Analiză a algoritmilor Huffman coding si Arithmetic coding, folosiți în compresia datelor

Universitatea Politehnica București, Facultatea de Automatică și Calculatoare

Naumencu Mihai - 322CD

1 Introducere

a. Descrierea problemei rezolvate

Compresia datelor este ansamblul prelucrărilor ce se aplică unor date în scopul reducerii dimensiunii reprezentării acestora , eficiența unei astfel de compresii putând fi apreciată prin folosirea raportului de compresie, acesta fiind egal cu raportul dintre dimensiunea reprezentării datelor în lipsa compresiei și dimensiunea reprezentării datelor obținute în urma compresiei.

Compresia se realizează prin schimbarea modului de reprezentare a datelor având deci de a face cu un caz particular de codare. Codarea se face în raport cu un anumit model al datelor, aflându-se astfel în cazul mai general de modelare, al căutării unui model corespunzător al datelor.

Cea mai generală clasificare a metodelor e compresie se face după eroarea de refacere a datelor în raport cu acest criteriu se disting două categorii de metode:

- Metode cu pierderi – în care datele se refac în limita unor erori considerate acceptabile. (LOSSY)
- Metode fără pierderi – în care datele se refac în totalitate, fără a exista nicio diferență între datele originale și cele refăcute. (LOSSLESS)

În realizarea acestui proiect va fi luată în considerare doar metoda fără pierderi. (1)

b. Exemple de aplicații practice

Fiind o parte fundamentală a transferului și a manipulării de informații, compresia datelor este folosită într-o varietate de cazuri, de la compresie de text, imagini, compresie audio, video, pâna la compresia materialului genetic.

c. Soluții alese

În cadrul acestui proiect vor fi analizați doi algoritmi de compresie a datelor fără pierderi de informație, și anume Huffman coding și Arithmetic coding.

Huffman coding

Huffman coding este o formă de codificare entropică utilizată în compresia de date fără pierdere de informații.

Ieșirea din algoritmul Huffman poate fi văzută ca un tabel de coduri cu lungime variabilă pentru codificarea unui simbol sursă. Algoritmul derivă acest tabel din probabilitatea estimată sau frecvența de apariție pentru fiecare valoare posibilă a simbolului sursă. Ca și în alte metode de codificare a entropiei , simbolurile mai frecvente sunt în general reprezentate folosind mai puțini biți decât simbolurile mai puțin comune. (2)

Arithmetic coding

Arithmetic coding este, de asemenea, o formă de codificare entropică utilizată în compresia datelor fără pierdere de informații.

În general, codificatorii aritmetici pot produce o ieșire aproape optimă pentru orice set dat de simboluri și probabilități. Algoritmii de compresie care utilizează codarea aritmetică încep prin determinarea unui model al datelor - practic o predicție a tiparelor care vor fi găsite în simbolurile mesajului. Cu cât această predicție este mai precisă, cu atât rezultatul va fi mai aproape de optim. (3)

d. Criterii de evaluare

În vederea evaluării performanțelor celor doi algoritmi, am folosit 12 fisiere text cu dimensiuni diferite (testul 2 - Luceafărul, Mihai Eminescu 11 KB, testul 8 - Razboaiele Galice, Iulius Caesar 495 KB, testul 11 - Biblia 4,2 MB).

Evaluarea a fost făcută pe două criterii importante: raportul de compresie (dimensiunea originală a fișierului / dimensiunea după efectuarea compresiei) și resuresele consumate la comprimare (timpul de execuție și memoria folosită).

2 Prezentarea soluțiilor

a. Descrierea modului în care funcționează algoritmii aleși

Huffman coding

Codificarea Huffman este o varianta foarte populara de codificare cu lungime variabila. Ea presupune ca datele sunt reprezentate, in general, ca un sir de simboluri (octeti, cifre, litere, de exemplu), unde fiecare simbol din multimea tuturor simbolurilor posibile (alfabetul A = a1, a2, . . . , an), are asociata o anumita probabilitate de aparitie pi. Probabilitatile se pot determina fie estimativ (de exemplu pe baza tipului sirului de date: text, imagine, cod executabil), fie inspectand sirul de la cap la coada si numarand aparitiile fiecarui simbol, apoi impartind fiecare contor la numarul total de simboluri. (Figura 1)

