Structuri de date și algoritmi

Aprofundare Divite et Impera - Greedy

Lect. Dr. Ing. Cristian Chilipirea – cristian.chilipirea@mta.ro

Subsecvența de sumă maximă

Un șir de numere întregi : $S = \{s_1, s_2, \dots, s_N\}$.

O subsecvență a șirului este de forma: $\{s_i, s_{i+1}, \dots, s_j\}$, $i \leq j$. Să se determine subsecvența de sumă maximă.

Exemplu: N = 7, S={ 5 , -6 , 3 , 4 , -2 , 3 , -3 }.

Subsecvența de sumă maximă

Un șir de numere întregi : $S = \{s_1, s_2, \dots, s_N\}$.

O subsecvență a șirului este de forma: $\{s_i, s_{i+1}, \dots, s_j\}$, $i \leq j$. Să se determine subsecvența de sumă maximă.

Exemplu: N = 7, S={ 5 , -6 , 3 , 4 , -2 , 3 , -3 }.

Subsecvența de sumă maximă este (3 4 -2 3) ⇒ 8.

Subsecvența de sumă maximă – Brute Force

```
int subsecmax(int* v, int n)
{
    int max = -1000, i, j, k;
    for (i = 0; i < n; i++) {      // indexul de început
        for (j = i; j < n; j++) {  // indexul de final
            int sum = 0;
            for (k = i; k <= j; k++)
                sum += v[k];  // suma subsecvenței
            if (sum > max) max = sum;
        }
    }
    return max;
}
```

Complexitate?

$$T(n) = \sum_{i=0}^{n-1} \left(\sum_{j=i}^{n-1} (j - i + 1) \right) =$$

Complexitate?

$$T(n) = \sum_{i=0}^{n-1}\left(\sum_{j=i}^{n-1}(j - i + 1)\right) = \sum_{i=0}^{n-1}[1 + 2 + \cdots (n - i)]$$

Complexitate?

$$T(n) = \sum_{i=0}^{n-1}\left(\sum_{j=i}^{n-1}(j-i+1)\right) = \sum_{i=0}^{n-1}[1+2+\cdots(n-i)]$$

$$= \sum_{i=0}^{n-1}\left[\frac{(n-i)(n-i+1)}{2}\right] = \frac{1}{2}\sum_{i=0}^{n-1}(n-i)^2+(n-i)$$

Complexitate?

$$T(n) = \sum_{i=0}^{n-1}\left(\sum_{j=i}^{n-1}(j-i+1)\right) = \sum_{i=0}^{n-1}[1+2+\cdots(n-i)]$$

$$= \sum_{i=0}^{n-1}\left[\frac{(n-i)(n-i+1)}{2}\right] = \frac{1}{2}\sum_{i=0}^{n-1}(n-i)^2+(n-i)$$

$$= \frac{1}{2}\sum_{k=1}^{n}k^2 + \frac{1}{2}\sum_{k=1}^{n}k$$

Complexitate?

$$T(n) = \sum_{i=0}^{n-1}\left(\sum_{j=i}^{n-1}(j-i+1)\right) = \sum_{i=0}^{n-1}[1+2+\cdots(n-i)]$$

$$= \sum_{i=0}^{n-1}\left[\frac{(n-i)(n-i+1)}{2}\right] = \frac{1}{2}\sum_{i=0}^{n-1}(n-i)^2+(n-i)$$

$$= \frac{1}{2}\sum_{k=1}^{n}k^2 + \frac{1}{2}\sum_{k=1}^{n}k =$$

$$= \boldsymbol{\frac{n(n+1)(2n+1)}{12} + \frac{n(n+1)}{4}}$$

Complexitate?

$$T(n) = \sum_{i=0}^{n-1}\left(\sum_{j=i}^{n-1}(j-i+1)\right) = \sum_{i=0}^{n-1}[1+2+\cdots(n-i)]$$

$$= \sum_{i=0}^{n-1}\left[\frac{(n-i)(n-i+1)}{2}\right] = \frac{1}{2}\sum_{i=0}^{n-1}(n-i)^2+(n-i)$$

$$= \frac{1}{2}\sum_{k=1}^{n}k^2 + \frac{1}{2}\sum_{k=1}^{n}k =$$

$$= \boldsymbol{\frac{n(n+1)(2n+1)}{12} + \frac{n(n+1)}{4} = \theta(n^3)}$$

Complexitate?

$$\theta(n^3)$$

Subsecvența de sumă maximă – Brute Force

```
int subsecmax(int* v, int n)
{
    int max = -1000, i, j, k;
    for (i = 0; i < n; i++) {       // indexul de început
        for (j = i; j < n; j++) {  // indexul de final
            int sum = 0;
            for (k = i; k <= j; k++)
                sum += v[k];  // suma subsecvenței
            if (sum > max) max = sum;
        }
    }
    return max;
}
```

Subsecvența de sumă maximă – Brute Force Clean

```
int subsecmax(int* v, int n)
{
    int max = -1000, i, j;
    for (i = 0; i < n; i++) {  // indexul de început
        int sum = 0;
        for (j = i; j < n; j++) {  // indexul de final
            sum += v[j];
            if (sum > max)
                max = sum;
        }
    }
    return max;
}
```

Complexitate?

Complexitate?

$$\theta(n^2)$$

Subsecvența de sumă maximă – Divide et Impera - Naiv

```
int subsecmax(int* v, int l, int r)
{
    if (l == r)
        return v[l];
    int max1 = subsecmax(v, l, r - 1);
    int max2 = subsecmax(v, l + 1, r);
    int max = max1 > max2 ? max1 : max2;
    int sum = 0, i;
    for (i = l; i <= r; i++) sum += v[i];
    if (sum > max)
        return sum;
    else
        return max;
}
```

Complexitate?

