

Codul Huffman - Arborele de codificare Huffman

Algoritmul Huffman este folosit pentru compresia datelor fără pierdere de informație. Presupunând că se codifică un text alcătuit din caractere ascii, algoritmul Huffman utilizează un tablou de frecvențe, în care se specifică pentru fiecare caracter o valoare care reprezintă frecvența de apariție a acestuia. Lungimea secvenței de codificare (în biți) pentru fiecare caracter depinde de frecvența acestuia. Cu cât un caracter apare mai frecvent în text, cu atât lungimea codului utilizată pentru el este mai mică.

Coduri prefix (prefix-codes)

Un cod prefix este un cod în care nici un element al codului nu reprezintă un prefix pentru al element. Acest lucru permite codificare/decodificare fără ambiguități.

Exemplu:

Se consideră alfabetul alcătuit din primele 6 litere {a, b, c, d, e, f} și următorul tabel de codificare:

	a	b	c	d	e	f
Frecvența (la 1000 caractere în lb. engl.)	45	13	12	16	9	5
Codificare cu cod de lungime fixă (3 biți) – cod1	000	001	010	011	100	101
Codificare cu cod de lungime variabilă – cod2	0	101	100	111	1101	1100

Codificarea cu codul 2 00101100110110100 reprezintă 0. 0. 101. 100. 1101. 101.0.0 adică textul aabcebaa

Arborele Huffman

Un mod de construcție a unui astfel de cod prefix de lungime variabilă se realizează cu ajutorul arborelui Huffman. În acest arbore, frunzele reprezintă caracterele de codificat. Arcul de la un nod părinte la descendentul stâng se marchează cu caracterul 0, iar cel către descendentul drept se marchează cu 1.

Codul corespunzător unui caracter dintr-o frunză se obține prin concatenarea caracterelor care se află pe drumul de la rădăcină către frunza respectivă.

Observație: un cod optimal se obține pe baza unui arbore binar în care fiecare nod neterminal are exact doi descendenți.

Construcția arborelui de codificare Huffman:

Se plasează fiecare caracter într-o frunză și se plasează într-o coadă de prioritate (min) de noduri. Deci în vârf va fi nodul de prioritate minimă.

Cât timp în coadă se găsesc cel puțin 2 noduri: se extrag primele două minime, se leagă ca fii de către un nou nod părinte, a cărui frecvență va fi dată de suma frecvențelor celor 2 copii. Acest nod nou se inserează în coada de prioritate.

După obținerea arborelui Huffman se poate construi un look-up table în care fiecărui caracter i se asociază codul obținut pe baza arborelui. Acest tabel se folosește apoi pentru codificare.

Decodificarea: pentru a putea decodifica corect documentul, este necesar ca atât cel care codifică, cât și cel care decodifică să utilizeze același arbore. Acest lucru se realizează ori prin folosirea de comun acord a aceluiași set de caractere cu aceeași frecvență sau prin transmiterea împreună cu textul codificat și a arborelui de codificare. În literatură sunt propuse diferite moduri de soluționare a acestei probleme.

Exemplu:

Considerăm următorul text: acesta este un test interesant.

Avem următorul tablou de frecvențe:

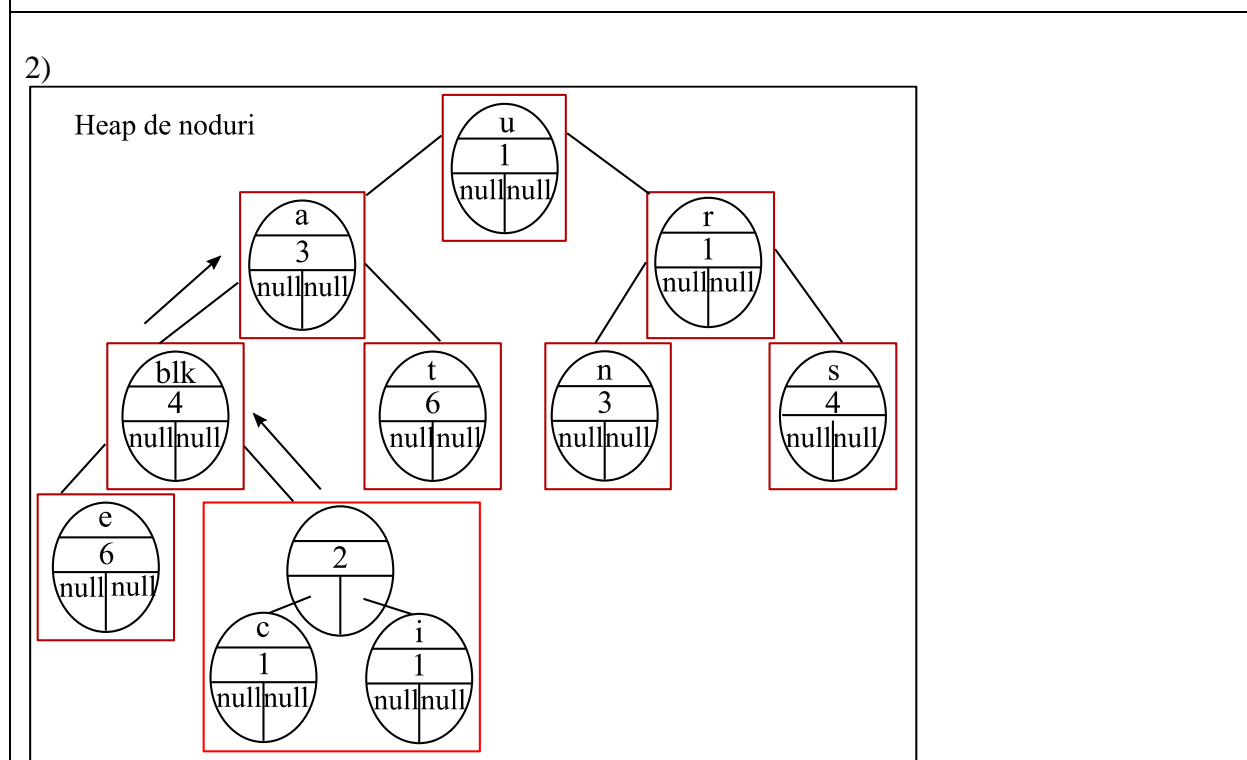
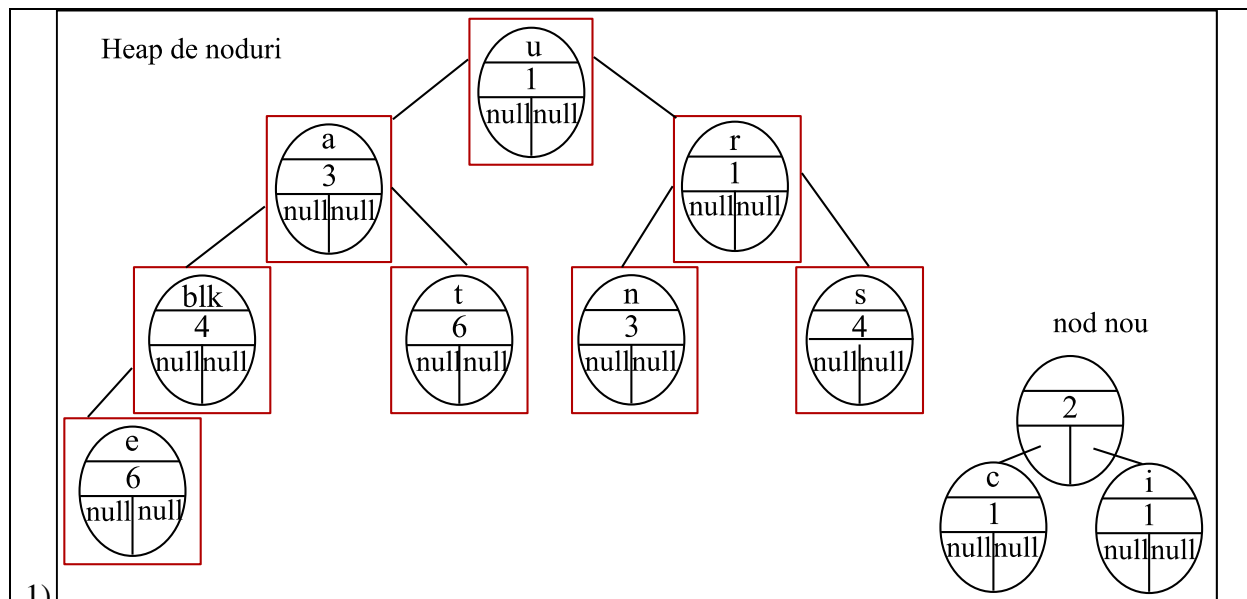
a	c	e	i	n	R	s	t	u	Blank
3	1	6	1	3	1	4	6	1	4