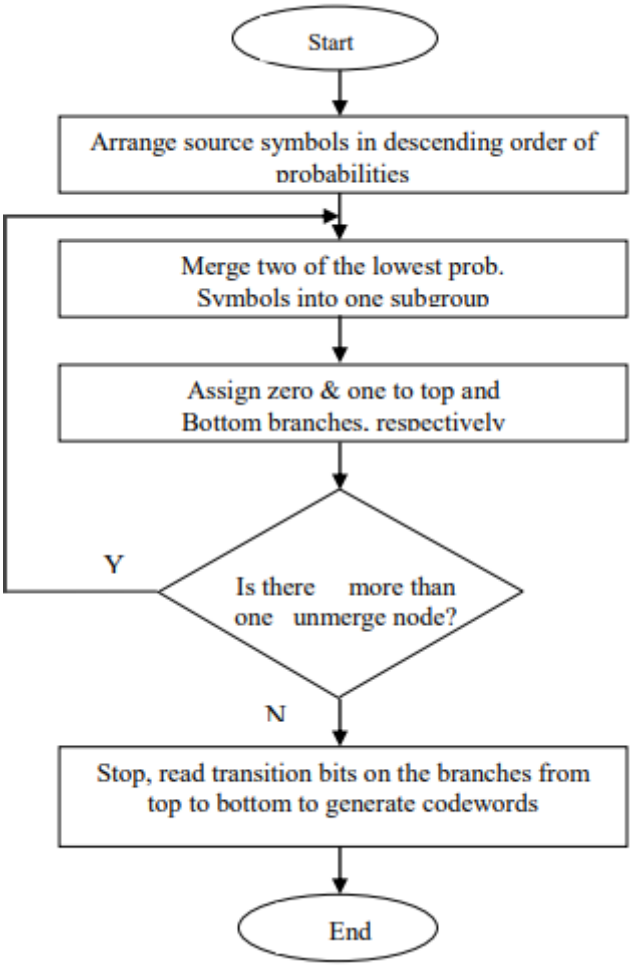

Figure 1: Huffman coding algorithm flowchart (5)

Arithmetic coding

Codificarea aritmetică este o formă de codificare entropică utilizată în compresia de date fără pierderi. În mod normal, un șir de caractere, este reprezentat folosind un număr fix de biți pe caracter, ca în codul ASCII . Când un șir este convertit în codificare aritmetică, caracterele utilizate frecvent vor fi stocate cu mai puțini biți și caracterele care nu apar atât de frecvent vor fi stocate cu mai mulți biți, rezultând mai puțini biți utilizați în total. Codificarea aritmetică diferă de alte forme de codificare a entropiei, cum ar fi codarea Huffman , prin faptul că, mai degrabă decât separarea intrării în simboluri componente și înlocuirea fiecăruia cu un cod, codificarea aritmetică codifică întregul mesaj într-un singur număr, o fracție de precizie arbitrară q unde 0,0 ¡= q ¡ 1,0. Reprezintă informațiile curente ca un interval, definit prin două numere. Această metodă permite implementări mai rapide datorită faptului că operează direct pe un singur număr natural care reprezintă informațiile curente. (Figura 2)

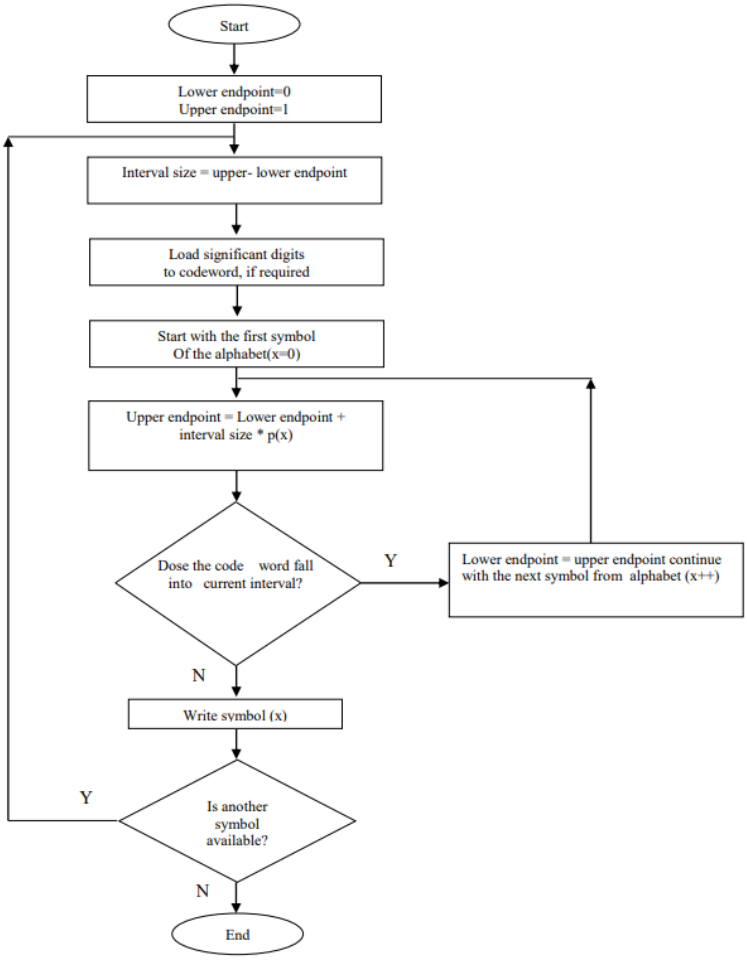

Figure 2: Arithmetic coding algorithm flowchart (5)

