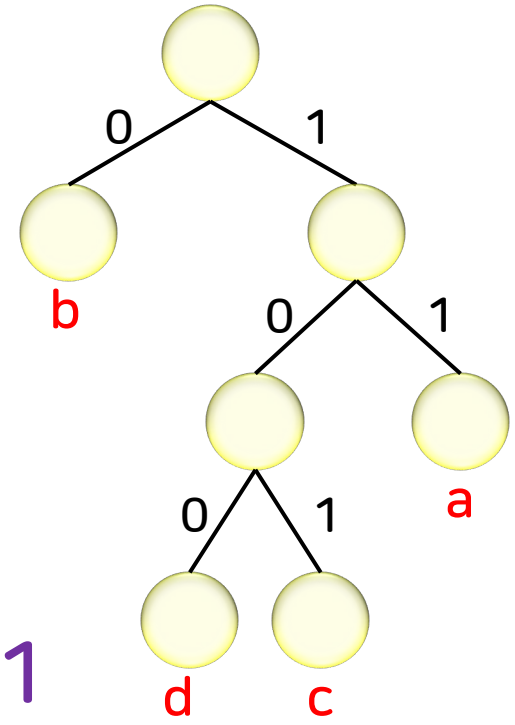
허프만 코딩_디코딩

① 허프만 트리의 루트 노드에 위치

② 압축된 스트링을 1비트씩 읽으며 자식 노드로 내려감

③ 리프 노드에 도착하면 그 노드에 해당하는 문자 출력 후 ①로 이동

문자	코드
a	11
b	0
c	101
d	100



1101101101011000110101

허프만 코딩_디코딩

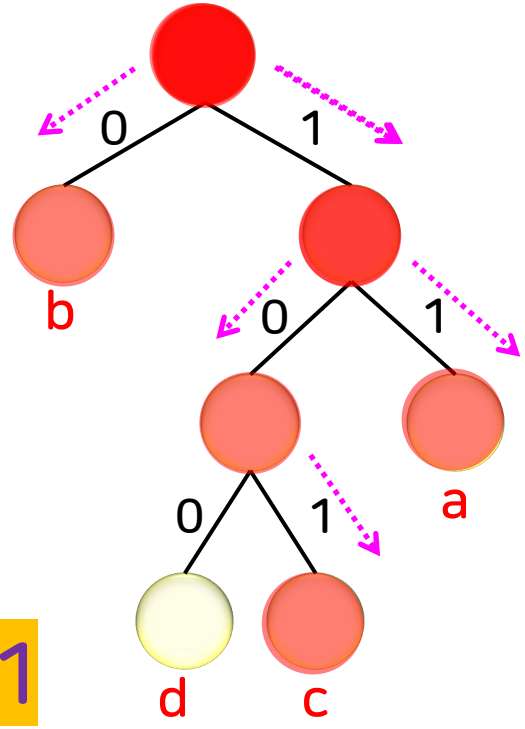
① 허프만 트리의 루트 노드에 위치

② 압축된 스트링을 1비트씩 읽으며 자식 노드로 내려감

③ 리프 노드에 도착하면 그 노드에 해당하는 문자 출력 후 ①로 이동

문자	코드
a	11
b	0
c	101
d	100

ababababcdbabc



1101101101011000110101

성능과 특징

인코딩 성능 → O(|Σ|log|Σ|+n)

- 각 문자의 빈도수 계산 → O(n)
- 허프만 트리 생성 및 이진 코드 생성 → O(|Σ|log|Σ|)
  - ✓ 최소 힙 구축 → O(|Σ|)
  - ✓ 알파벳 크기(|Σ|)만큼 힙에서 최솟값 삭제와 삽입(O(log|Σ|)) 반복하며 트리 생성
  - ✓ 트리 순회하며 각 문자의 이진 코드 추출 → O(|Σ|)
- 스트링의 길이 만큼 수행 → O(n)

디코딩 성능 → O(|Σ|log|Σ|+m)