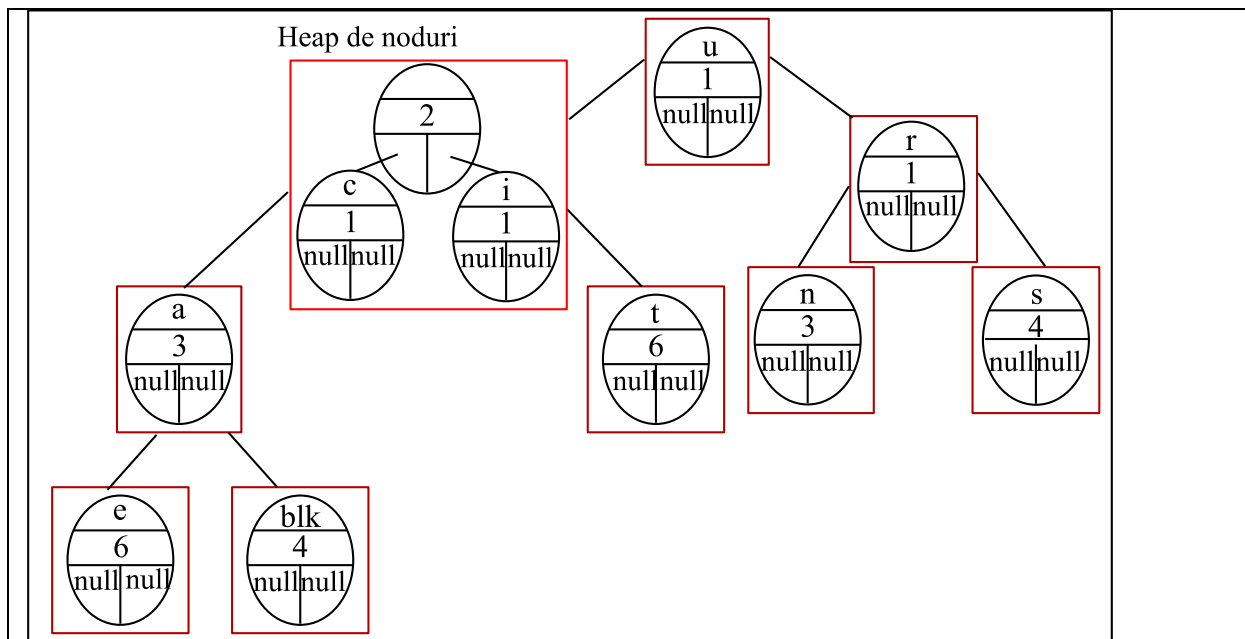
Construim un arbore în care fiecare nod are: un câmp frecvență, un câmp caracter și câmpuri de legătură către fii stâng și drept și către părinte (eventual). Dacă algoritmul de parcurgere al arborelui pentru determinarea codurilor este recursiv, nu este nevoie de legătura la părinte.

Pas 1: se inserează într-o coadă de priorități (min) nodurile corespunzătoare fiecărui caracter:

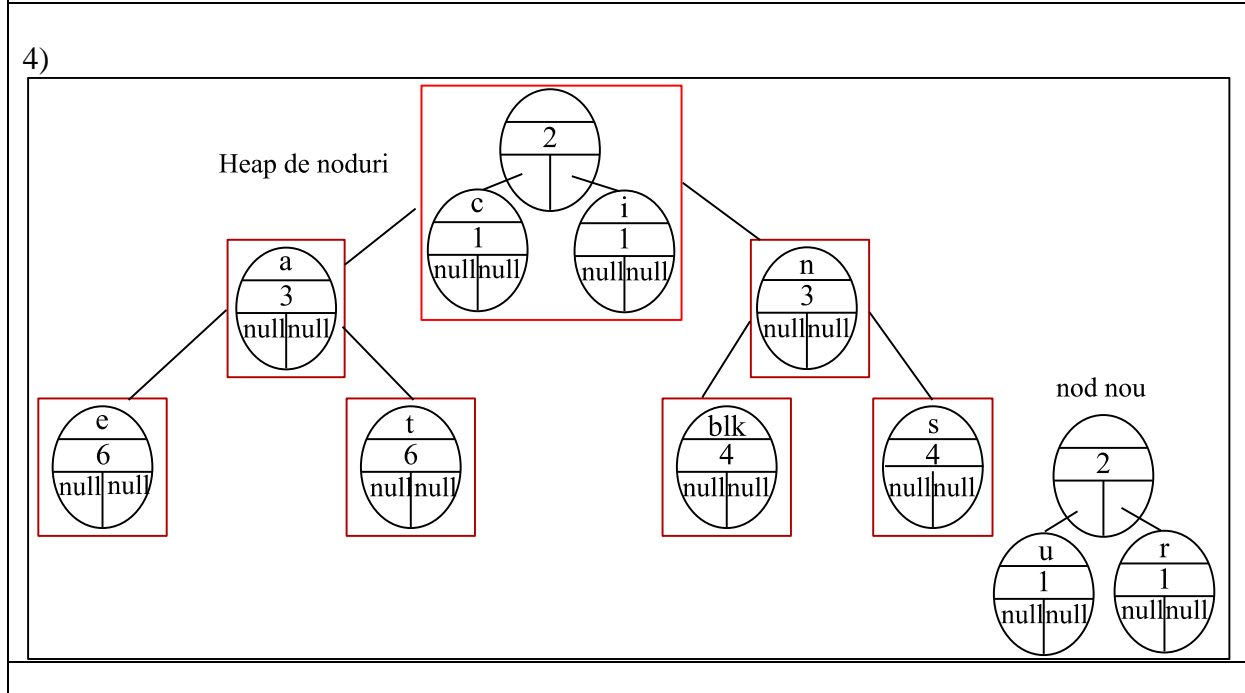
<div> <div>c</div> <div>1</div> <div> <div>null</div> <div>null</div> </div> </div>	<div> <div>i</div> <div>1</div> <div> <div>null</div> <div>null</div> </div> </div>	<div> <div>r</div> <div>1</div> <div> <div>null</div> <div>null</div> </div> </div>	<div> <div>u</div> <div>1</div> <div> <div>null</div> <div>null</div> </div> </div>	<div> <div>a</div> <div>3</div> <div> <div>null</div> <div>null</div> </div> </div>	<div> <div>n</div> <div>3</div> <div> <div>null</div> <div>null</div> </div> </div>	<div> <div>s</div> <div>4</div> <div> <div>null</div> <div>null</div> </div> </div>	<div> <div>blk</div> <div>4</div> <div> <div>null</div> <div>null</div> </div> </div>	<div> <div>t</div> <div>6</div> <div> <div>null</div> <div>null</div> </div> </div>	<div> <div>e</div> <div>6</div> <div> <div>null</div> <div>null</div> </div> </div>
---	---	---	---	---	---	---	---	---	---

