알고리즘 5장 -
데이터 압축 알고리즘

멀티미디어공학과
조 미 경 교수

RLE(Run-Length Encoding)

- 동일 문자가 연속적으로 나타나면 이를 반복 개수를 줄여서 표현하는 방법

aaaaaefbbbbbbcdefg

↓

&5aef&6bcdefg

- 반복 개수가 3이상일 때 효과적
- 일반 문서에서 셋 이상 반복되는 문자가 희박
- 화상(image)데이터 압축에는 효과적

Huffman 코딩

- 문자가 텍스트에 나오는 빈도에 따라 다른 길이의 부호를 부여
  - 많이 나타나는 문자에 짧은 부호를 부여하고 드물게 나타나는 문자에는 긴 부호를 부여하여 전체 메시지의 길이를 줄이려는 방법
- 알파벳 Σ의 각 문자에 대응하는 부호를 코드워드(codeword)라 함

[예제] 'abracadabra'의 코딩 예.

- 영문 알파벳 26자. 5비트로 인코딩 → 55비트 필요.
- 빈도수
  - a => 5번, b => 2번, c와d => 1번, r => 2번
  - 코드

    a = '0', b = '1', r = '00', c = '01', d = '10'

    01000010100100 0 → 15비트

    → 디코딩 불가
- 빈도에 따라 단순하게 코드워드를 생성할 때의 문제점

Huffman 코딩

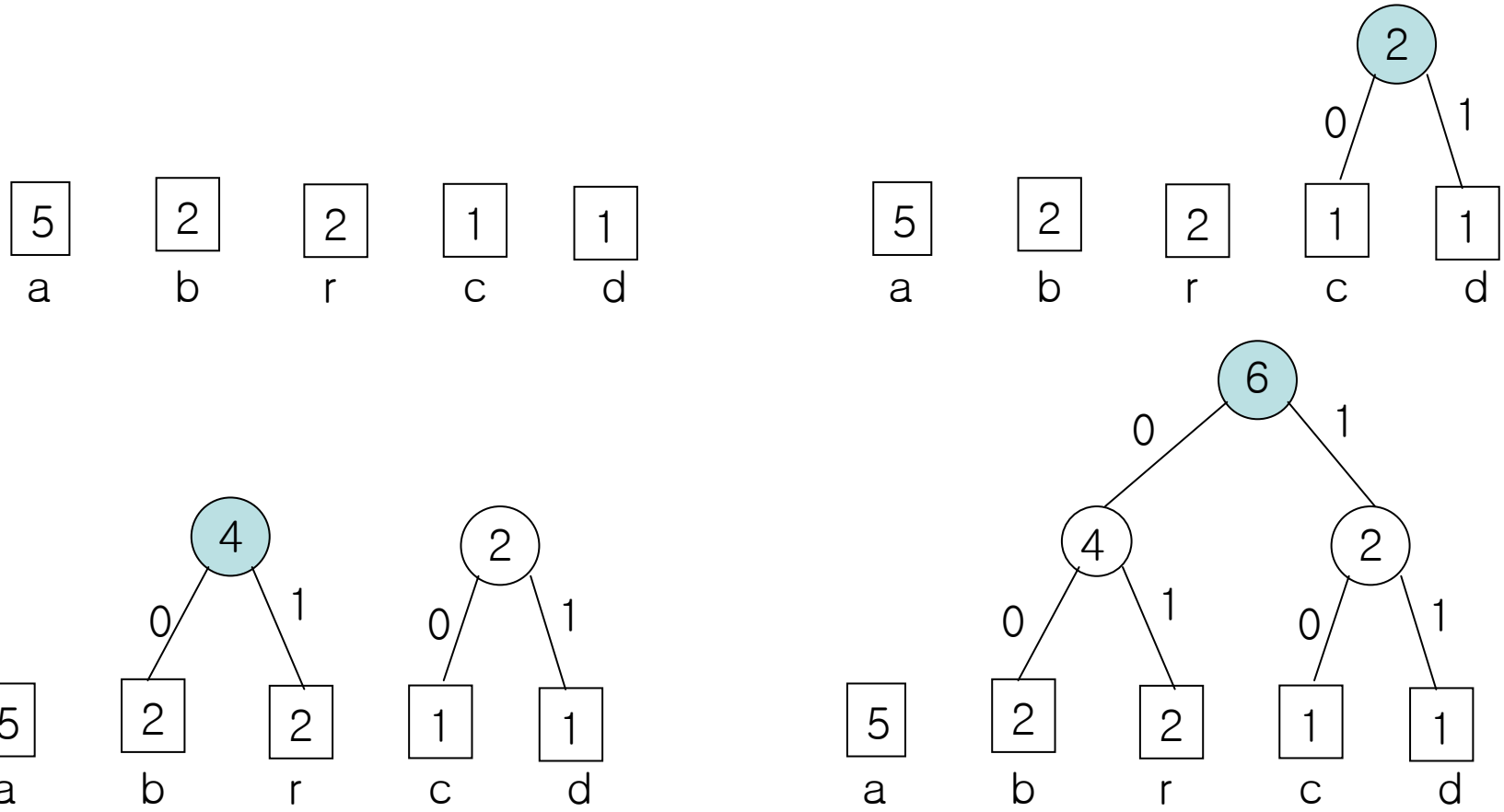
- 모호함이 없는 디코딩 가능하기 위해서는 어떠한 코드워드도 다른 코드워드의 접두부와 일치하지 않아야 함
- 접두부 코드(prefix code)
  - 어떠한 코드워드도 다른 코드워드의 접두부와 일치하지 않는 코드
  - 인코딩된 메시지의 길이가 가장 짧은 최적 코드
- 접두부 불일치 코드(prefix-free code)

허프만 코드

- 접두부 코드임
- 허프만 나무를 상향식으로 만들어 코드 부여 함
  - 이진나무
  - 알파벳의 각 문자는 잎에 나타남
  - 노드는 빈도수로 레이블. 좌우의 두 간선은 각각 0과 1로 레이블한다.
  - 레이블이 작은 노드들을 합쳐 부모 노드를 만들어 나가는데 이때 부모 노드는 자식 노드 레이블의 합으로 레이블됨.

