



Laborator PS

ALGORITMI DE COMPRESIE

Algoritmul Huffman adaptiv

Prof. dr. ing. Dan STEFANOIU

As. Ing. Alexandru DUMITRASCU



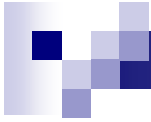
3. Algoritmul HUFFMAN adaptiv

Algoritmul de compresie adaptiva:

Pas 1: Initializarea arborelui binar

Pas 2: Pentru fiecare simbol citit de pe fluxul de intrare se executa urmatoarele operatiuni:

- Se genereaza noul cod corespunzator, pe baza modelului statistic curent
- Se reactualizeaza modelul statistic, pe baza noii informatii aduse de simbolul citit.



Algoritmul de decompresie adaptiva:

Pas 1: Initializarea modelului statistic (identica cu cea din faza de compresie)

Pas 2: Pentru fiecare cod citit de pe fluxul de intrare se executa urmatoarele operatiuni:

Se decripteaza codul, generand simbolul corespunzator, pe baza modelului statistic curent

Se reactualizeaza modelul statistic, pe baza noului simbol decriptat.

Obs: Rezultate corecte se obtin doar atunci cand se folosesc algoritmi identici de initializare si de reactualizare in ambele faze.



Constructia adaptiva a arborelui HUFFMAN

Obs: In cazul adaptiv, **nu** se cunosc dinainte dimensiunile setului de date si ale alfabetului corespondent, deci nu se mai foloseste aceeaasi maniera de constructie a arborilor binari prezentati la metodele anterioare.

Proprietatea de fraternitate:

- Fii aceluiasi nod parinte sunt indexati consecutiv
- Ponderea fiecarui nod creste sau ramane constanta odata cu cresterea indexului asociat
- Daca doua noduri sunt pe acelasi nivel ierarhic, fii nodului de indice mai mic au indici inferiori indicilor fiilor celuilalt nod.

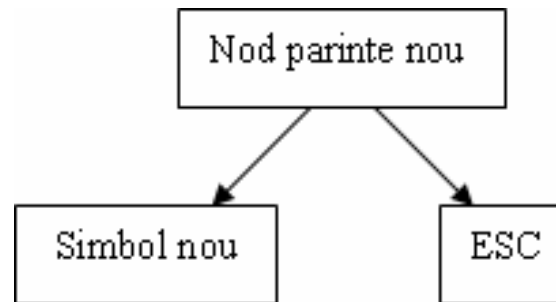
Def: Un arbore HUFFMAN este un arbore binar cu noduri ponderate care are proprietatea de fraternitate si ponderea unui nod parinte este egala cu suma ponderilor fiilor sai directi.

Fiecare nod al arborelui HUFFMAN este caracterizat de 3 parametri: ponderea asociata, pozitia sa in cadrul arborelui si tipul sau.

1) Initializarea arborelui HUFFMAN

Necunoasterea a priori a dimensiunilor setului de date si A^0 determina adaugarea la alfabet a 2 simbolii virtuali, diferiti de oricare dintre simbolii:

- simbol de sfarsit – **EOS** (end of string) – marcheaza sfarsitul setului de date si este reprezentat pe 2 octeti si are valoarea 256
- simbol de evitare – **ESC** (escape) – reprezinta fratele noului simbol citit/decriptat, adica fiul pereche al nodului parinte care se creaza cu ocazia aparitiei noului simbol. Este reprezentat pe 2 octeti si are valoarea de 257.

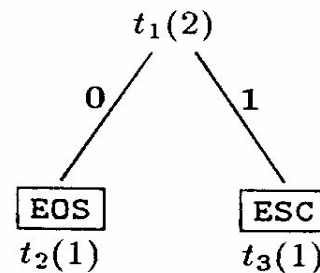


→ Simbolul ESC apare in sirul de date utile comprimate ori de cate ori codul care ii urmeaza are 8 biti si descrie un nou simbol original, diferit de cei anteriori.