- 허프만 트리 생성 → O(|Σ|log|Σ|)
- 압축된 비트 스트링의 길이 만큼 수행 → O(m)

성능과 특징

각 문자에 부여하는 코드는 접두부 코드이며 최적의 코드

- 접두부 코드 → 각 문자에 부여된 이진 코드가 다른 문자에 부여된 이진 코드의 접두부가 되지 않는 것

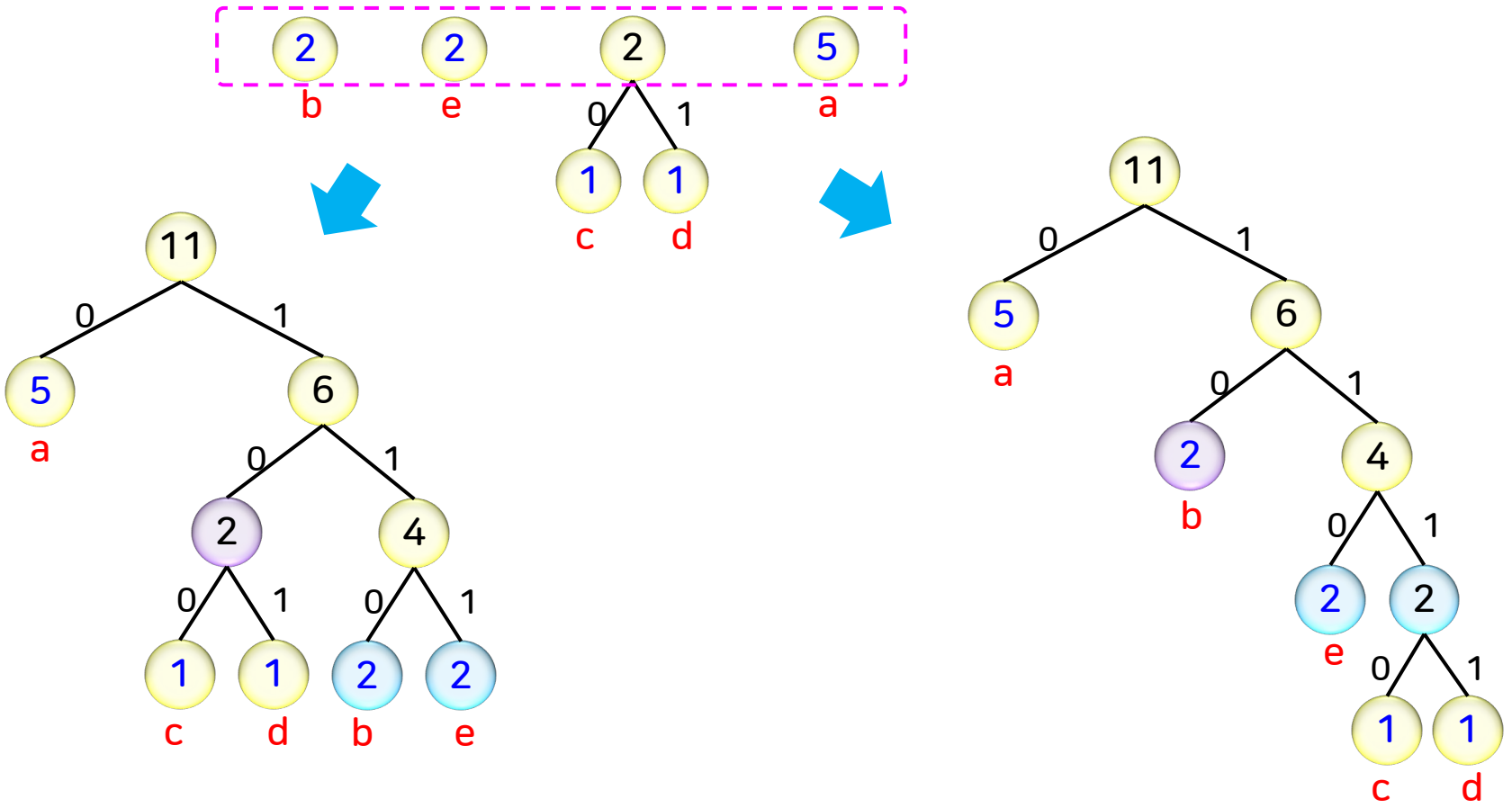
문자	코드
a	11
b	0
c	101
d	100