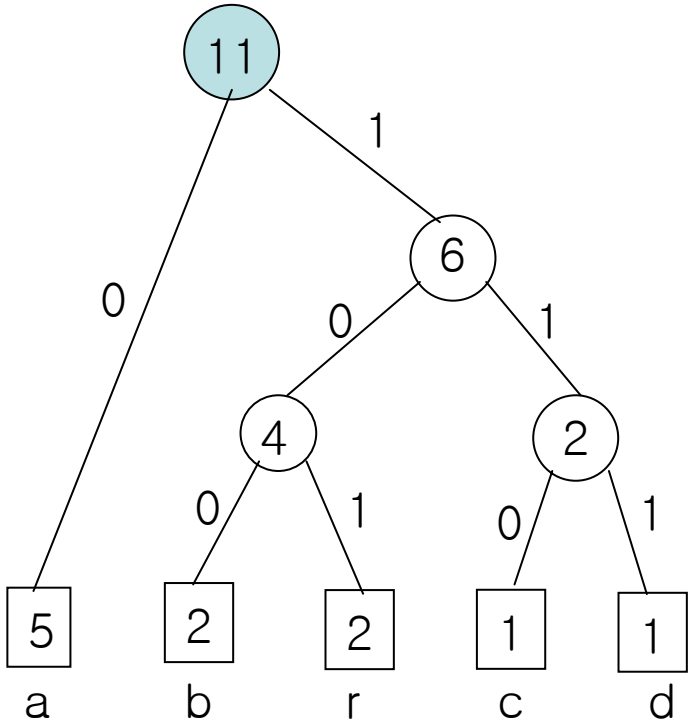
허프만 나무

'abracadabra'



허프만 나무

'abracadabra'



허프만 나무는 유일하지 않음

01001010110011101001010

허프만 코드

- 빈도수(또는 확률)를 알아야 코드를 구할 수 있음
  - 모르면 텍스트를 두 번 읽어야 함
- 디코딩은 허프만 나무 구조와 알파벳에 관한 정보가 있어야 가능
  - 최대 2|Σ|-1개 노드
  - 원래 k비트 코드였다면, 약 2|Σ|+k|Σ| 비트 정보가 헤더에 공급되어야 함

허프만 나무

- 전이진나무(full binary tree)
- 나무 T의 비용 D(T)

$$D(T) = \sum_{i=1}^{n} f_i d_T(c_i)$$

$d_T(l)$:나무 T에서 잎 l의 깊이

- 허프만 나무 T는 D(T)가 최소인 나무.

허프만 나무 알고리즘

```
Huffman(C, F, n)  /* Huffman tree의 생성 */
{
  //입력 : n:알파벳의 크기, C[1..n]:알파벳, F[1..n]:문자 의 빈도
  //출력 : 허프만 나무
    빈도 F[]에 따라 우선순위 큐 Q 생성;
    // 큐의 각 원소는 문자와 빈도의 쌍으로 구성
    for (i=1; i<n ; i+) {
      Q에서 최소값을 삭제하여 이를 u로 함;
      Q에서 최소값을 삭제하여 이를 v로 함;
      새 노드 x를 생성하여 u와 v를 각각 x의 왼쪽, 오른쪽 자식으로 함;
      노드 x의 빈도를 u와 v의 빈도의 합으로 함;
    }
}
```

허프만 나무 알고리즘 정확성

[보조정리] 최적 코드에서 $f_i<f_j$이면 $|c_i|\geq|c_j|$이다.

[보조정리] 빈도가 가장 낮은 두 코드워드는 마지막 비트는 다르지만 같은 길이가 되도록 하는 최적 접두부 코드가 존재한다.

[보조정리] 알파벳 $S=\{a_1,a_2,\ldots,a_n\}$와 각 a_i에 대응하는 빈도 f_{a_i}에 대한 최적 접두부 코드를 나타낸 전 이진 나무를 T라고 하자. T의 잎이면서 동기로 나타나는 두 노드 u, v의 문자가 x, y라고 하자. z를 u와 v의 부모 노드 w의 문자라고 하고, $f_z=f_x+f_y$ 라고 하면, $T'=T-\{u,v\}$ 는 알파벳 $S'=(S-\{x,y\})\cup\{z\}$대한 최적 접두부 코드를 나타내는 나무이다.

동적 허프만 코딩

- adaptive Huffman coding 이라고도 한다
- 문자의 빈도수를 만들어나가면서 코딩을 하는 방법. 텍스트를 한 번만 읽으면 된다
- 빈 나무에서 시작해서 텍스트의 문자를 하나씩 읽어 들이면서 Huffman 나무를 만듦과 동시에 인코딩/디코딩을 행한다

동적 허프만 인코딩

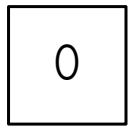
- '$'를 텍스트에 나타나지 않는 문자라고 가정. '$'의 빈도는 0.

```
Adaptive_Huffman(Text) {
  // 입력 : Text – 텍스트
  //  출력 : 압축된 기호, 처리 과정에서 허프먼 나무 T를 만듦

    빈도는 0이고 이름은 $인 한 노드로 구성된 초기 나무 T를 생성;
    while (Text의 끝이 아님) {
      x = Text의 다음 문자;
      if (x가 T에 이미 있던 문자) {
         T에서 x에 부여된 코드워드 출력;
         x의 빈도를 하나 증가;
      }
      else {
         $의 현재 코드워드와 x의 기호 출력;
         x를 빈도 1로 하여 T에 삽입;
      }
      ADJUST(T);
    }
}
```

동적 인코딩 과정

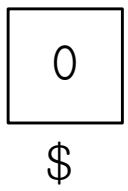
[예제] 텍스트가 'abcdabedbad'라고 하자. a, b, c, d, e의 원래 코드는 각각 5비트 코드로서 00001, 00010, 00011, 00100, 00101라고 하자. 동적 인코딩 과정?