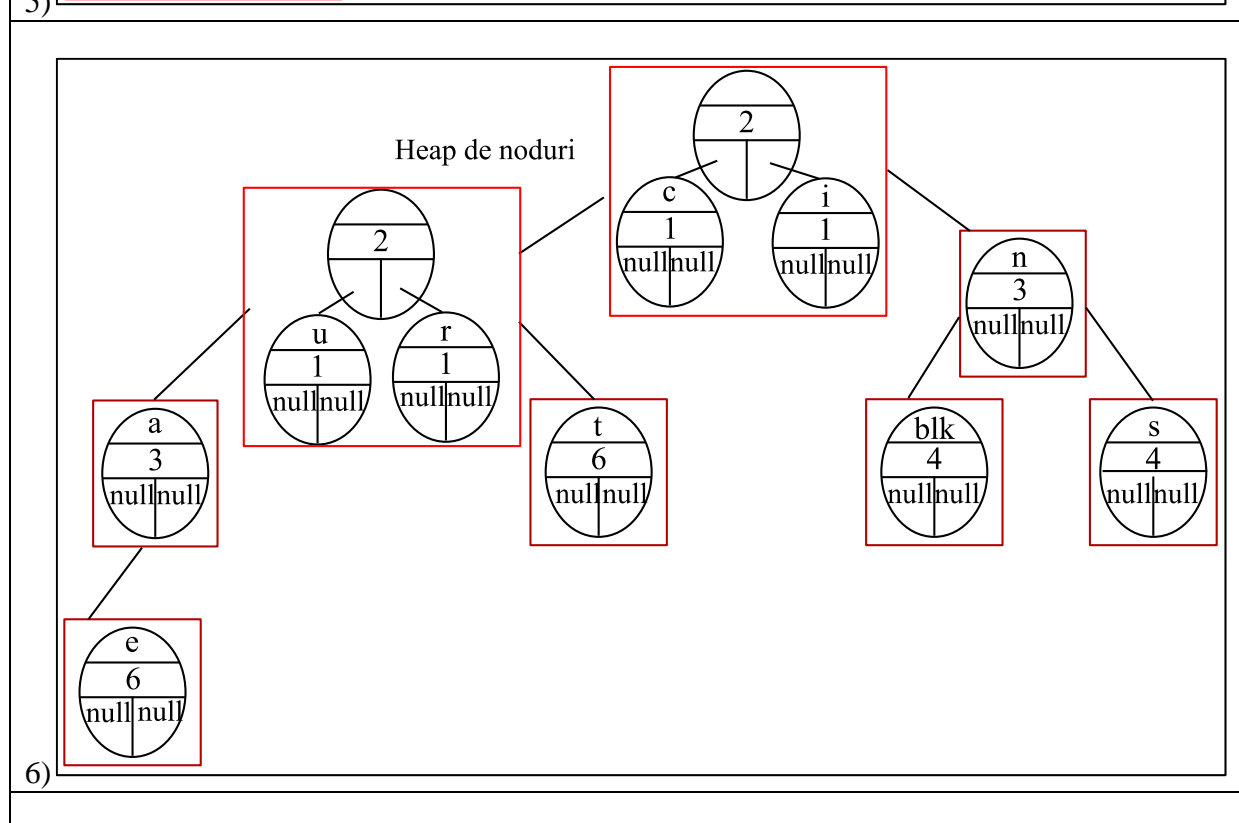
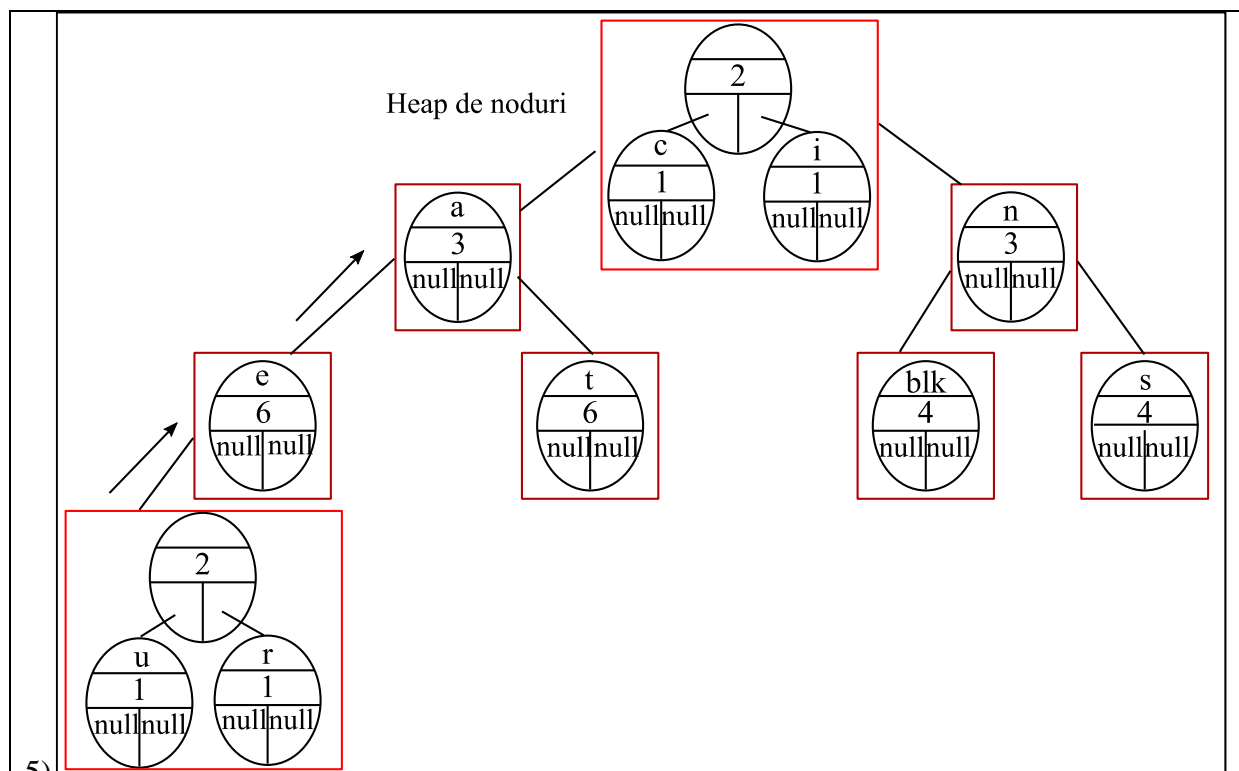
Pas 2: cât timp în coadă există mai mult decât un nod, se procedează în modul următor: se extrag primele noduri (cu frecvența minimă) din coadă, se creează un nou nod care are ca frecvență suma frecvențelor celor două noduri, se leagă cele două noduri ca fii ai noului nod și apoi se înserează apoi în coada de priorități. Dimensiunea heap-ului se reduce cu 1.