Complexitate?

$$T(n) = 2T(n-1) + \theta(n)$$

Complexitate?

$$T(n) = 2T(n-1) + \theta(n)$$

$$\boldsymbol{T(n) = \theta(2^n)}$$

Subsecvența de sumă maximă – Divide et Impera

5	-6	3	4	-2	3	-3
5	-6	**3**	**4**			
				-2	**3**	-3
5	-6	**3**	**4**	**-2**	**3**	-3

Subsecvența de sumă maximă – Divide et Impera

5	-6	3	4	-2	3	-3
5	-6	**3**	**4**			
				-2	**3**	-3
5	-6	**3**	**4**	**-2**	**3**	-3

suma maximă = max (3+4 , 3, 3+4-2+3)

Subsecvența de sumă maximă – Divide et Impera

```
int subsecmaxd(int* v, int l, int r)
{
    if (l == r) return v[l];
    int m = (l + r) / 2;
    int max1 = subsecmaxd(v, l, m);
    int max2 = subsecmaxd(v, m + 1, r);
    int sum1 = v[m], sumLeft = v[m], i;
    for (i = m - 1; i >= l; i--) {
        sum1 += v[i];
        if (sum1 > sumLeft) sumLeft = sum1;
    }
    int sum2 = v[m + 1], sumRight = v[m + 1];
    for (i = m + 2; i <= r; i++) {
        sum2 += v[i];
        if (sum2 > sumRight) sumRight = sum2;
    }
    int max = max1 > max2 ? max1 : max2;
    if (sumLeft + sumRight < max) return max;
    return sumLeft + sumRight;
}
```

Complexitate?

Complexitate?

$$T(n) = 2T\left(\frac{n}{2}\right) + \theta(n)$$

Metoda Master – calcul complexităţi soluţii recursive

Problema de dimensiune n, divizată în $\boldsymbol{a}$ subprobleme de dimensiune $n/\boldsymbol{b}$. Combinarea subproblemelor şi divizarea problemei se realizează într-un timp $f(n)$. $\boldsymbol{a} \geq 1$; $\boldsymbol{b} > 1$

$$\boldsymbol{T(n) = aT\left(\frac{n}{b}\right) + f(n)}$$

1. $f(n) = O\left(n^{\log_b a - \epsilon}\right), \epsilon > 0 \Rightarrow \boldsymbol{T(n) = \Theta\left(n^{\log_b a}\right)}$
2. $f(n) = \Theta\left(n^{\log_b a}\right) \Rightarrow \boldsymbol{T(n) = \Theta\left(n^{\log_b a} \lg n\right)}$
3. $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right), \epsilon > 0,\ af\left(\frac{n}{b}\right) \leq cf(n), c < 1 \Rightarrow \boldsymbol{T(n) = \Theta(f(n))}$

Complexitate?

$$T(n) = 2T\left(\frac{n}{2}\right) + \theta(n)$$

$$\boldsymbol{T(n) = aT\left(\frac{n}{b}\right) + f(n)}$$

$$\boldsymbol{a = 2 \qquad b = 2 \qquad f(n) = \theta(n)}$$

1. $f(n) = O\left(n^{log_b a-\epsilon}\right), \epsilon > 0 \Rightarrow \boldsymbol{T(n) = \Theta\left(n^{log_b a}\right)}$
2. $f(n) = \Theta\left(n^{log_b a}\right) \Rightarrow \boldsymbol{T(n) = \Theta\left(n^{log_b a} lgn\right)}$
3. $f(n) = \Omega\left(n^{log_b a+\epsilon}\right), \epsilon > 0,\ af\left(\frac{n}{b}\right) \leq cf(n), c < 1 \Rightarrow \boldsymbol{T(n) = \Theta(f(n))}$

Complexitate?

$$T(n) = 2T\left(\frac{n}{2}\right) + \theta(n)$$

$$\boldsymbol{T(n) = aT\left(\frac{n}{b}\right) + f(n)}$$

$$\boldsymbol{a = 2} \quad \boldsymbol{b = 2} \quad \boldsymbol{f(n)} = \theta(n)$$

1. $\theta(n) = O\left(n^{\log_2 2 - \epsilon}\right), \epsilon > 0 \Rightarrow \boldsymbol{T(n) = \Theta\left(n^{\log_b a}\right)}$
2. $\theta(n) = \Theta\left(n^{\log_2 2}\right) \Rightarrow \boldsymbol{T(n) = \Theta\left(n^{\log_b a} \lg n\right)}$
3. $\theta(n) = \Omega\left(n^{\log_2 2 + \epsilon}\right), \epsilon > 0,\ af\left(\frac{n}{b}\right) \leq cf(n), c < 1 \Rightarrow \boldsymbol{T(n) = \Theta(f(n))}$

Complexitate?

$$T(n) = 2T\left(\frac{n}{2}\right) + \theta(n)$$

$$\boldsymbol{T(n) = aT\left(\frac{n}{b}\right) + f(n)}$$

$$\boldsymbol{a = 2 \quad b = 2 \quad f(n) = \theta(n)}$$

1. $\theta(n) = O(n^{1-\epsilon}), \epsilon > 0 \Rightarrow \boldsymbol{T(n) = \Theta\left(n^{\log_b a}\right)}$
2. $\theta(n) = \Theta(n^1) \Rightarrow \boldsymbol{T(n) = \Theta\left(n^{\log_b a} \lg n\right)}$
3. $\theta(n) = \Omega(n^{1+\epsilon}), \epsilon > 0,\ af\left(\frac{n}{b}\right) \le cf(n), c < 1 \ \Rightarrow \boldsymbol{T(n) = \Theta(f(n))}$

Complexitate?

$$T(n) = 2T\left(\frac{n}{2}\right) + \theta(n)$$

$$\boldsymbol{T(n) = aT\left(\frac{n}{b}\right) + f(n)}$$

$$\boldsymbol{a = 2} \quad \boldsymbol{b = 2} \quad \boldsymbol{f(n)} = \theta(n)$$

$2.\theta(n) = \Theta(n^1) \Rightarrow \boldsymbol{T(n)} = \boldsymbol{\Theta\left(n^{log_b a} lgn\right)}$

Complexitate?

$$T(n) = 2T\left(\frac{n}{2}\right) + \theta(n)$$

$$\boldsymbol{T(n) = aT\left(\frac{n}{b}\right) + f(n)}$$

$$\boldsymbol{a = 2} \qquad \boldsymbol{b = 2} \qquad \boldsymbol{f(n)} = \theta(n)$$

$$2.\theta(n) = \Theta(n^1) \Rightarrow \boldsymbol{T(n)} = \boldsymbol{\Theta\left(n^1 lgn\right)}$$