- 최적의 코드 → 인코딩된 스트링의 길이가 가장 짧음

성능과 특징

허프만 트리는 유일하지 않을 수 있음

- 같은 빈도수를 갖는 노드들이 존재하면 허프만 트리 모양이 달라질 수 있음

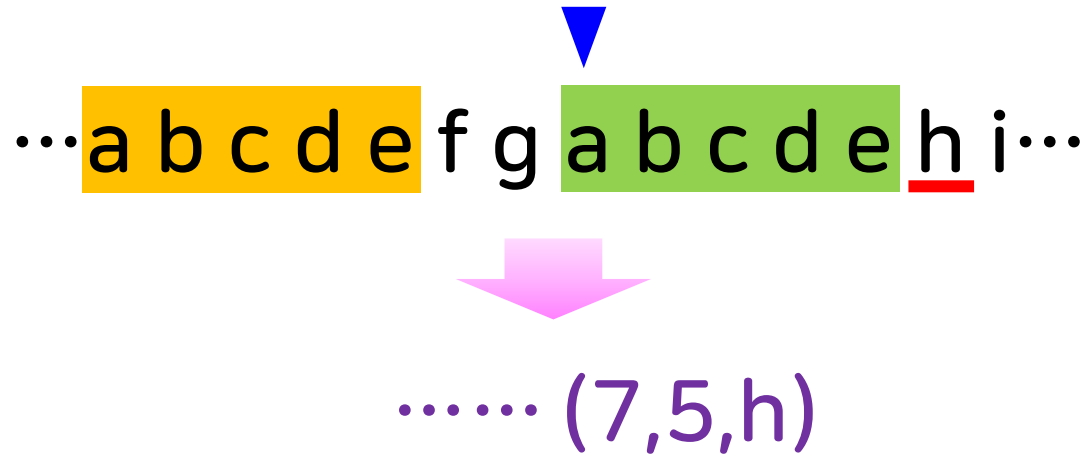


02. LZ77

LZ77?

Lempel-Ziv 77

- 스트링에서 한 번 나왔던 문자열이 다시 나오면 앞서 나온 위치와 길이 그리고 바로 다음 문자로 변환하여 압축하는 방법



LZ77?