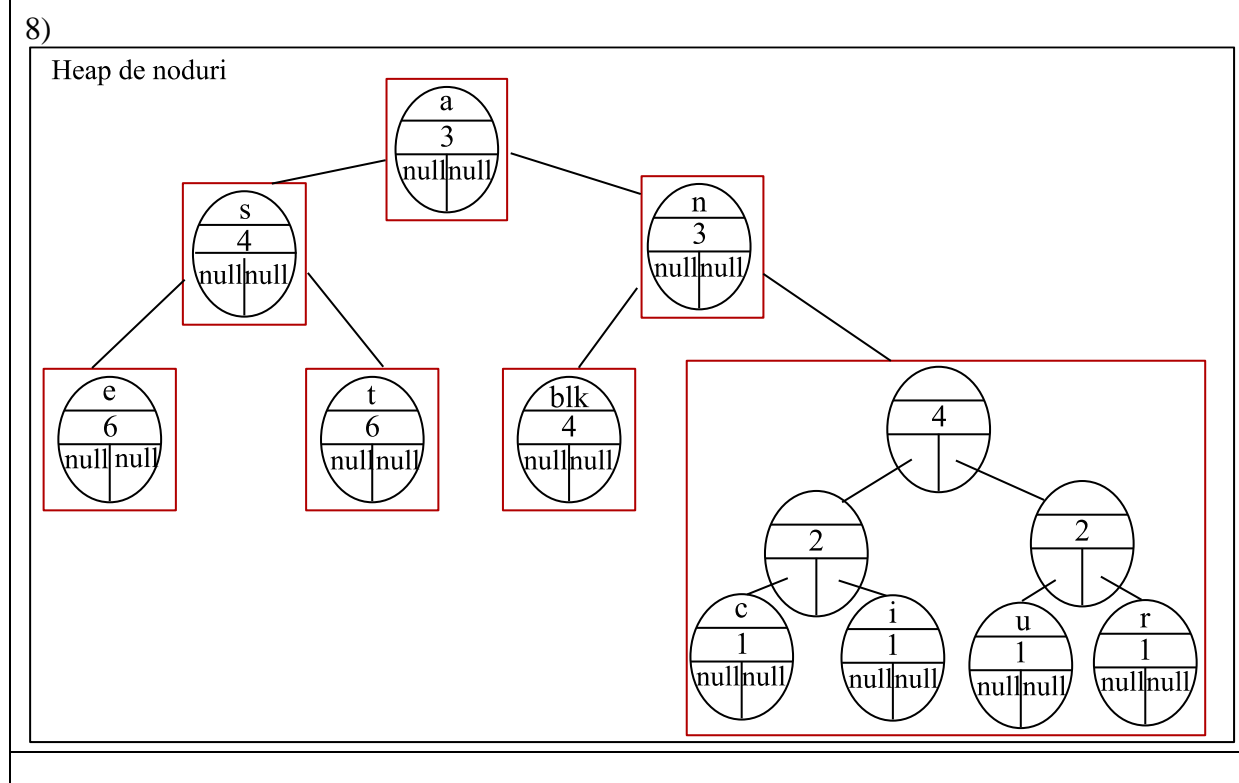
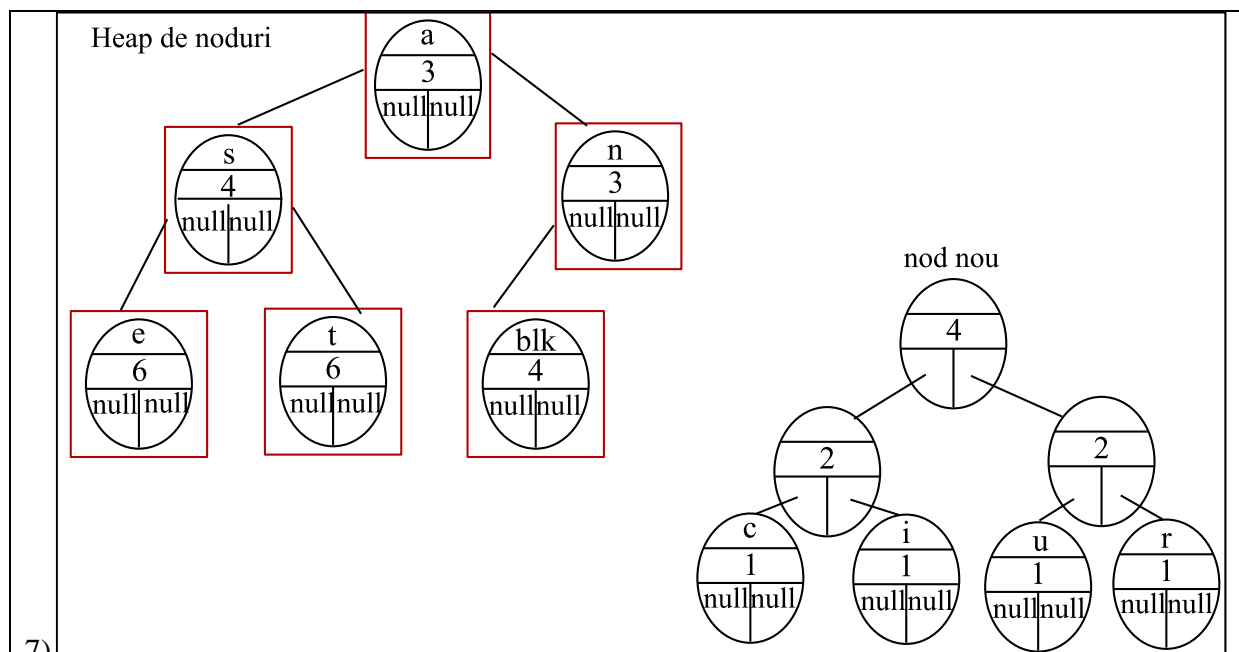




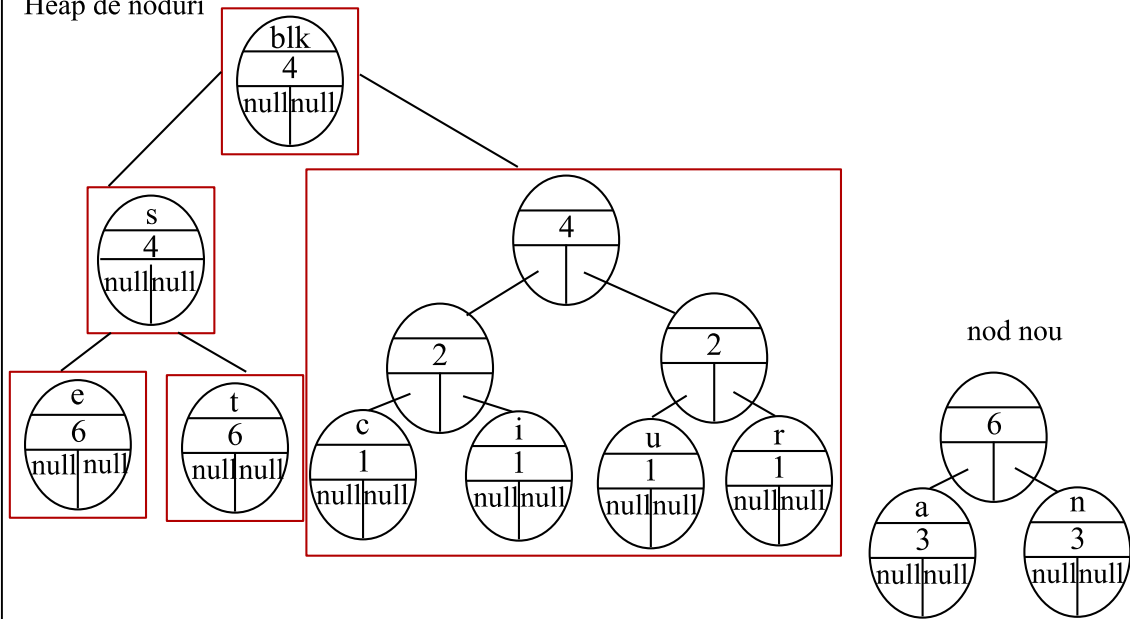
3)