Complexitate?

$$\Theta(n \lg n)$$

Subsecvența de sumă maximă – Algoritmul Kadane

Presupunând că știm care este suma maximă care se încheie la poziția $i \Rightarrow$ care este suma maximă care se încheie la poziția $i + 1$?

			v_i	v_{i+1}					
			v_i	v_{i+1}					
				v_{i+1}					

- Fie prelungim suma maximă care se încheie la i
- Fie luăm doar valoarea de la indexul $i + 1$

Maximul acestor sume = suma maximă căutată

Subsecvența de sumă maximă – Algoritmul Kadane

5	-6	3	4	-2	3	-3	Suma
5							**5**
5	-6						**-1**
5	-6	3					**3**
5	-6	3	4				**7**
5	-6	3	4	-2			**5**
5	-6	3	4	-2	3		**8**
5	-6	3	4	-2	3	-3	**5**

Subsecvența de sumă maximă – Algoritmul Kadane

```
int subsecmax(int* v, int n)
{
    int sumArray = v[0];     // suma maximă a unui subșir
    int sumMaxIndex = v[0];  // suma maximă pt. indexul i
    int i;
    for (i = 1; i < n; i++) {
        if (sumMaxIndex > 0)
            sumMaxIndex += v[i];
        else
            sumMaxIndex = v[i];
        if (sumMaxIndex > sumArray)
            sumArray = sumMaxIndex;
    }
    return sumArray;
}
```

Complexitate?

Complexitate?

$$\theta(n)$$

Fracţii egiptene

Scrierea unei fracţii ca sumă de inverse de întregi :

$$\frac{87}{110} = \frac{1}{2} + \frac{1}{5} + \frac{1}{11}$$

Fracţii egiptene

Scrierea unei fracţii ca sumă de inverse de întregi :

$$\frac{87}{110} = \frac{1}{2} + \frac{1}{5} + \frac{1}{11}$$

Idee: încercarea celui mai mic număr natural posibil

$$\frac{87}{110} = \frac{\mathbf{1}}{\mathbf{2}} + \frac{a}{b} \Rightarrow \frac{a}{b} = \frac{87 * 2 - 110}{110 * 2} = \frac{32}{110} = \frac{16}{55}$$

Fracții egiptene

Scrierea unei fracţii ca sumă de inverse de întregi :

$$\frac{87}{110} = \frac{1}{2} + \frac{1}{5} + \frac{1}{11}$$

Idee: încercarea celui mai mic număr natural posibil

$$\frac{87}{110} = \frac{\mathbf{1}}{\mathbf{2}} + \frac{a}{b} \Rightarrow \frac{a}{b} = \frac{87 * 2 - 110}{110 * 2} = \frac{32}{110} = \frac{16}{55}$$

$$\frac{16}{55} = \frac{\mathbf{1}}{\mathbf{4}} + \frac{a}{b} \Rightarrow \frac{a}{b} = \frac{9}{220} ; \frac{9}{220} = \frac{\mathbf{1}}{\mathbf{25}} + \frac{\mathbf{1}}{\mathbf{1100}}$$

Fracții egiptene - Greedy

```
void printEgyptian(int nr, int dr)
{
    while (dr != 0 && nr != 0) {
        if (dr % nr == 0) {
            printf("1/%d", dr / nr);
            return;
        }
        int n = dr / nr + 1;
        printf("1/%d +", n);
        nr = nr * n - dr;
        dr = dr * n;
    }
}
```

Complexitate?

Codare Huffman

- Folosit în compresii (jpeg, mp3).
- În loc de a folosi valori cu un număr fix de biți folosim un număr de biți variabil.
  - Vrem să folosim puțini biți pentru valori care apar des (frecvență mare)
  - Vrem să folosim mulți biți pentru valori care apar rar (frecvență mică)
- Pentru a putea avea număr variabil de biți trebuie ca nici o valoare să nu fie prefix pentru alta.