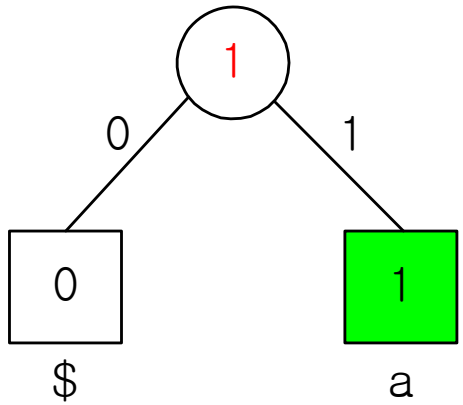
(a) 초기 상태

동적 인코딩 과정

[예제] 텍스트가 'abcdabedbad'라고 하자. a, b, c, d, e의 원래 코드는 각각 5비트 코드로서 00001, 00010, 00011, 00100, 00101라고 하자. 동적 인코딩 과정?



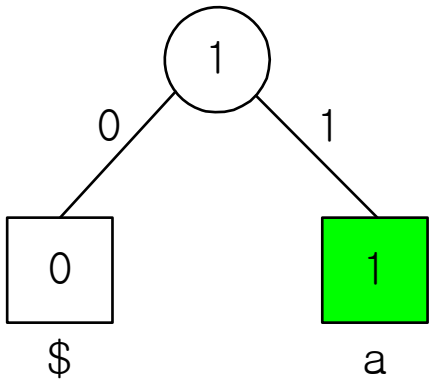
(a) 초기 상태



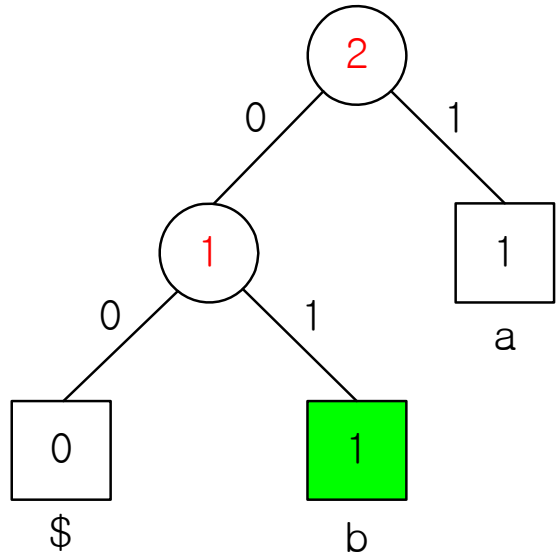
(b) a 입력. 출력 : 00001

동적 인코딩 과정

[예제] 텍스트가 'abcdabedbad'라고 하자. a, b, c, d, e의 원래 코드는 각각 5비트 코드로서 00001, 00010, 00011, 00100, 00101라고 하자. 동적 인코딩 과정?

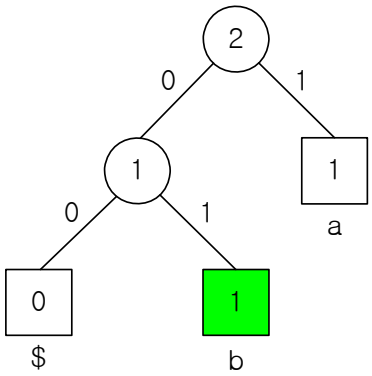


(b) a 입력. 출력 : 00001

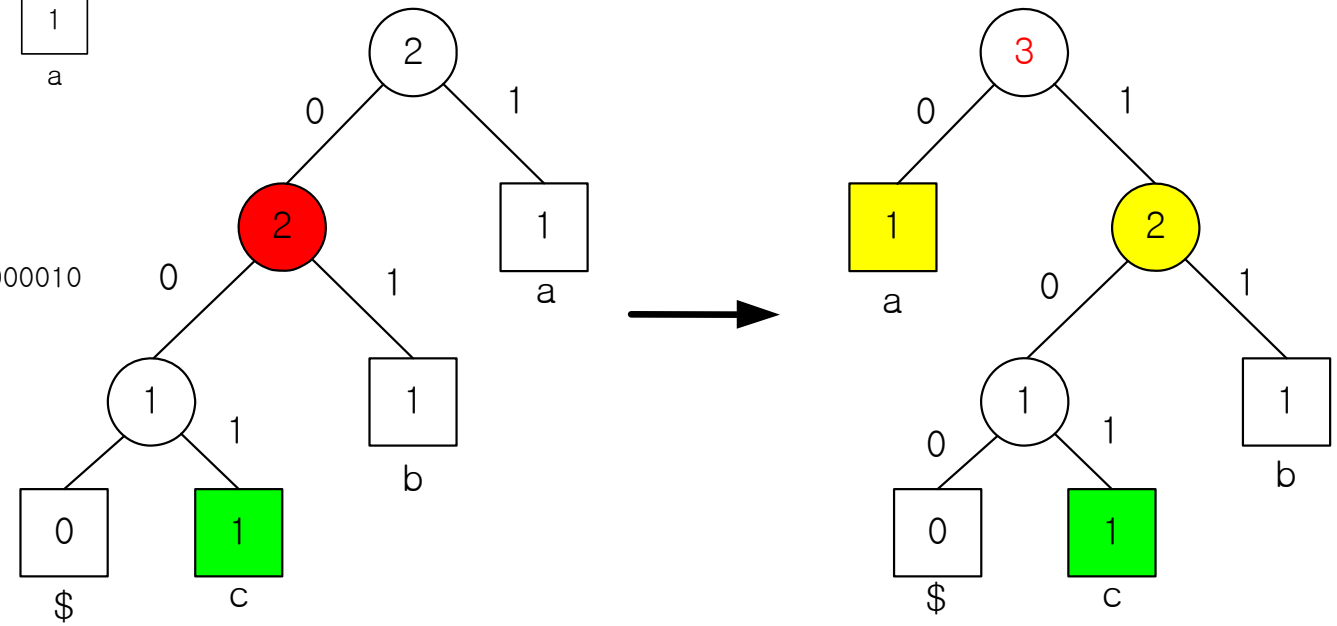


(c) b 입력. 출력 : 000010

[예제] 텍스트가 'abcdabedbad'라고 하자. a, b, c, d, e의 원래 코드는 각각 5비트 코드로서 00001, 00010, 00011, 00100, 00101라고 하자. 동적 인코딩 과정?