→ Simbolul EOS apare o singura data, marcand sfarsitul sirului de coduri. => lor li se atribuie cate un contor fix, nemodificabil in timpul compresiei sau decompresiei, de valoare egala cu 1.

Arborele initial:

$$\mathcal{A}^{0,2} = \{\text{EOS}, \text{ESC}\} .$$



Arborele HUFFMAN inițial.

Obs:

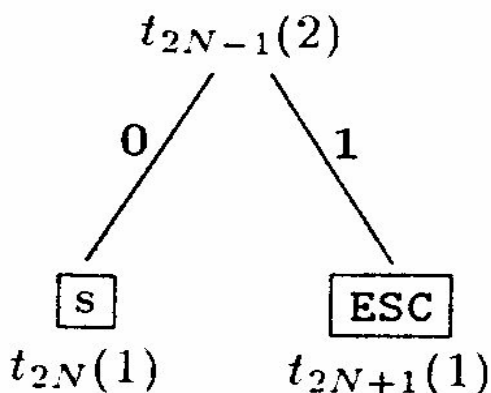
Ultimul nod in arbore va avea indicele maxim si va fi intotdeauna simbolul de evitare ESC, pentru ca, prin reactualizare, indicele atribuit simbolului de sfarsit nu trebuie modificat, si totodata se respecta ordinea lexicografica de pe acelasi nivel ierarhic.

2) Reactualizarea arborelui HUFFMAN

Obs: Arborele are un numar impar de noduri pe tot parcursul constructiei sale. Nodurile sunt indexate incepand cu numarul 1 de la radacina si terminand cu numarul $2N-1$ (ESC).

Pas 1: Daca noul simbol nu se afla in alfabetul curent atunci:

- nodul frunza ocupat de ESC devine nod intermediar, t_{2N-1} , iar ponderea sa devine egala cu 2
- noul nod intermediar, t_{2N-1} , devine nod parinet pentru 2 noduri fii: , t_{2N} ocupat de simbolul s , cu ponderea 1 si t_{2N+1} ocupat de simbolul ESC, cu ponderea 1
- odata cu modificarea ponderii nodului t_{2N-1} se poate pierde si *proprietatea de fraternitate* a arborelui. Aceasta este refacuta atat prin mutari de subarbori cat si prin reactualizari ale tuturor contoarelor nodurilor noului arbore (Pas 3).






Pas 2: Daca noul simbol exista deja in alfabetul curent atunci:

- contorul simbolului s tebuie incrementat cu o unitate
- schimbarea ponderii nodului poate conduce la pierderea proprietatii de fraternitate a arborelui (Pas 3).

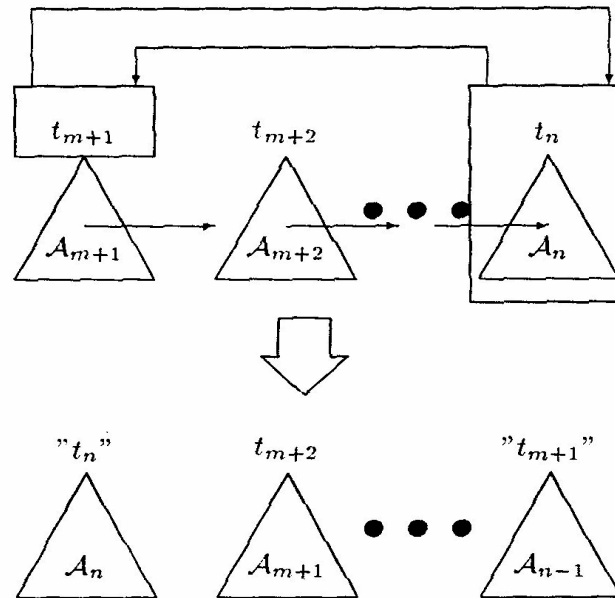
Pas 3: Refacerea proprietatii de fraternitate si reactualizarea ponderilor intregului arbore binar.