Codare Huffman

ana are mere

Codare Huffman

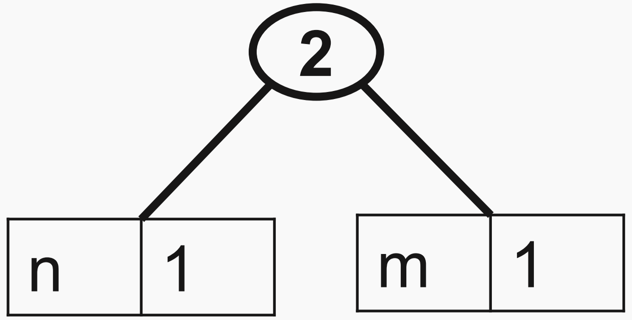
ana are mere

a	3
e	3
r	2
_	2
n	1
m	1

Codare Huffman

ana are mere

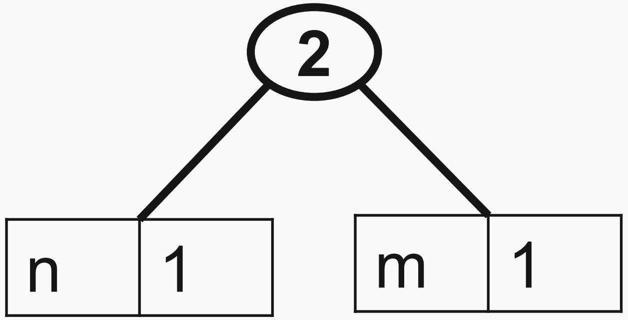
a	3
e	3
r	2
_	2
n	1
m	1



Codare Huffman

ana are mere