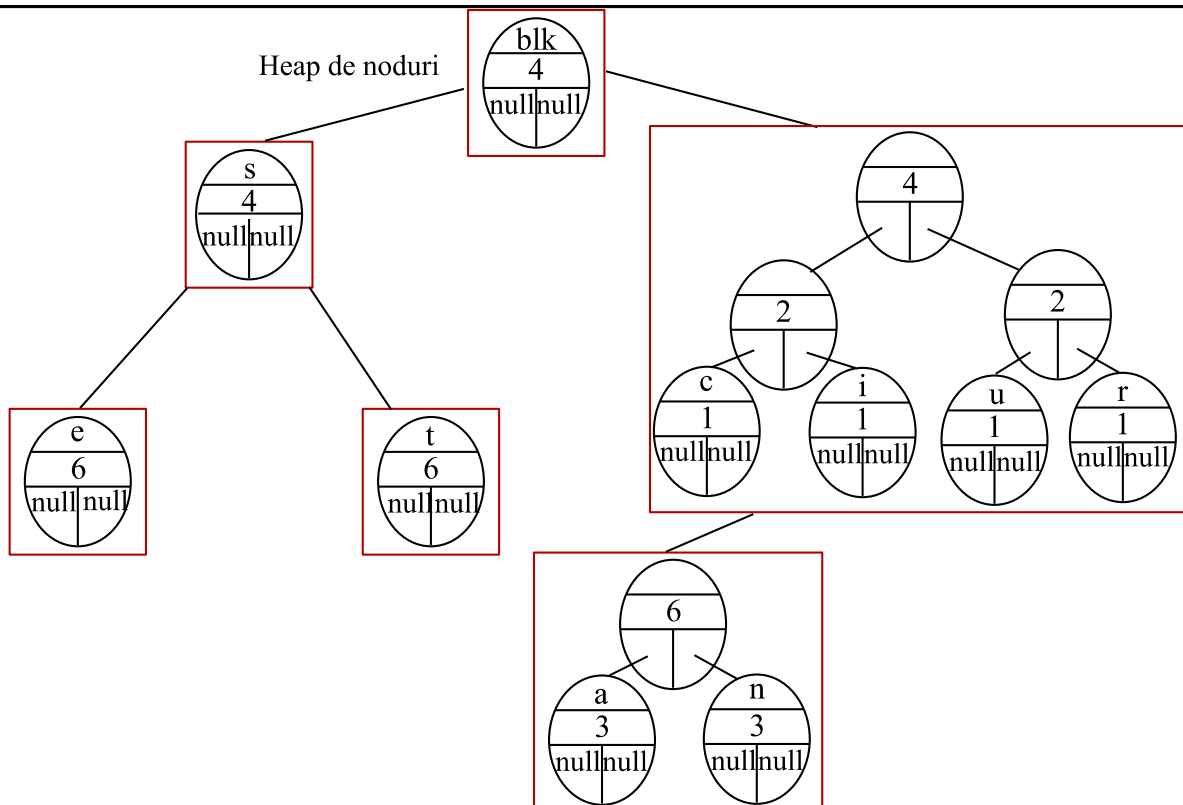


Heap de noduri



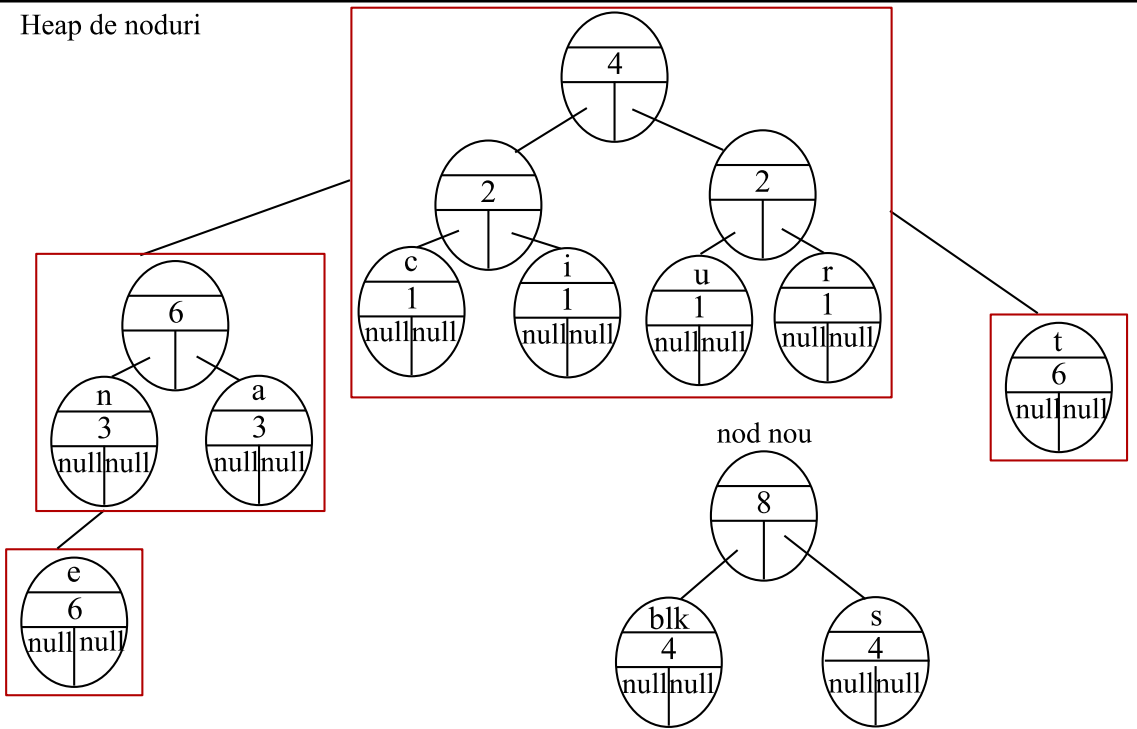
9)