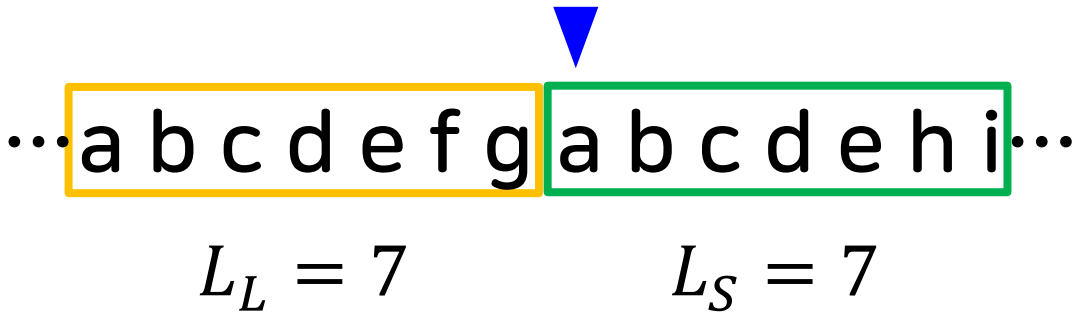
Lempel-Ziv 77

- 스트링에서 한 번 나왔던 문자열이 다시 나오면 앞서 나온 위치와 길이 그리고 바로 다음 문자로 변환하여 압축하는 방법

슬라이딩 윈도

Sliding window

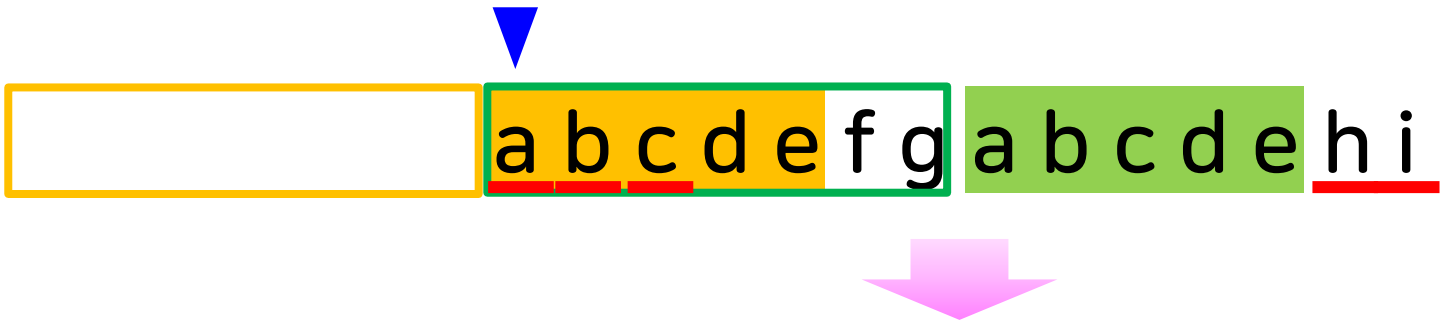
- 스트링에서 일정 크기의 범위만 참조하도록 제한
- 위치 이동 가능



LZ77_인코딩

- 주어진 스트링을 차례로 보며 슬라이딩 윈도 안에서 (매치의 시작 위치 사이의 거리, 매치 길이, 다음 문자)로 변환

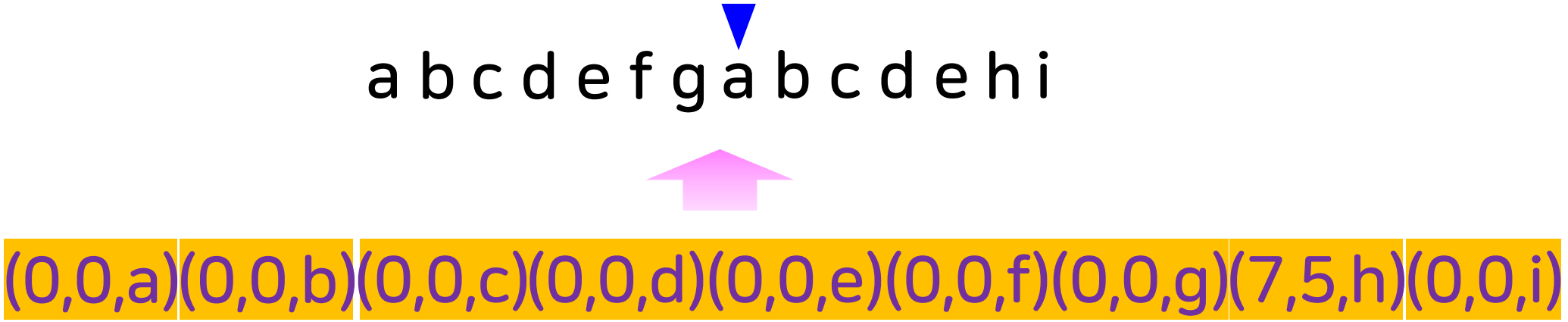
(예) 슬라이딩 윈도 크기 → $L_L = 7,\ L_S = 7$



(0,0,a)(0,0,b)(0,0,c)(0,0,d)(0,0,e)(0,0,f)(0,0,g)(7,5,h)(0,0,i)