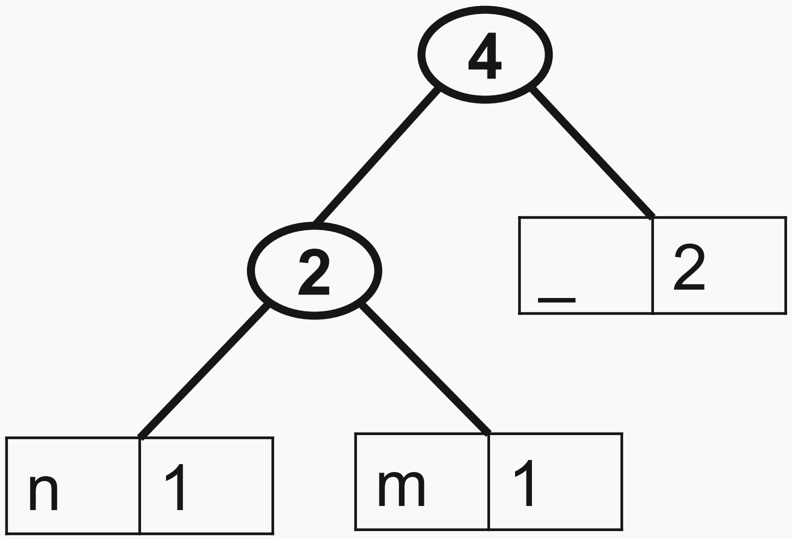
a	3
e	3
r	2
_	2
nm	2



Codare Huffman

ana are mere

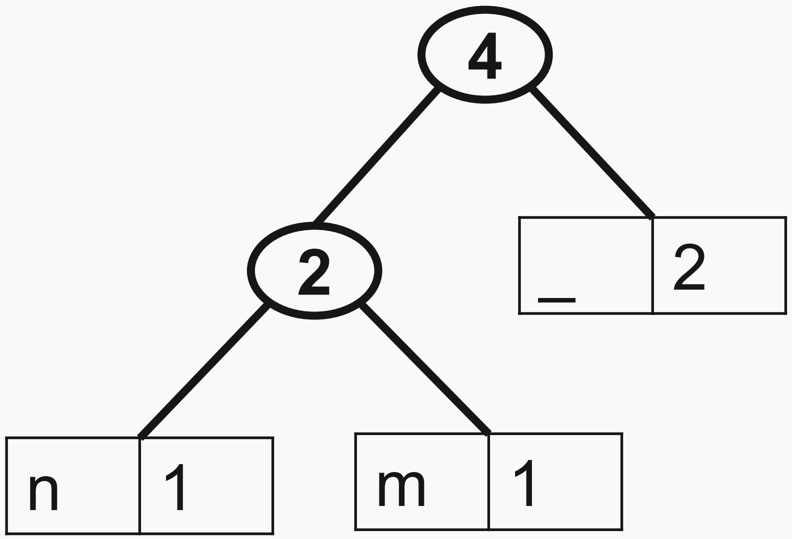
a	3
e	3
r	2
_	2
nm	2



Codare Huffman

ana are mere

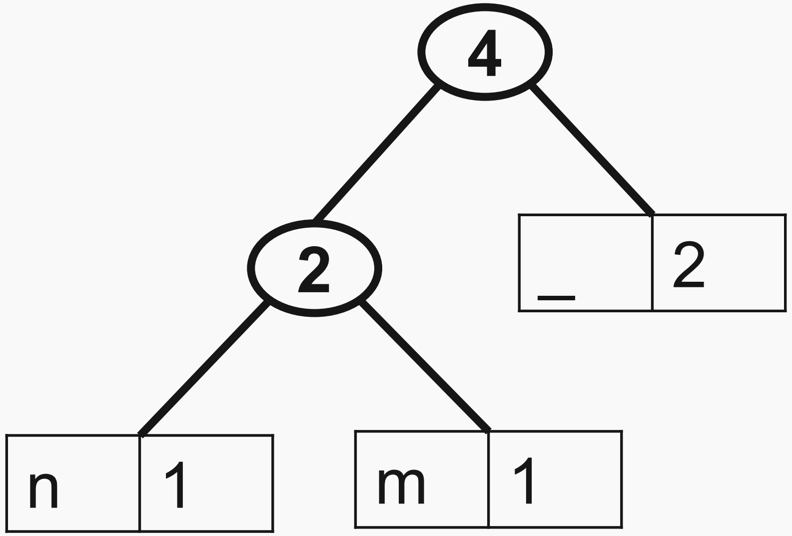
nm_	4
a	3
e	3
r	2