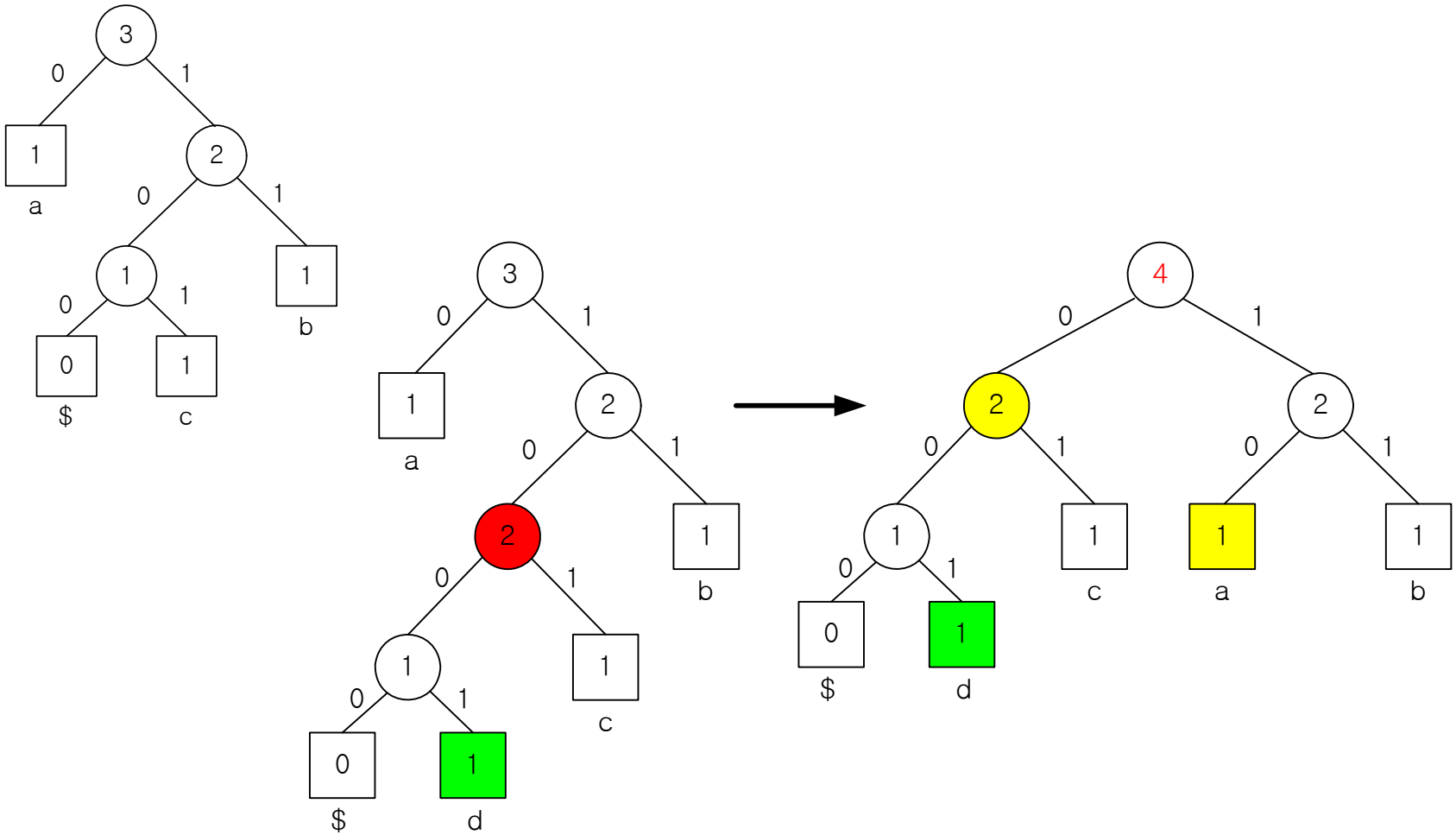


(c) b 입력. 출력 : 000010



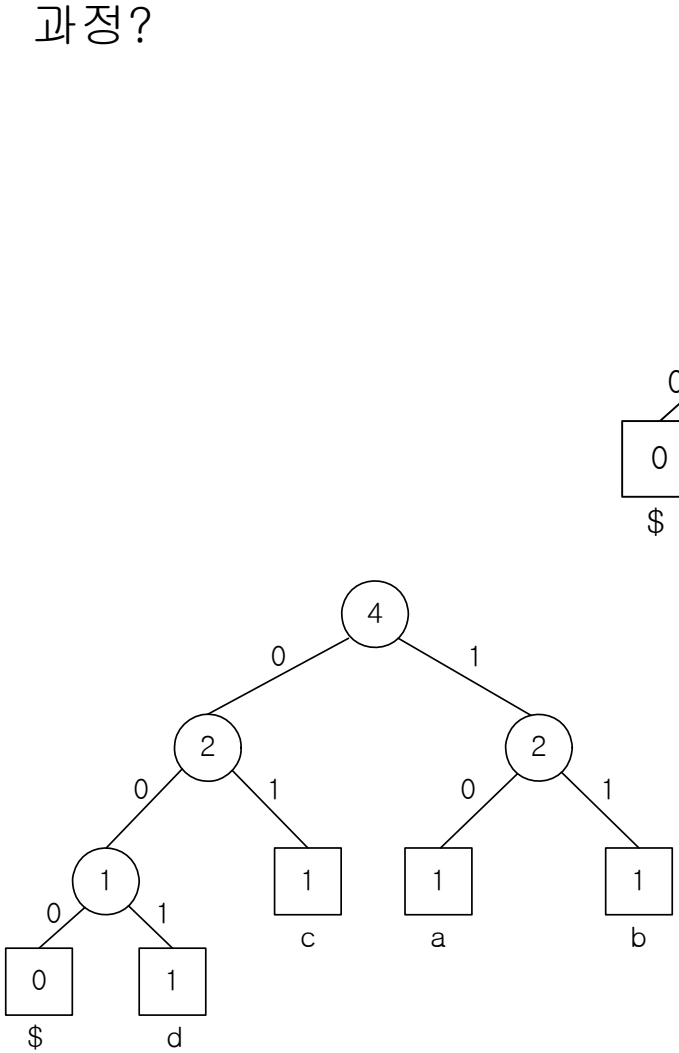
(d) c 입력. 출력 : 0000011

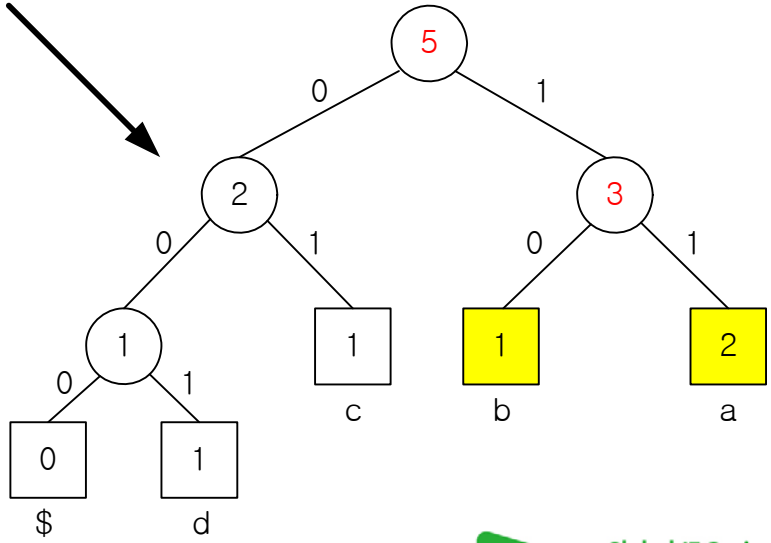
[예제] 텍스트가 'abcdabedbad'라고 하자. a, b, c, d, e의 원래 코드는 각각 5비트 코드로서 