b. Analiza complexității soluțiilor

- Complexitate Huffman coding: $O(n^2)$
- Complexitate Arithmetic coding O(n log(n))

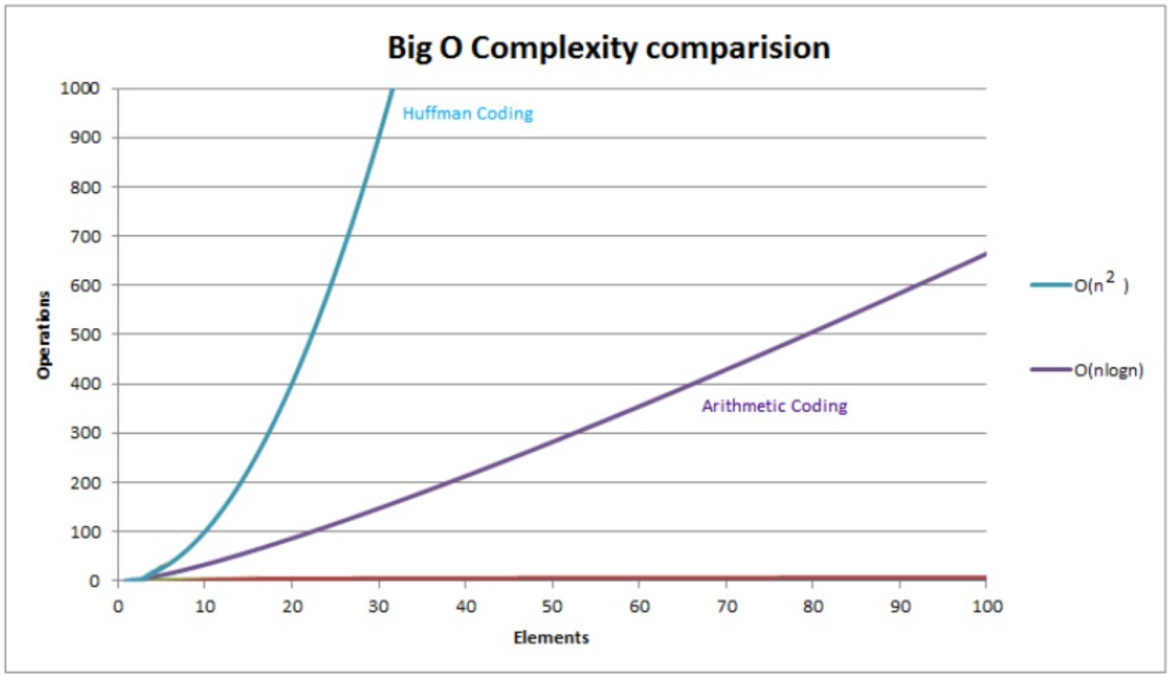

Figure 3: Grafic complexități (6)

c. Principale avantaje și dezavantaje

Un dezavantaj al codificării Huffman este că acesta necesită două treceri, una pentru a construi un model statistic al datelor și o a doua pentru a codifica, deci este un proces relativ lent. Ceea ce înseamnă că tehnicile de codare fără pierderi care utilizează codificarea Huffman sunt în mod semnificativ mai lente decât alte tehnici în vederea citirii sau scrierii unor fișiere.

Un avantaj al codificării aritmetice față de alte metode similare de compresie a datelor este comoditatea adaptării. Datele decodate se potrivesc cu datele originale, atâta timp cât tabelul de frecvențe din decodare este înlocuit în același mod și în codificare. Sincronizarea se bazează, de obicei, pe o combinație de simboluri care apar în timpul procesului de codificare și decodare.