Heap de noduri



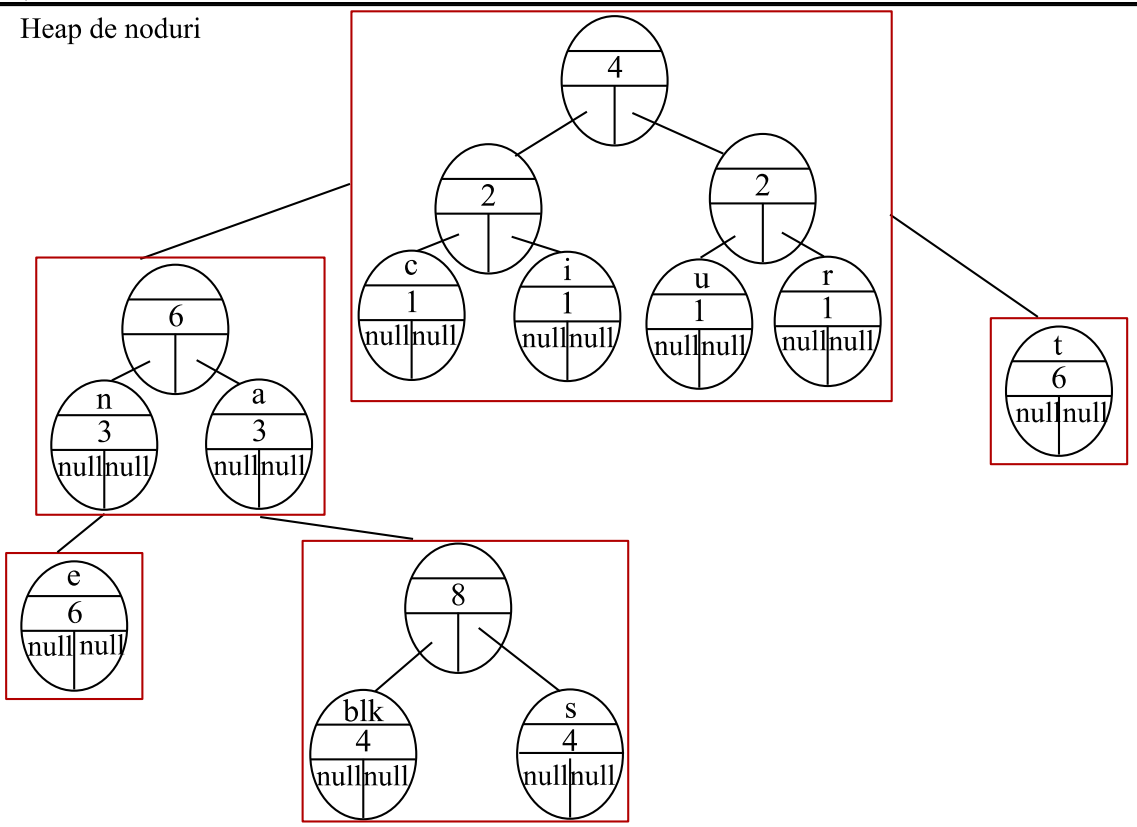
10)

Heap de noduri



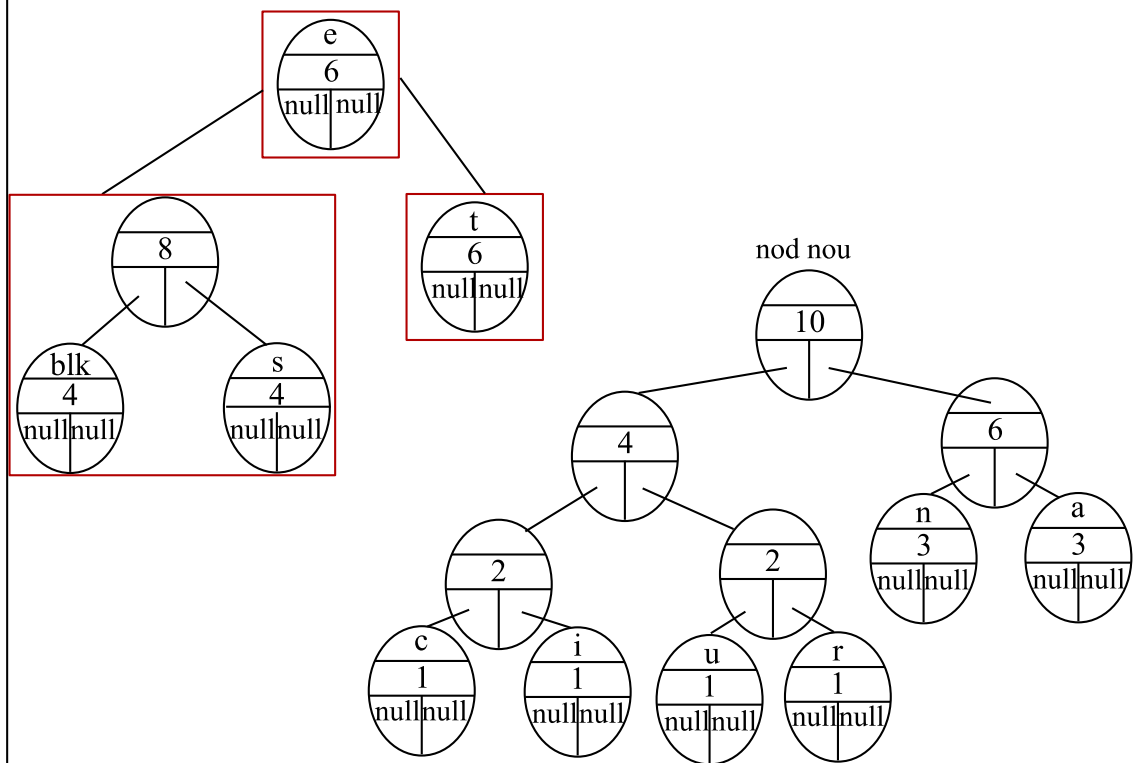
11)

Heap de noduri



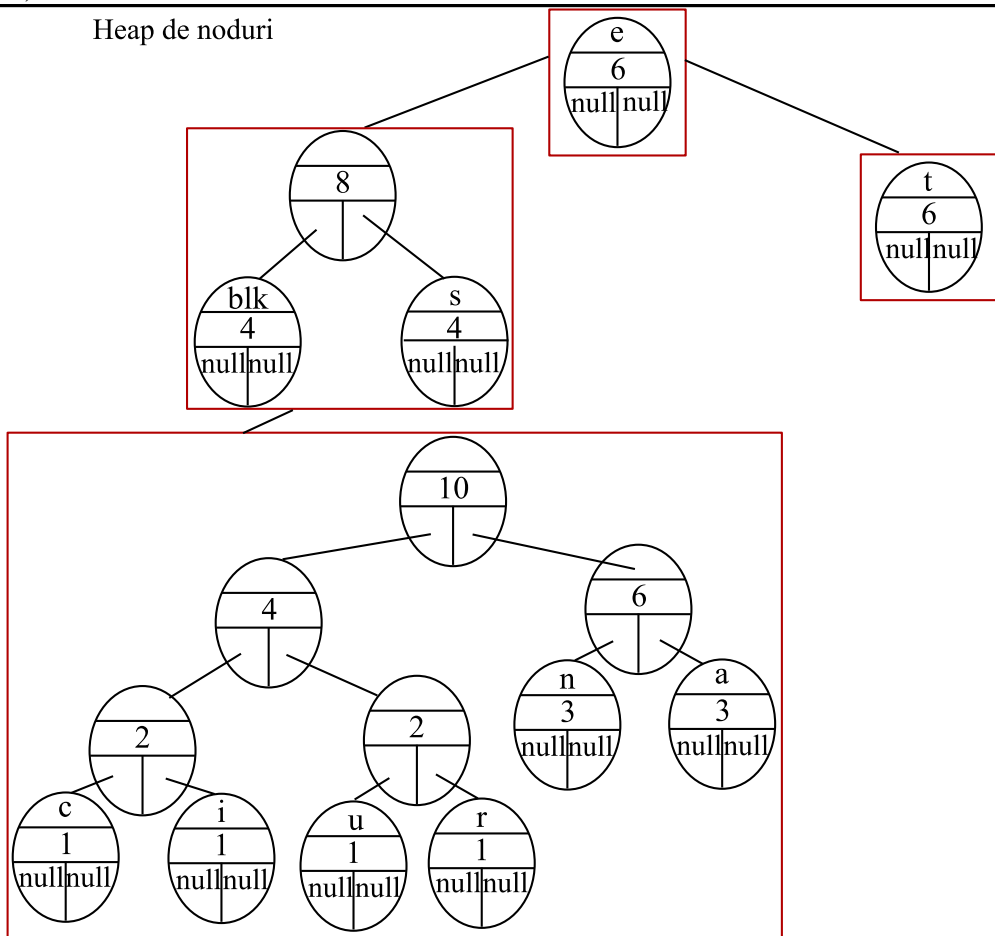
12)