LZ77_디코딩

- 압축된 데이터를 차례로 보며 디코딩 중인 스트링에서 거리만큼 앞쪽에서 길이만큼 문자를 출력 후 다음 문자 출력



성능과 특징

허프만 코딩 알고리즘과 함께 zip에 사용됨

- DEFLATE (RFC 1951) 알고리즘

LZ78, LZW 등 다양한 LZ- 계열 알고리즘 존재

03. 영상 압축

영상 압축

2차원 이미지나 3차원 동영상이 갖는 특성을 압축에 반영

- 2차원 이미지 → 인접한 픽셀 정보의 유사성
- 3차원 동영상 → 인접한 시간의 영상들의 유사성

2차원 이미지 압축 종류

- JPEG, GIF, TIFF, PNG 등

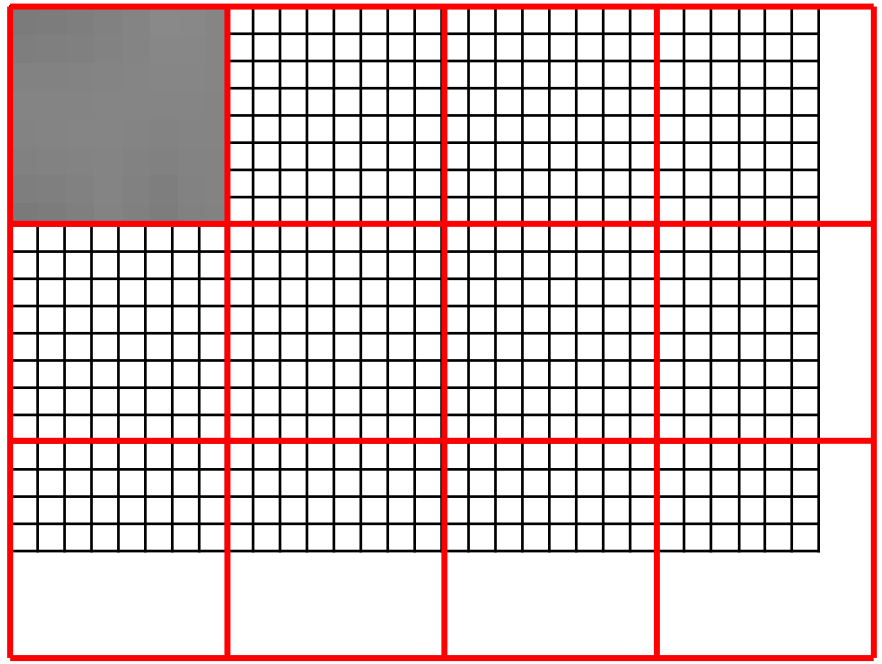
3차원 동영상 압축 종류

- MPEG-1, MPEG-2, MPEG-4 등

JPEG 표준_인코딩

2차원 데이터를 단위 블록으로 나눈 후 블록별로 압축

- 블록화 → DCT → 양자화 → 엔트로피 코딩



JPEG 표준_인코딩

2차원 데이터를 단위 블록으로 나눈 후 블록별로 압축

- 블록화 → DCT → 양자화 → 엔트로피 코딩

-128

123	124	126	129	132	137	136	133
126	127	128	130	132	135	135	133
130	130	131	132	133	134	134	133
132	132	132	133	133	133	133	132
131	132	133	133	132	131	132	132
129	130	131	132	130	129	131	131
126	127	129	132	129	126	130	131
122	124	127	131	128	125	129	130

JPEG 표준_인코딩

2차원 데이터를 단위 블록으로 나눈 후 블록별로 압축

- 블록화 → DCT → 양자화 → 엔트로피 코딩

-128 Discrete Cosine Transform

-5	-4	-2	1	4	9	8	5
-2	-1	0	2	4	7	7	5
2	2	3	4	5	6	6	5
4	4	4	5	5	5	5	4
3	4	5	5	4	3	4	4
1	2	3	4	2	1	3	3
-2	-1	1	4	1	-2	2	3
-6	-4	-1	3	0	-3	1	2