3 Evaluare

a. Teste folosite pentru evaluare

Am folosit un set de teste (12 la număr) text cu dimensiuni diferite (primele teste au cațiva KB în timp ce ultimul are 100MB), acestea sunt diverse texte literare (poezii în cazul primelor teste precum Luceafărul de Mihai Eminescu sau o colecție de poezii de Lucian Blaga; texte epice istorice: Republica de Platon, Prințul de Machiavelli, Odisea de Homer; Biblia creștină și o baza de date Wikipedia - testul 12).

b. Specificațiile sistemului de calcul folosit

Testele au fost rulate pe un sitem Windows 10 de 64 de biți, cu un procesor Intel i5-9300H si 8GB de memorie RAM.

c. Rezultatele evaluarii

Raportul de Compresie

Fisier Test	Dimensiune originala (KB)	Dimensiune dupa Compresie (KB)		Raport de Compresie	
		Huffman	Arithmetic	Huffman	Arithmetic
Test 1	1	1	2	1.00	0.50
Test 2	11	8	8	1.37	1.37
Test 3	14	10	10	1.40	1.40
Test 4	19	12	12	1.58	1.58
Test 5	93	55	54	1.69	1.72
Test 6	102	58	57	1.76	1.79
Test 7	299	171	169	1.75	1.77
Test 8	495	277	274	1.79	1.80
Test 9	608	340	337	1.79	1.80
Test 10	680	381	377	1.78	1.80
Test 11	4.329	2.508	2.488	1.72	1.74
Test 12	97.657	NA	62.015	NA	1.57

* NA – in cazul testului 12, la rularea compresiei Huffman rezulta eroarea OutofMemoryError

Figure 4: Tabel raport de compresie

Timpii de executie (s)

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Test 11	Test 12
Huffman	0.0221	0.3671	0.3751	0.1887	0.3652	0.3506	0.8693	1.4174	1.7297	1.9402	12.36	NA
Arithmetic	0.0103	0.0079	0.0045	0.0078	0.0189	0.0120	0.0312	0.0522	0.0668	0.0900	0.4914	10.78

* NA – in cazul testului 12, la rularea compresiei Huffman rezulta eroarea OutofMemoryError

Figure 5: Tabel timpi de execuție

Memoria folosita (KB)

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Test 11	Test 12
Huffman	2,992.60	11,184.13	13,785.10	17,408.0	25,869.58	33,546.17	56,182.10	133,121.33	126,749.3	112,717.55	626,176.00	NA
Arithmetic	2,521.67	2,992.82	2,993.11	2,993.10	4,016.11	4,174.32	5,608.65	7,640.55	6,694.88	8,538.30	10,409.75	51,652.43

* NA – in cazul testului 12, la rularea compresiei Huffman rezulta eroarea OutofMemoryError

Figure 6: Tabel memorie folosită

d. Valorile obținute

Se poate observa din figura 4 că raportul de compresie rezultat în urma aplicării clor doi algoritmi este unul similar, în schimb, când sunt luați în considerare timpii de execuție (figura 5) situația este foarte diferită, timpii ambilor algoritmi crește în funcție de mărimea fișierului asupra căruia se realizează procesul de compresie însă creșterea este mai rapidă în cazul codificării Huffman decât în cazul celei Aritmetice. O situație similară este și în cazul memoriei folosite în cadrul proceselor de compresie, algoritmul Huffman ajunge să consume foarte multe resurse odată cu creșterea dimensiunii fișierului. În cazul testului 11 (cu o dimensiune originală de 4.2MB) codificarea Huffman consumă în jur de 600KB, iar în cazul testului 12 (cu o dimensiune originală de 100MB) apare o eroare la rulare legată de memoria insuficientă a sistemului (există posibilitatea ca implemetarea codificarii Huffman realizată să nu fie una optimă).

4 Concluzii

În concluzie, pentru compresia fișierelor de mici dimensiuni, ambele metode sunt viabile, însă algoritmul de codificare Aritmetică îl surclasează pe cel Huffman cel puțin când vine vorba despre compresia fișierelor de tip text. În cadrul implementării codificării Aritmetice se regăsesc mai puține operații aritmetice, ceea ce duce și la o complexitate mai redusă a acestuia.

Î̂n ansamblu se poate spune că, cu privire la compresia fișierelor de tip text, algoritmul de codificare Aritmetic este superior și mai aproape de optim decât cel Huffman.

5 Referințe

(1) http://webspace.ulbsibiu.ro/macarie.breazu/ACM/Introducere.pdf
(2) https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/
(3) https://www.geeksforgeeks.org/arithmetic-encoding-and-decoding-using-matlab/
(4) https://ocw.cs.pub.ro/courses/sd-ca/2015/teme/doc-tema04
(5) https://arxiv.org/ftp/arxiv/papers/1109/1109.0216.pdf
(6) https://www.slideshare.net/ramakantsoni/performance-analysis-of-huffman-and-arithmeti
(7) http://webspace.ulbsibiu.ro/arpad.gellert/html/Algoritmi.pdf
(8) http://mattmahoney.net/dc/text.html