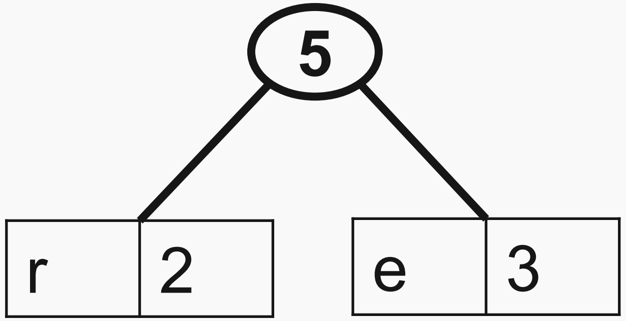


Codare Huffman

ana are mere

nm_	4
a	3
e	3
r	2

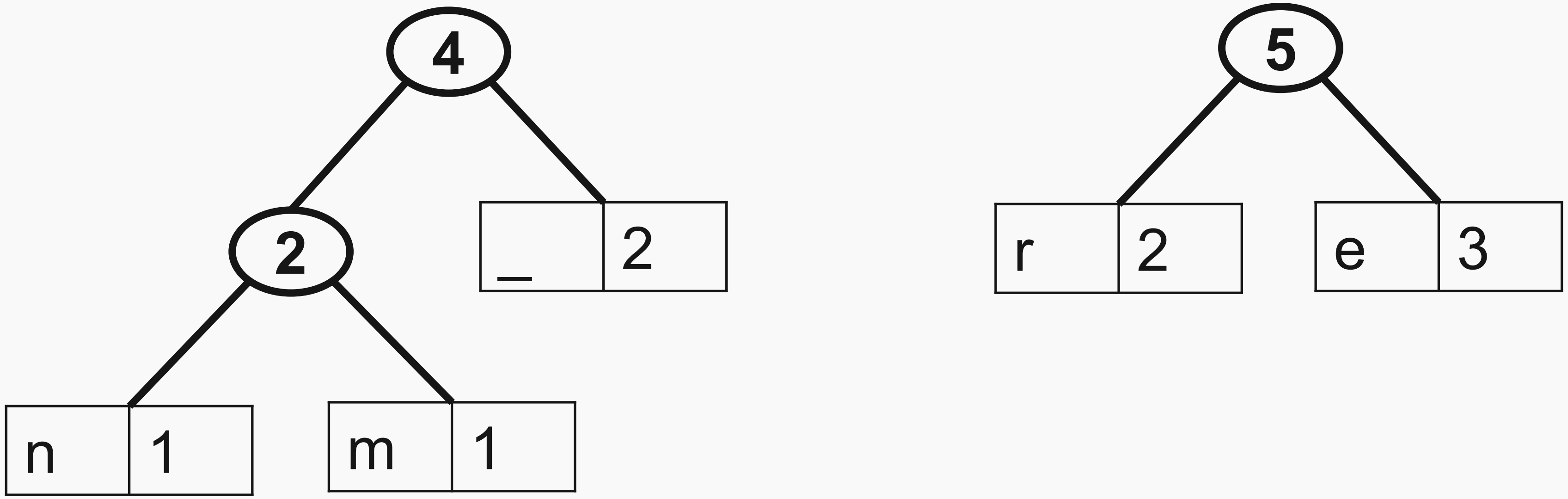




Codare Huffman

ana are mere

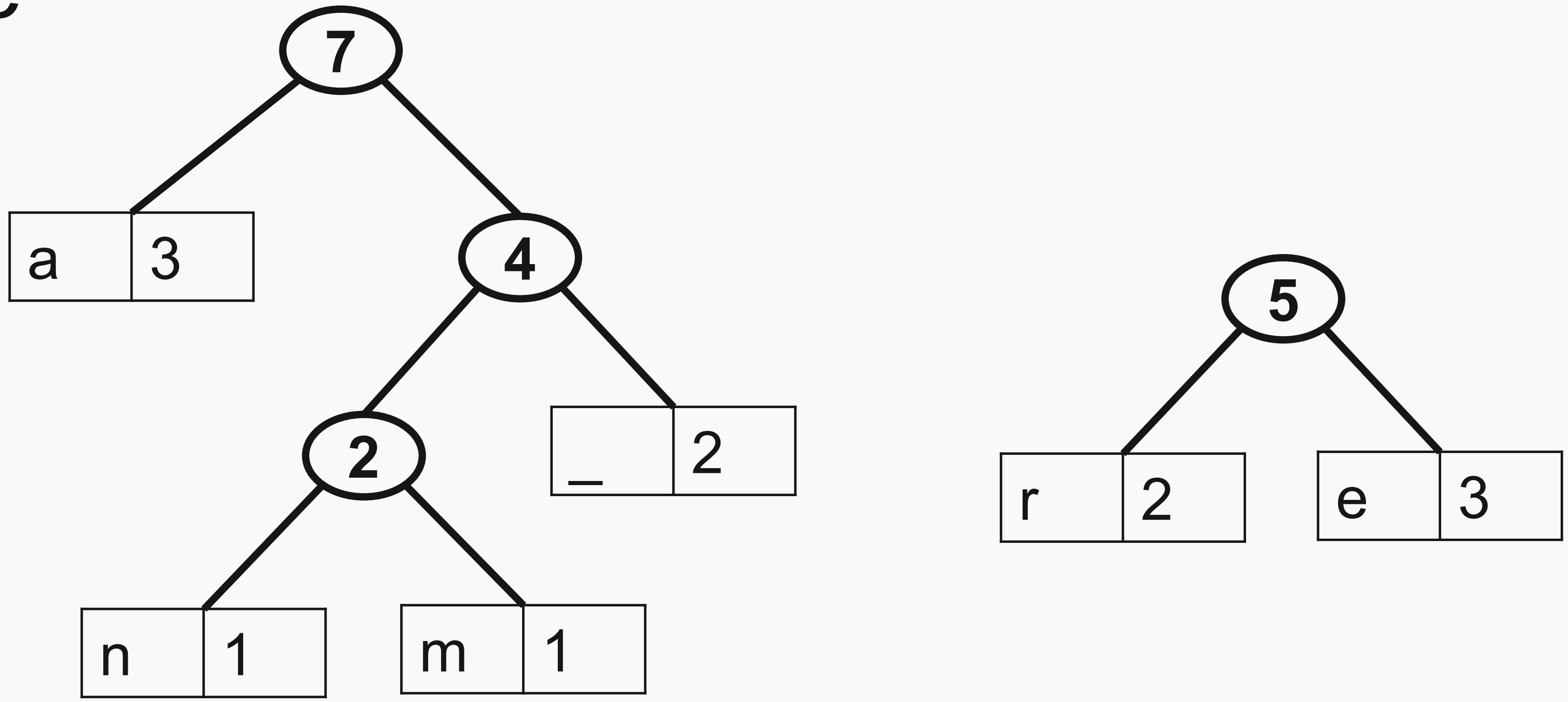
er	5
nm_	4
a	3



Codare Huffman

ana are mere