Heap de noduri

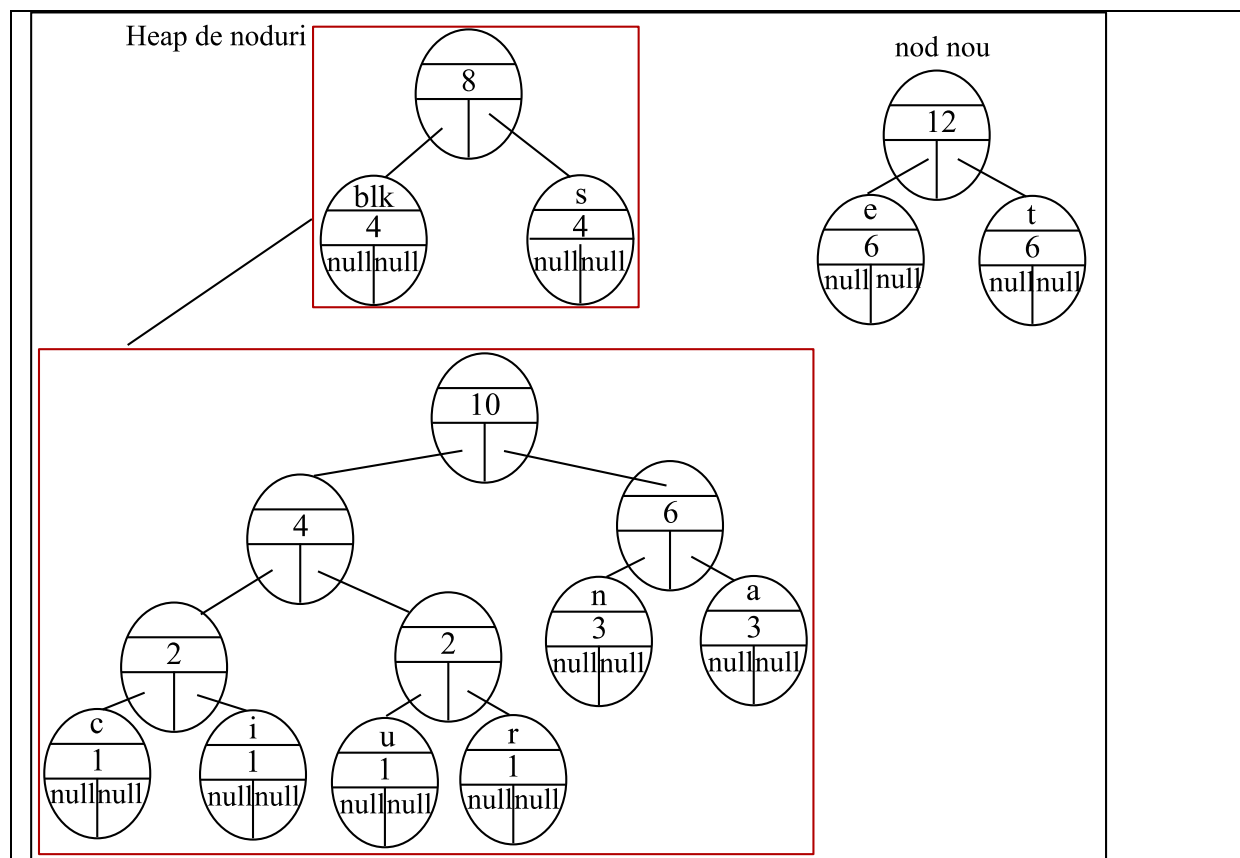


13)