00001, 00010, 00011, 00100, 00101라고 하자. 동적 인코딩 과정?



(e) d 입력. 출력 : 10000100

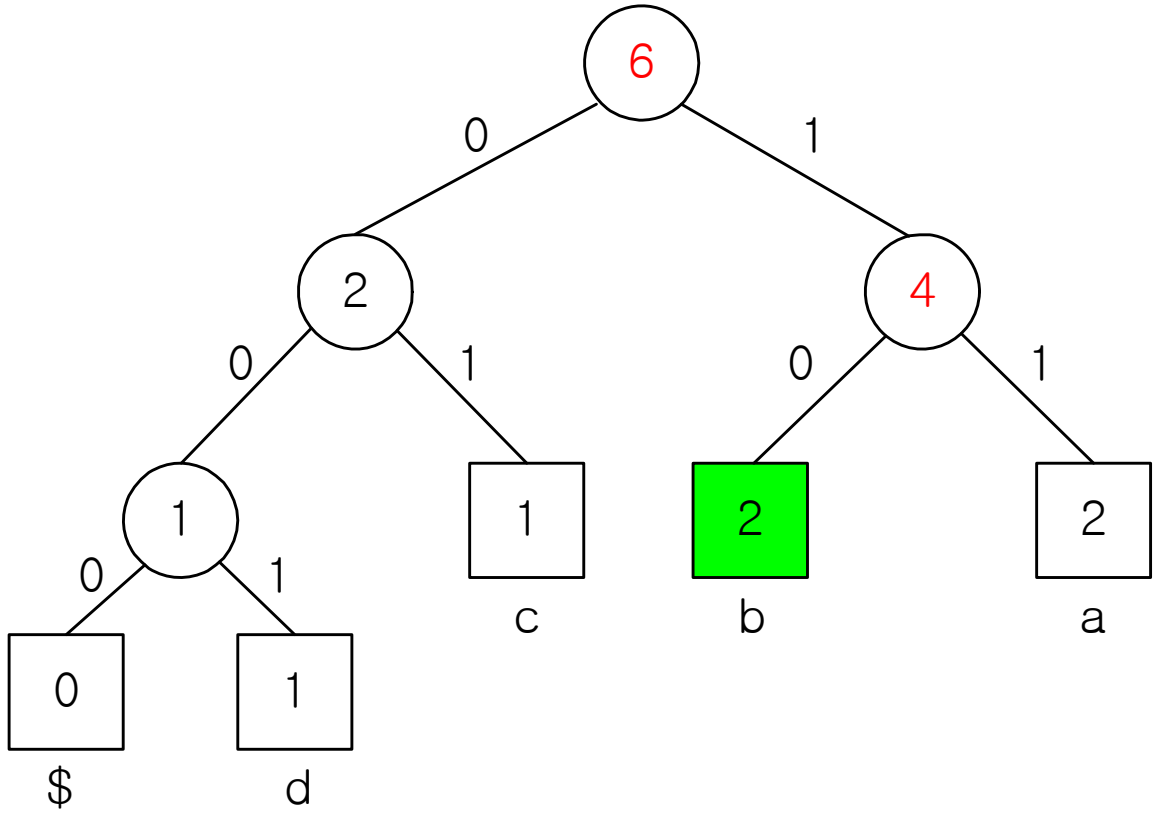
[예제] 텍스트가 'abcdabedbad'라고 하자. a, b, c, d, e의 원래 코드는 각각 5비트 코드로서 00001, 00010, 00011, 00100, 00101라고 하자. 동적 인코딩 과정?