JPEG 표준_인코딩

2차원 데이터를 단위 블록으로 나눈 후 블록별로 압축

- 블록화 → DCT → 양자화 → 엔트로피 코딩

Discrete Cosine Transform

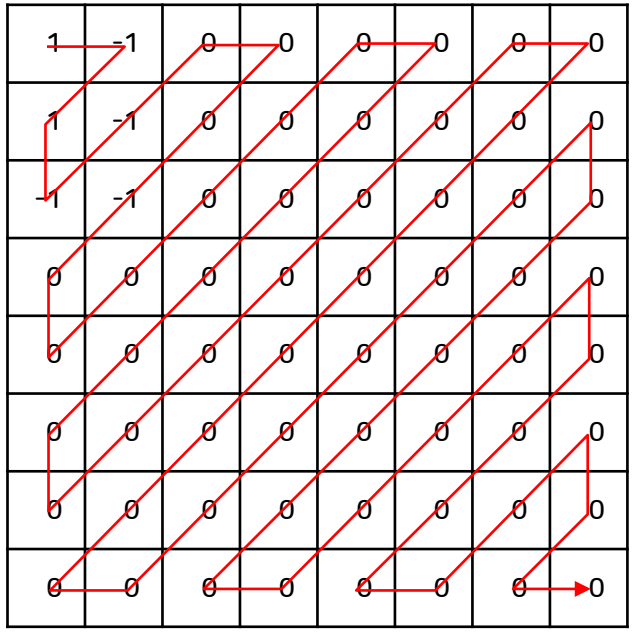
19.5	-12.4	-4.5	-2.2	0.3	2.4	-1.7	-0.3
8.0	-7.6	-0.1	7.4	-3.7	-1.1	2.0	-0.6
-11.4	-9.5	-2.1	-0.3	0.1	0.9	-0.5	-0.5
0.2	-0.8	0.3	0.5	-1.1	0.0	0.3	-0.2
-0.5	-1.0	-1.0	0.4	-0.3	-0.1	0.2	-0.1
0.7	0.2	0.1	0.5	0.5	0.3	-0.1	-0.1
-0.1	-0.3	-0.2	0.3	-0.3	0.0	0.4	0.0
0.6	0.0	-0.1	-0.7	0.3	0.0	0.2	0.3

JPEG 표준_인코딩

2차원 데이터를 단위 블록으로 나눈 후 블록별로 압축

- 블록화 → DCT → 양자화 → 엔트로피 코딩

quantization



1 -1 1 -1 -1 0 0 0 -1 0 0 0 … 0

JPEG 표준_디코딩

인코딩의 역순으로 진행

- 엔트로피 디코딩 → 역양자화 → 역DCT

1 -1 1 -1 -1 0 0 0 -1 0 0 0 … 0

손실 압축

122	123	125	128	131	134	136	138
125	126	128	130	132	134	136	136
130	130	131	132	133	133	134	134
133	133	133	133	133	132	132	132
133	133	133	132	132	131	131	130
131	131	130	130	130	129	129	129
127	127	127	127	127	128	128	128
124	124	125	125	126	127	127	128