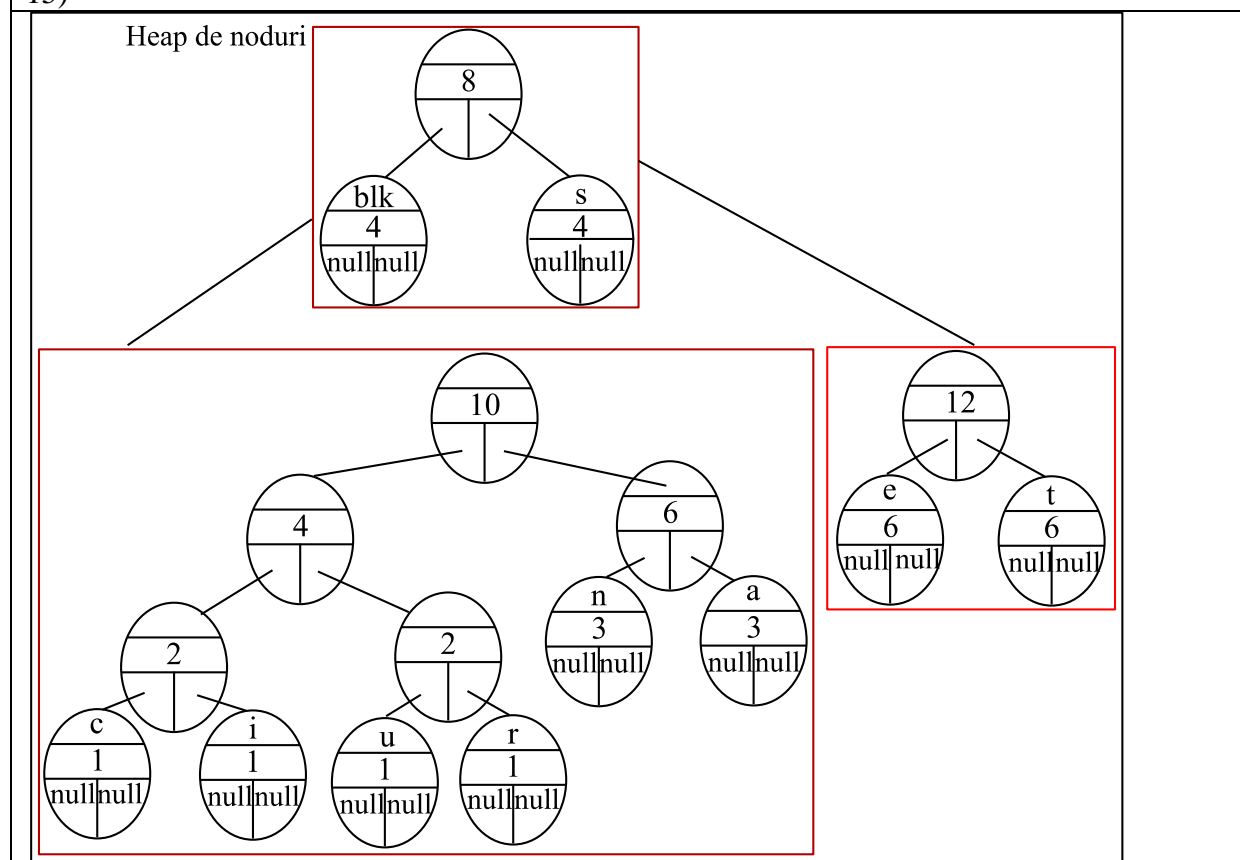
Heap de noduri



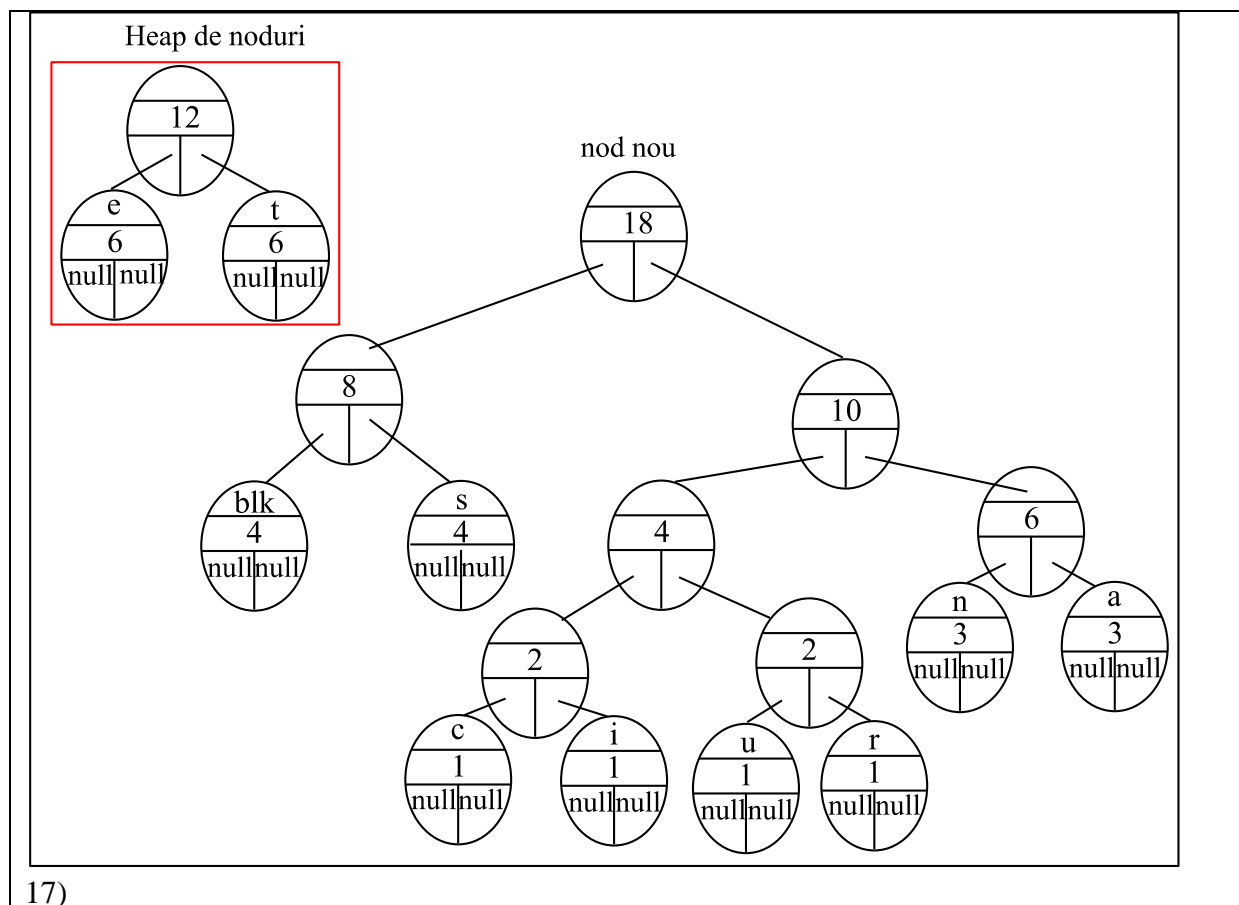
14)



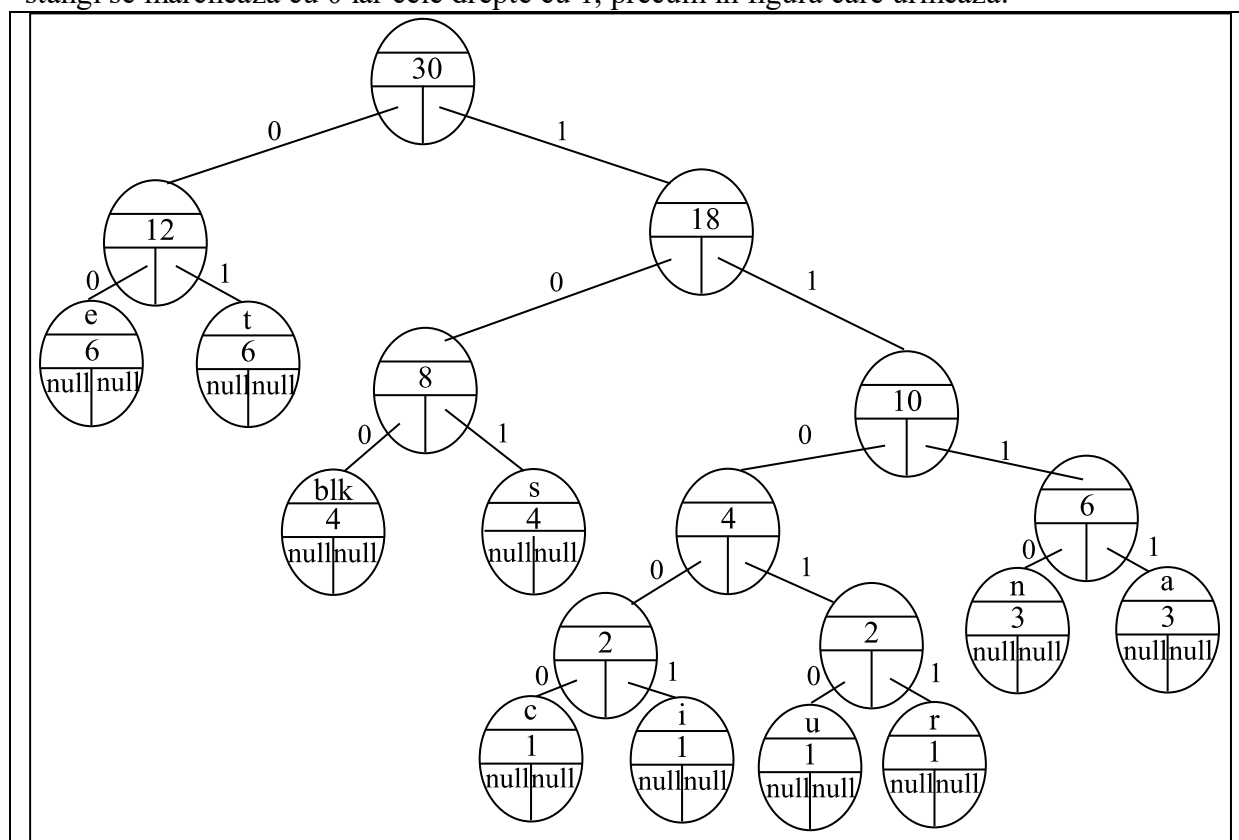
15)



16)



În final, după procesarea ultimelor două noduri se obține arborele Huffmann în care ramurile stângi se marchează cu 0 iar cele drepte cu 1, precum în figura care urmează:



Codurile pentru fiecare caracter dintr-o frunză se obțin prin alăturarea caracterelor de 0 și 1, care se întâlnesc pe drumul de la rădăcină la frunza respectivă.
Astfel pentru arborele de mai sus caracterele vor avea codurile:

Car	a	c	e	i	n	r	s	t	u	Blank
Fecv	3	1	6	1	3	1	4	6	1	4
Cod	1111	11000	00	11001	1110	11011	101	01	11010	100