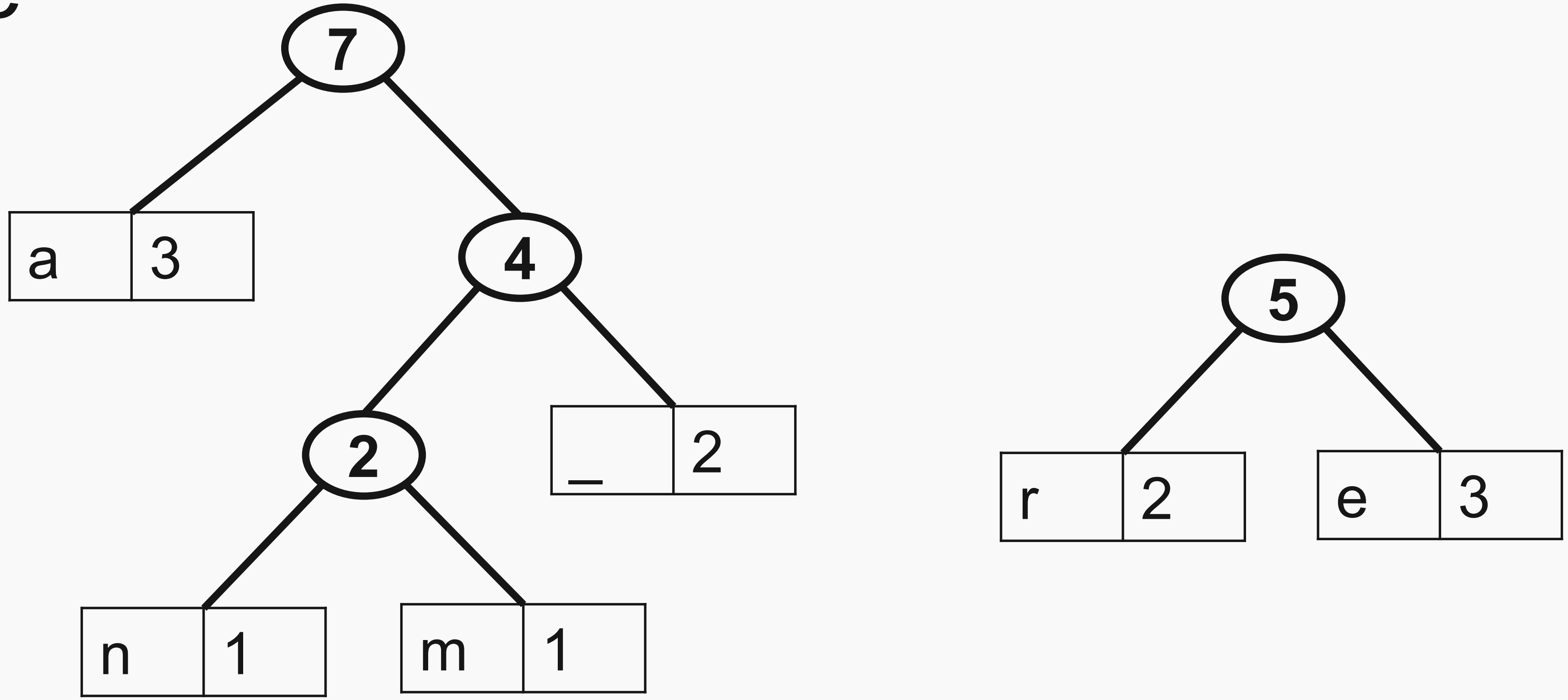
er	5
nm_	4
a	3



Codare Huffman

ana are mere

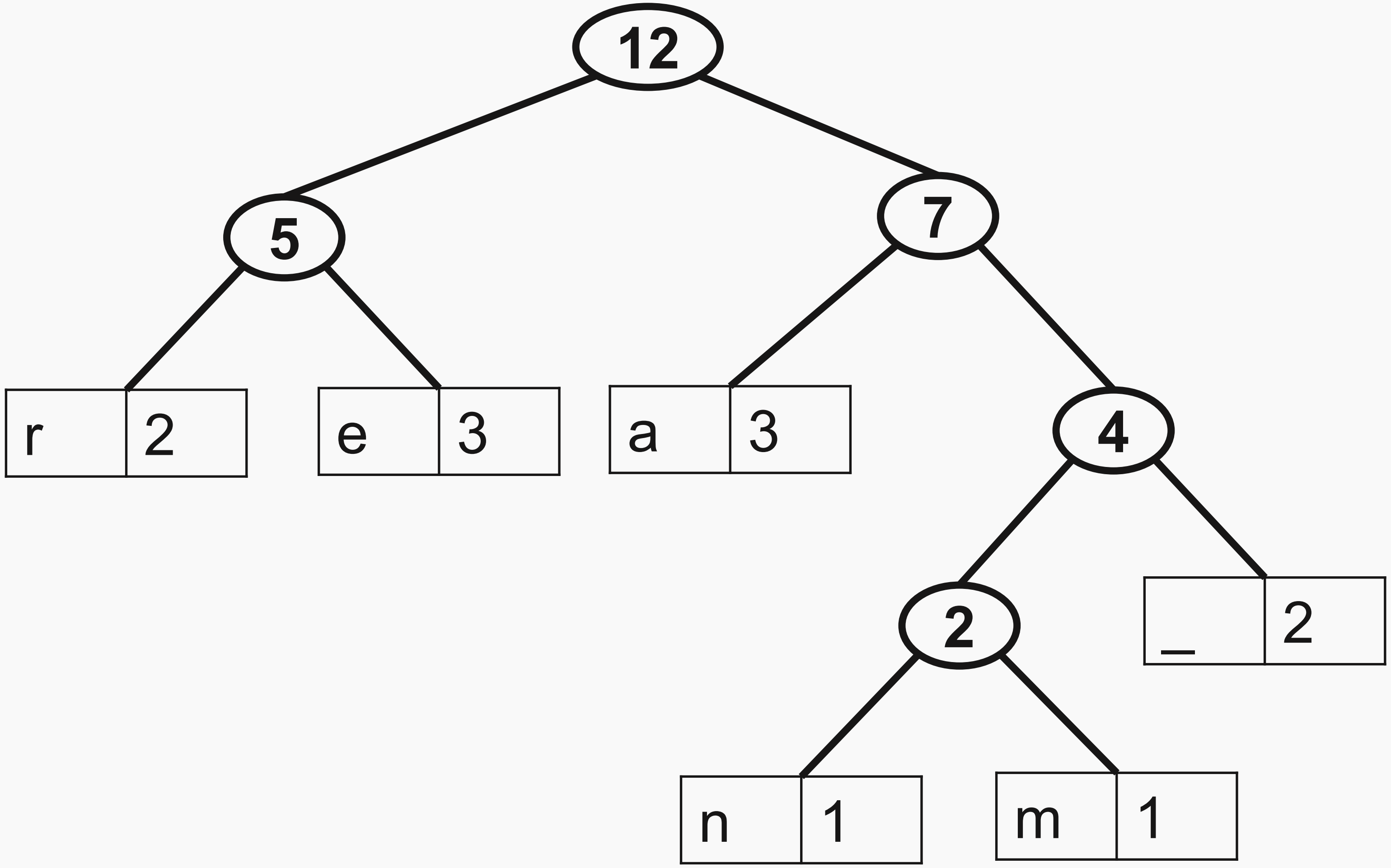
anm_	7
er	5



Codare Huffman

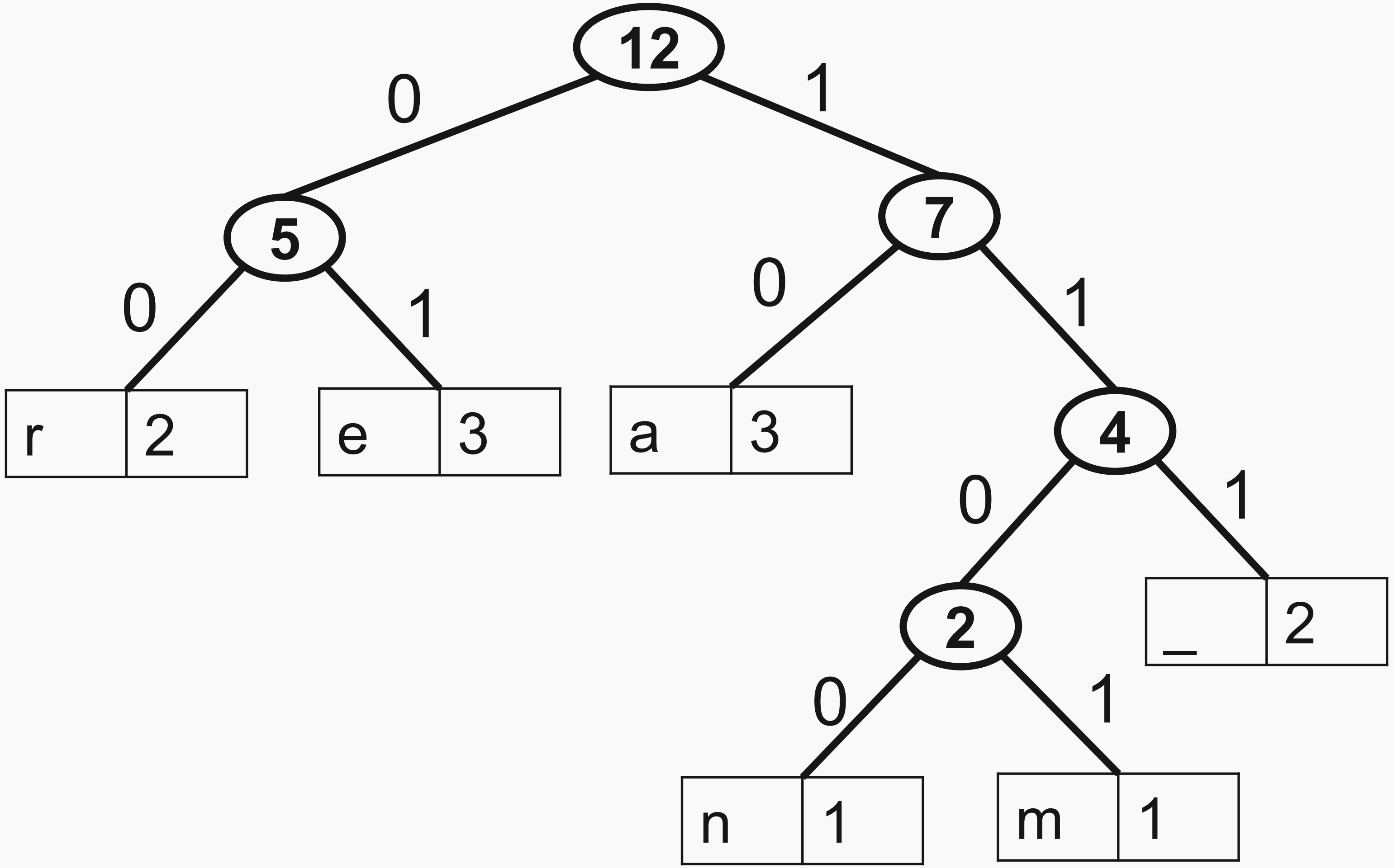
ana are mere

anm_	7
er	5



Codare Huffman