≒

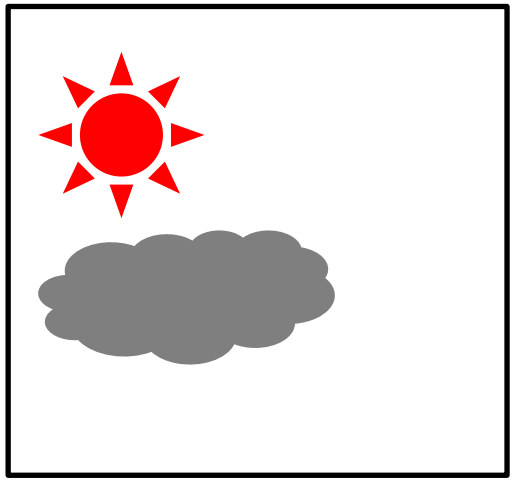
원래 이미지

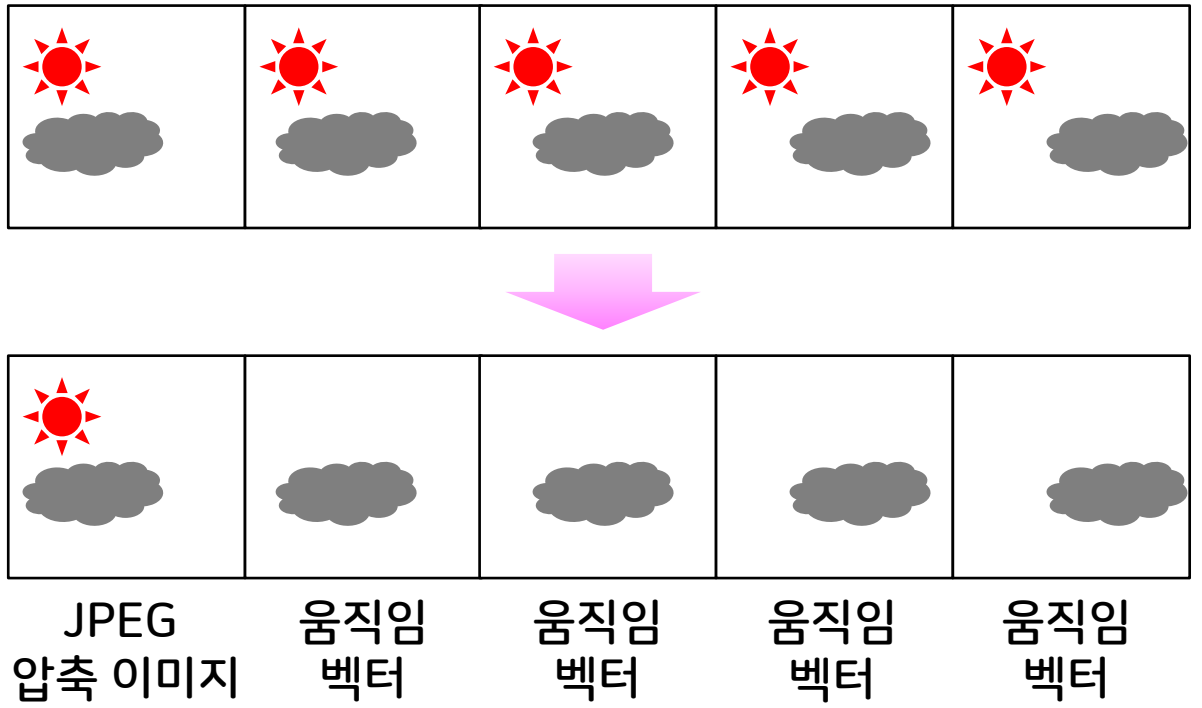
123	124	126	129	132	137	136	133
126	127	128	130	132	135	135	133
130	130	131	132	133	134	134	133
132	132	132	133	133	133	133	132
131	132	133	133	132	131	132	132
129	130	131	132	130	129	131	131
126	127	129	132	129	126	130	131
122	124	127	131	128	125	129	130

MPEG 표준

시간순으로 나열된 2차원 데이터를 연속된 시간의 특성을 이용하여 압축

- 움직임 벡터 → 움직임 정보를 가로와 세로의 변화량으로 표현





정리하기

1. 허프만 코딩

- 스트링에서 각 문자의 빈도 정보를 이용하는 통계적 압축 방법
- 허프만 트리 이용
- 각 문자에 부여하는 코드는 접두부 코드이며 최적의 코드
- 성능-인코딩 $O(|\Sigma|\log|\Sigma|+n)$

2. LZ77

- 스트링에서 한 번 나왔던 문자열이 다시 나오면
  앞서 나온 위치와 길이 그리고 바로 다음 문자로 변환하여 압축하는 방법
- 슬라이딩 윈도 이용
- 허프만 코딩 알고리즘과 함께 zip에 사용됨

3. 영상 압축

- 2차원 이미지나 3차원 동영상이 갖는 특성을 압축에 반영
- JPEG, MPEG

ALGORITHM
알고리즘

다음시간에는

Lecture 15

NP-완전 문제

컴퓨터과학과 | 김진욱 교수

한국방송통신대학교