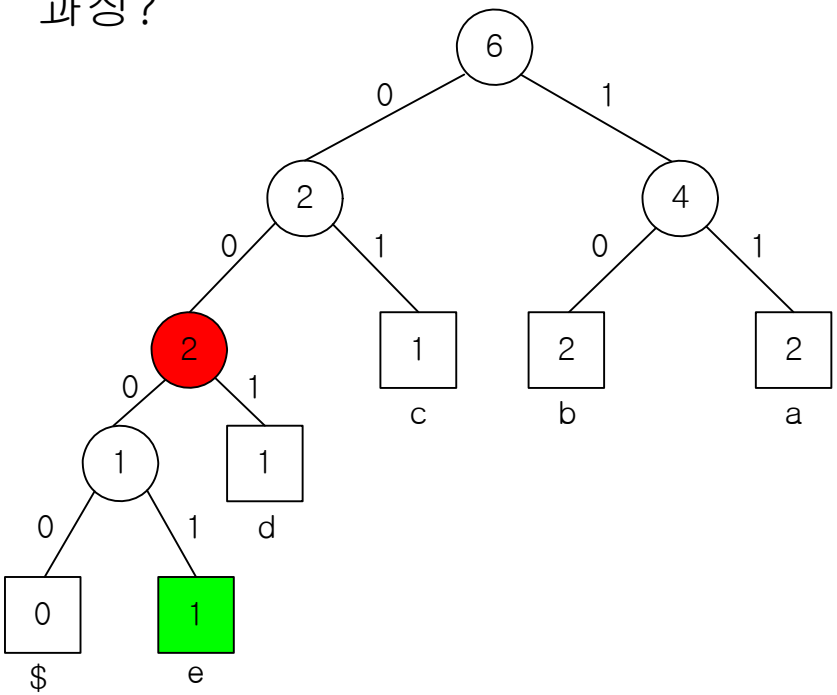
(f) a 입력, 출력 : 10

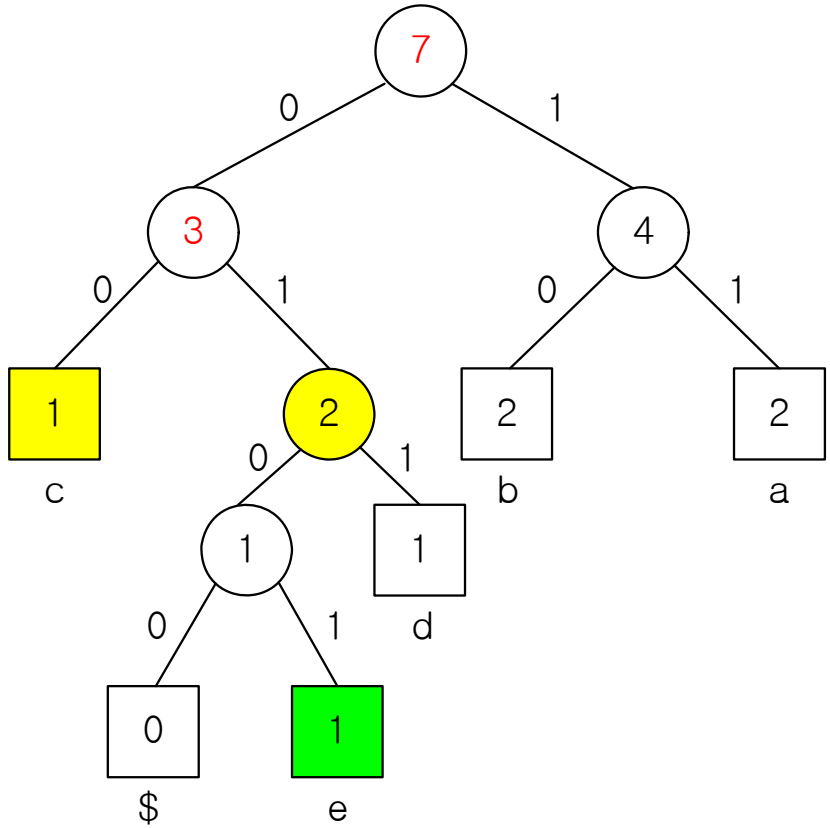
[예제] 텍스트가 'abcdabedbad'라고 하자. a, b, c, d, e의 원래 코드는 각각 5비트 코드로서 00001, 00010, 00011, 00100, 00101라고 하자. 동적 인코딩 과정?



(g) b 입력. 출력 : 10

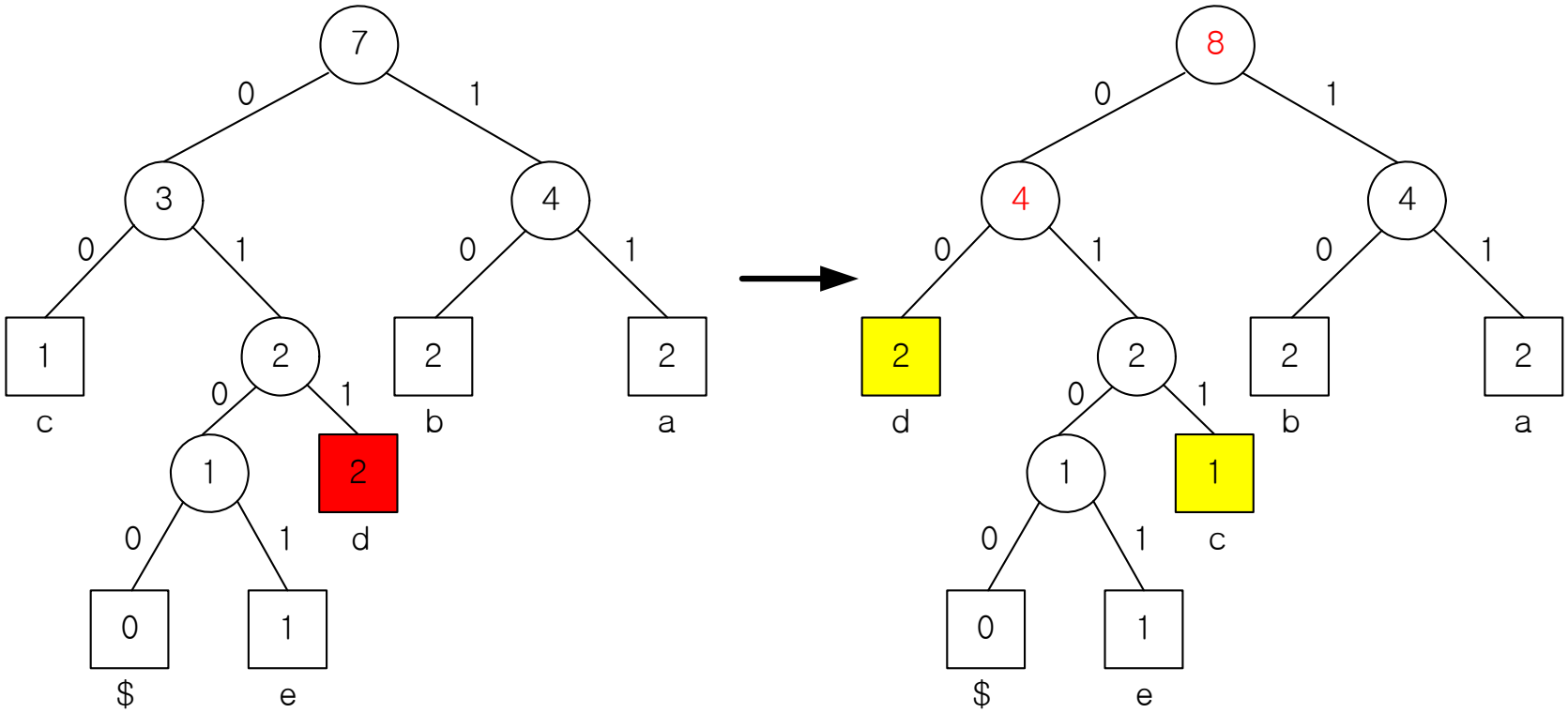
[예제] 텍스트가 'abcdabedbad'라고 하자. a, b, c, d, e의 원래 코드는 각각 5비트 코드로서 00001, 00010, 00011, 00100, 00101라고 하자. 동적 인코딩 과정?