In arborele binar se afla un *nod critic* t_n ($n = 2.....2N-1$) a carui pondere a crescut cu o unitate. Pentru refacerea proprietatii de fraternitate trebuie gasit noul loc pe care il va ocupa nodul critic t_n impreuna cu toti descendentii sai (daca exista) si sa se incrementeze cu o unitate ponderea noului sau parinte => acesta va deveni, la randul sau, noul nod critic si astfel se va relua procedura pana cand se va ajunge in radacina.

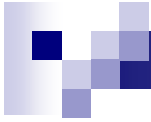


Reindexarea nodului critic t_n si a altor noduri in scopul recastigarii fraternitatii se face astfel:

- plecand de la indicele n se cauta in ordinea descrescatoare a indicilor primul nod cu pondere cel putin egala valorii $N(t_n)+1$. Fie acesta nodul t_m .
- daca $m+1=n$ nu mai este necesara mutarea lui t_n si a descendentilor sai, ci se incrementeaza ponderea parintelui lui t_n , care devine la randul sau nod critic si algoritmul se reia de la pasul 3.
- daca $m+1 < n \Rightarrow$ intre cele doua noduri exista cel putin un nod cu pondere strict inferioara ponderii nodului critic. Pentru refacerea proprietatii de fraternitate este necesara interschimbarea pozitiilor nodurilor t_{m+1} si t_n impreuna cu subarborii lor. Astfel:
 - daca $m+1=n-1 \Rightarrow$ intre t_m si t_n este un singur nod, t_{n-1} .
 - daca $m+1 < n-1 \Rightarrow$ intre t_m si t_n exista cel putin doua noduri intermediare $(t_{m+1}, t_{m+2}, \dots, t_{n-1})$. In acest caz, interschimbarea pozitiilor $t_{m+1} \leftrightarrow t_n$ se face astfel:



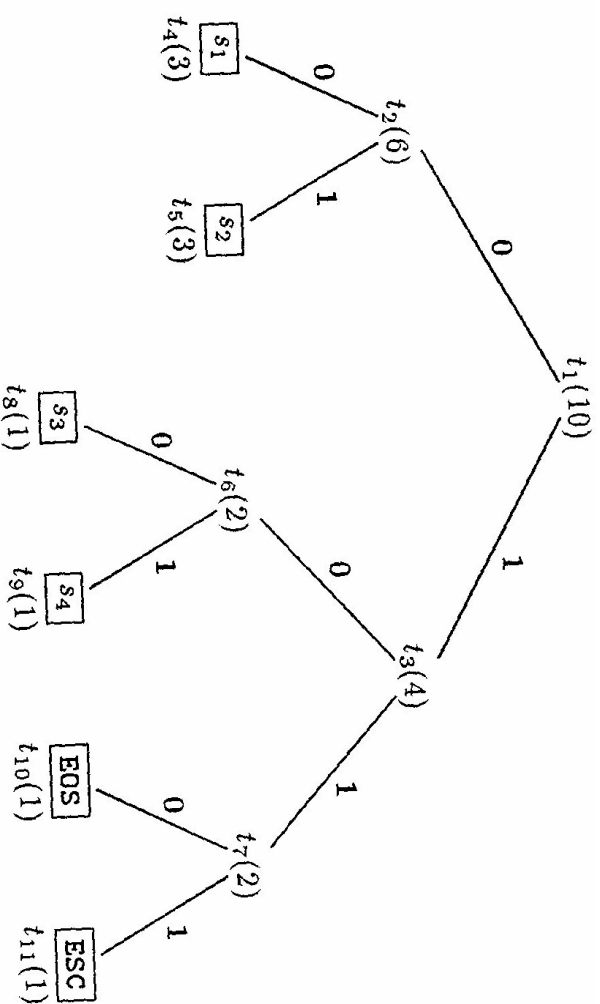
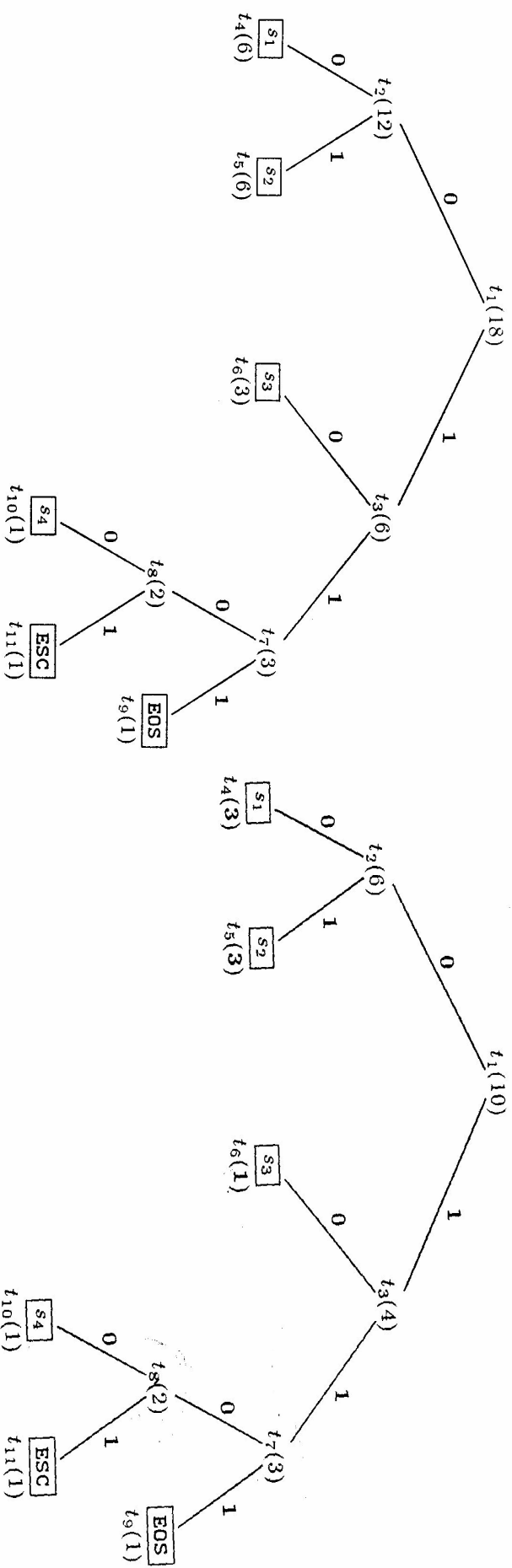
- t_n se muta împreuna cu arborele lui descendent in noua pozitie, t_{m+1} .
- Arborele descendent al lui t_{n-1} devine noul arbore descendent al lui t_{m+1} .
- Subarborii nodurilor intermediare isi schimba parintele direct cu o pozitie spre stanga.
- In final, nodul t_{m+2} primeste arborele descendent al nodului t_{m+1} .
- Daca printre nodurile intermediare se intalnesc frunze, ele vor fi sarite.
- Dupa interschimbare, ponderea nodului t_{m+1} se incrementeaza si astfel devine noul nod critic, algoritmul reluandu-se de la pasul 3.

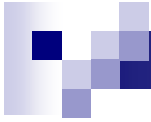


Scalarea contoarelor in cursul reactualizarii

In cazul sirurilor lungi de date este posibil ca operatia de incrementare a contoarelor nodurilor arborelui HUFFMAN sa conduca la depasirea numarului maxim ce poate fi reprezentat in memorie. Aceasta se poate remedia prin rescalarea contoarelor asociate nodurilor, in doua etape:

- se impart la 2 toate contoarele asociate frunzelor, pastrandu-se contoarele unitare
- se reface integral structura arborelui HUFFMAN, in maniera statica, respectand proprietatea de fraternitate.





Obs:

Forma arborelui binar se schimba → se schimba lungimea codurilor asociate frunzelor.

Scalarea contoarelor maresta ponderea simbolilor recent aparuti, in defavoarea simbolilor vechi, care sunt ponderati. Aceasta operatie este necesara pentru evitarea blocarii algoritmului.