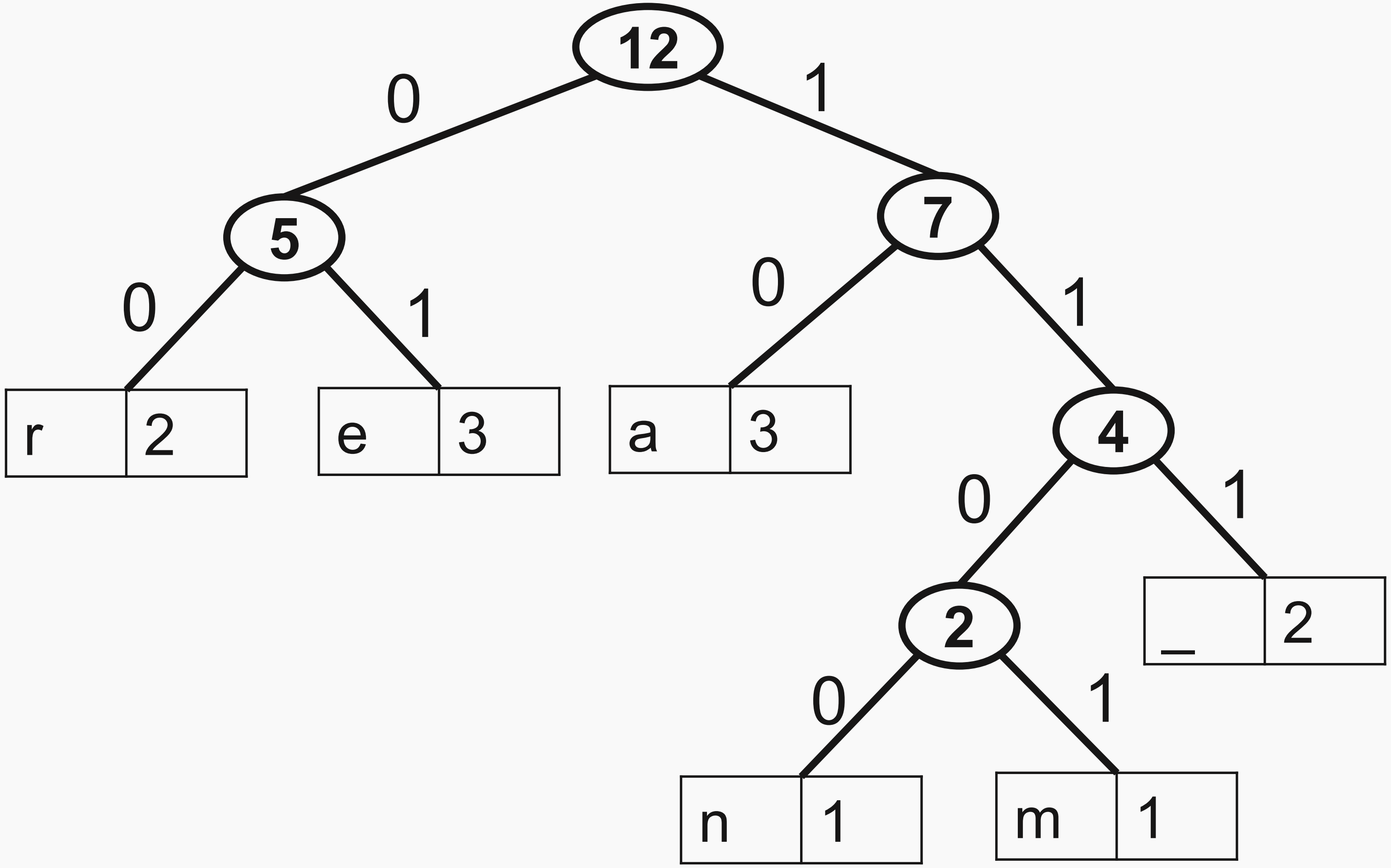
ana are mere



Codare Huffman

ana are mere

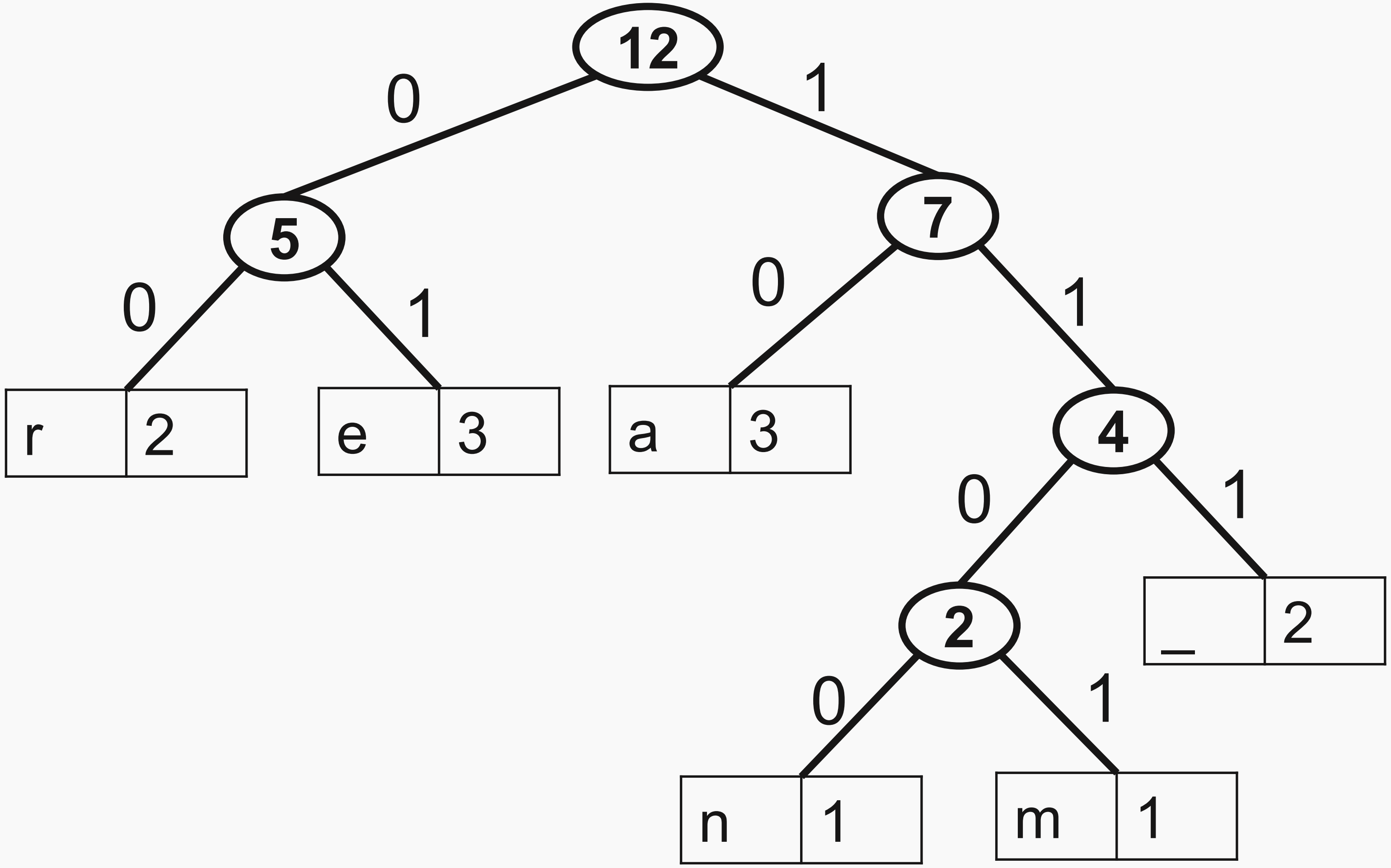
a	10
e	01
r	00
_	111
n	1100
m	1101



Codare Huffman

ana are mere

a	10
e	01
r	00
_	111
n	1100
m	1101



101100101111000011111101010001