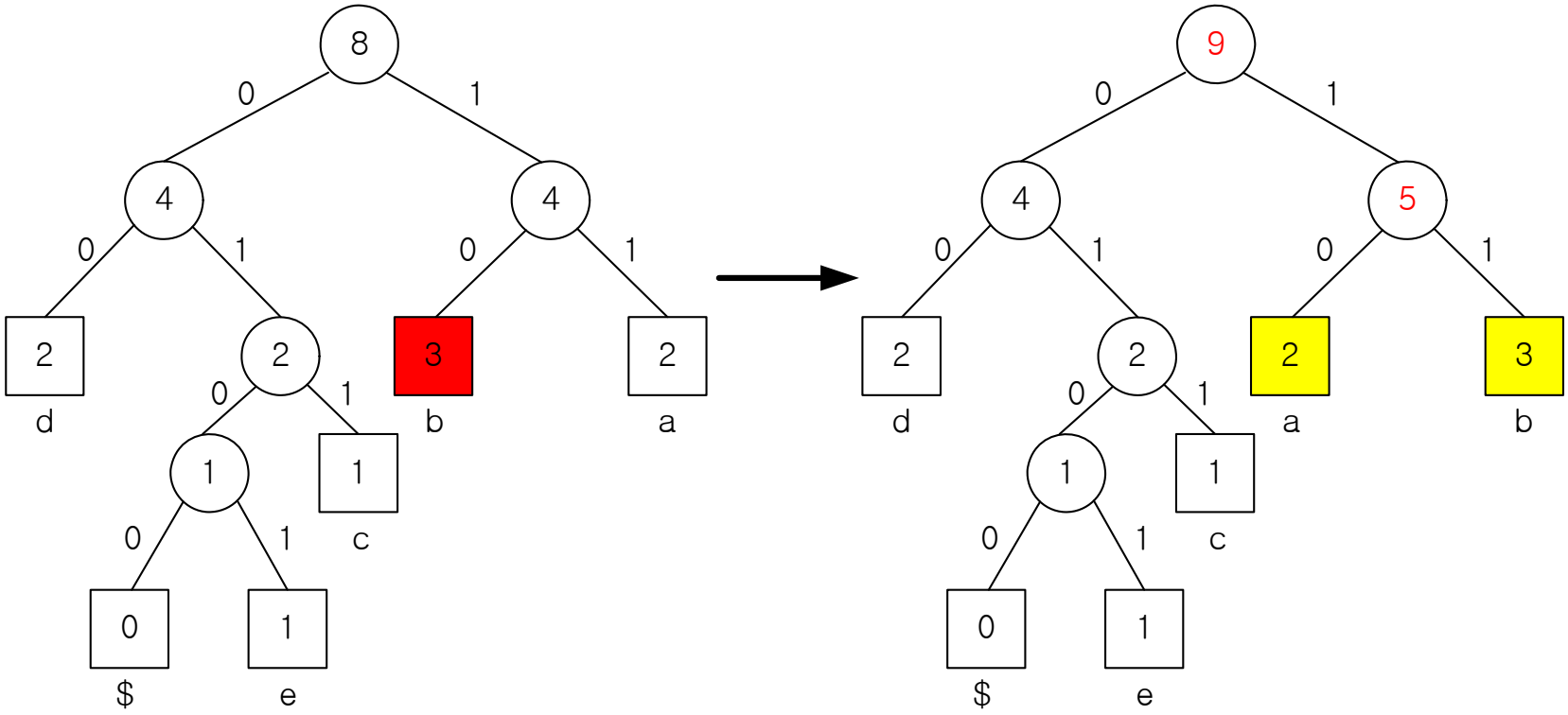
(h) e 삽입. 출력 : 00000101

[예제] 텍스트가 'abcdabedbad'라고 하자. a, b, c, d, e의 원래 코드는 각각 5비트 코드로서 00001, 00010, 00011, 00100, 00101라고 하자. 동적 인코딩 과정?



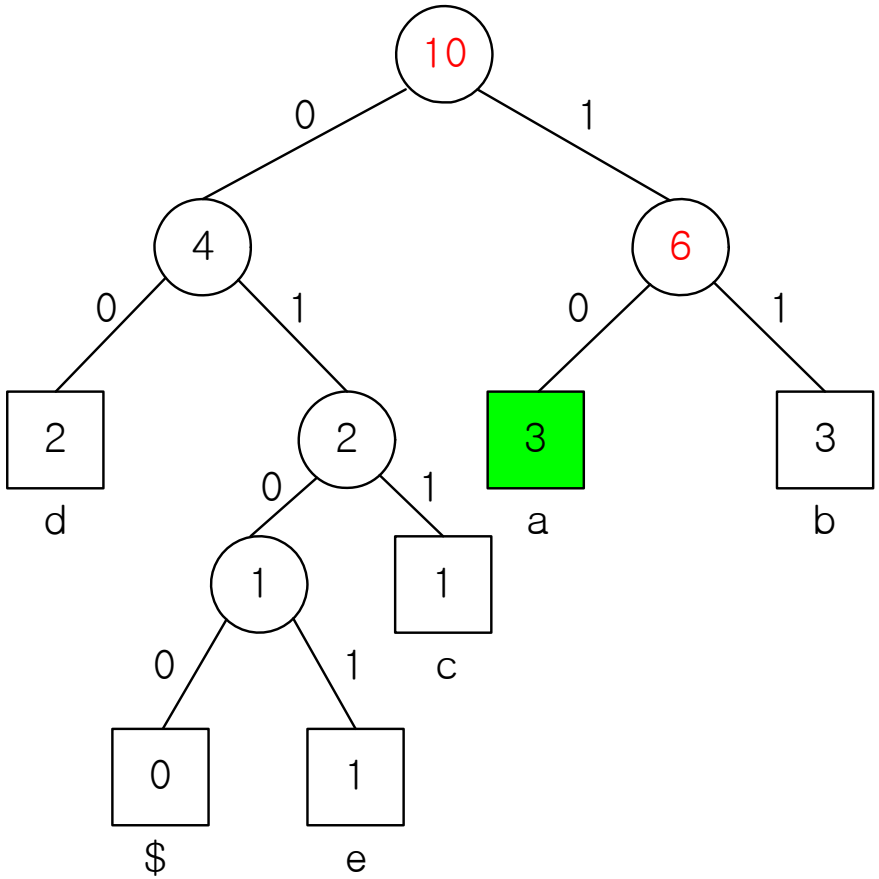
(i) d 입력. 출력 : 011

[예제] 텍스트가 'abcdabedbad'라고 하자. a, b, c, d, e의 원래 코드는 각각 5비트 코드로서 00001, 00010, 00011, 00100, 00101라고 하자. 동적 인코딩 과정?



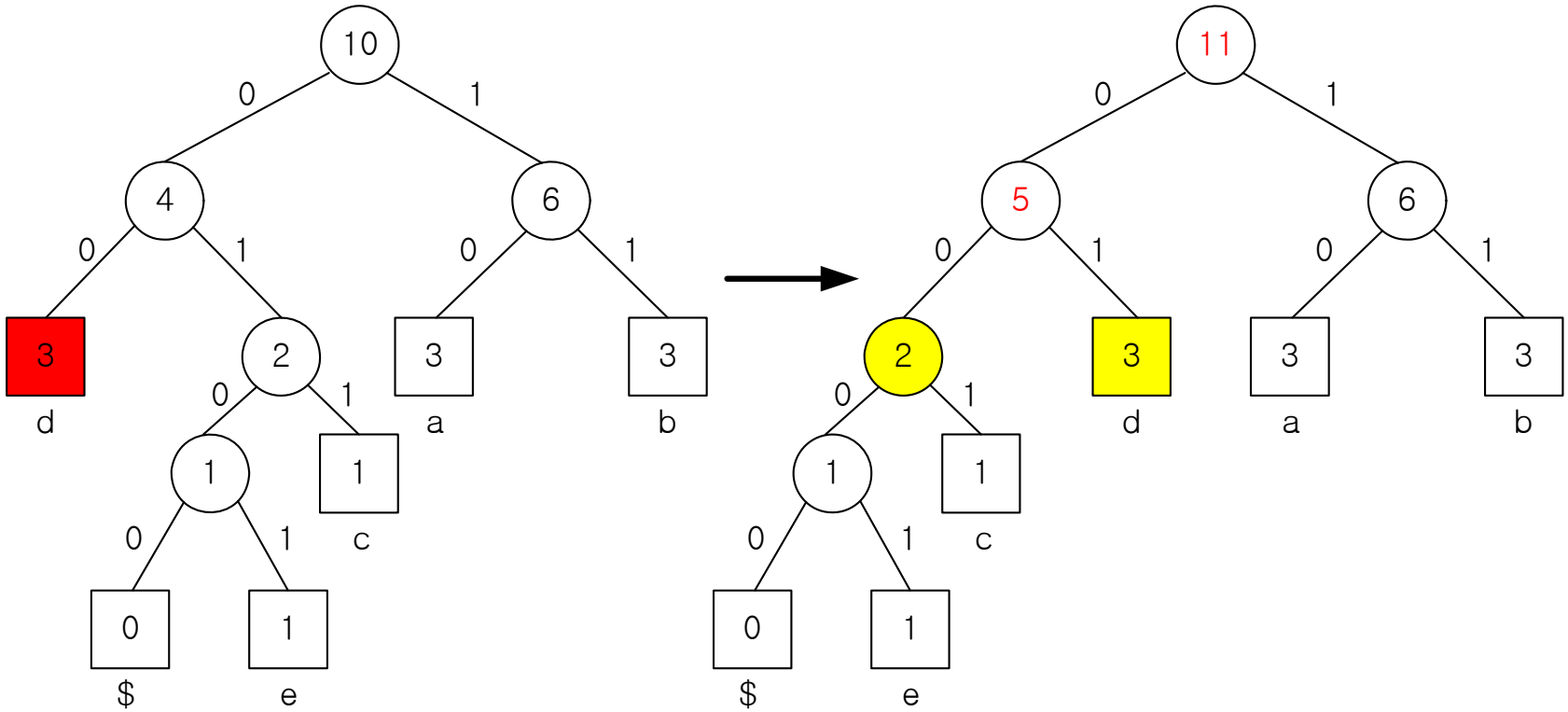
(j) b 입력. 출력 : 10

[예제] 텍스트가 'abcdabedbad'라고 하자. a, b, c, d, e의 원래 코드는 각각 5비트 코드로서 00001, 00010, 00011, 00100, 00101라고 하자. 동적 인코딩 과정?



(k) a 입력. 출력: 10

[예제] 텍스트가 'abcdabedbad'라고 하자. a, b, c, d, e의 원래 코드는 각각 5비트 코드로서 00001, 00010, 00011, 00100, 00101라고 하자. 동적 인코딩 과정?



(I) d 입력. 출력 : 00