

Compresia Datelor

Stoica Robert-Valentin

Universitatea Politehnica, Bucuresti

Abstract. Scopul acestei lucrari este acela de a aduce in atentie si a analiza eficienta algoritmilor de compresie Huffman si Lempel-Ziv-Welch (LSW) pe un set cat mai amplu de date.

Keywords: Compresie · Huffman Coding · Lempel-Ziv-Welch

1 Introducere

1.1 Descrierea problemei

Compresia datelor este un aspect foarte important atunci cand vorbim de lucrul cu fisiere de orice tip (audio, video, text etc) deoarece permite partajarea si pastrarea lor intr-o forma redusa, simplificata, putand astfel economisi resurse.

Folosirea fisierelor comprimate in locul celor originale ne ofera atat eficienta in cadrul utilizarii memoriei fizice (pastrarea lor intr-o forma redusa va ocupa mai putin spatiu pe HDD/ SSD) cat si una temporala in cazul in care dorim sa partajam resurse prin intermediul internetului, un fisier de dimensiune mai mica fiind incarcat si transferat mult mai usor.

1.2 Specificarea solutiilor alese

Cele doua metode de compresie ce urmeaza a fi analizate fac parte din clasa "lossless data compression" (compresie fara pierdere a datelor), asta insemanand ca, in urma procesului de decompimare, fisierul poate fi adus la forma sa initiala fara a suferi modificari de orice tip.

In cazul compresiei Huffman, algoritmul creeaza un cod binar unic pentru fiecare caracter (sau octet) existent in cadrul fisierului. Acest cod va fi mai scurt sau mai lung in functie de frecventa fiecarui caracter. Astfel, caracterele ce apar foarte des vor avea un cod binar **scurt** iar cele ce apar rar, vor avea unul mult mai **lung**. "Magia" acestui algoritm sta in capacitatea sa de a compensa codurile scurte ce apar de multe ori cu codurile lungi ce apar mult mai rar, in medie acesta reusind sa aduca fisierul la o dimensiune mai mica decat cea originala.

In schimb, metoda de compresie Lempel-Ziv-Welch aduce in plus posibilitatea de a codifica **secvente de mai multe caractere ce se repeta**. Algoritmul aduce cu sine un dictionar ce isi mareste capacitatea cu fiecare secventa noua gasita. Desi codificarile vor fi facute pe un numar fix de biti (in general mai mare decat dimensiunea unui octet, ajungand pana la 12 sau 13 biti), daca fisierul are cuvinte sau secvente de litere ce se repeta, acestea vor aparea in fisierul comprimat sub forma unui singur cod.

1.3 Evaluarea soluțiilor

Singurul aspect ce ne interesează în cazul algoritmilor de compresie este dat de raportul dintre dimensiunea fișierului comprimat și cea a fișierului original, altfel spus, eficiența algoritmilor constă în **procentul de reducere** al fișierului original.

Pentru evaluarea eficienței algoritmilor voi încerca să acopăr o plajă cât mai mare a datelor de intrare atât prin tipul, dimensiunea dar și conținutul lor.

În cazul fișierelor text, voi genera date de intrare mai mult sau mai puțin favorabile pentru ambele tipuri de compresie. Un exemplu bun în cazul fișierelor text ar fi un fișier ce conține **o singură literă de un număr mare de ori**:

- în cazul algoritmului Huffman, această compresie este foarte bună, în noul fișier reținându-se un singur cod foarte scurt (+ datele auxiliare necesare decompresării)
 - pentru algoritmul LZW, compresia este una mult mai slabă, acesta adăugând grupuri de aceeași literă dar dimensiuni diferite în dicționar și în fișierul nou rezultat
- Totuși, în ambele cazuri, fișierul comprimat va fi **mai mic** decât cel inițial.

Un alt exemplu ar putea fi un text ce conține **caractere complet diferite**. În acest caz, pentru date de intrare mai mari, ambii algoritmi vor fi **extrem de ineficienți**, existând posibilitatea ca noul fișier să fie chiar mai mare decât cel original.

Testele vor fi făcute atât pentru cazuri bine alese (ca cele de mai sus), cât și pentru fișiere generate aleator (text copiat de undeva sau imagini care nu au un tipar anume).

O altă abordare ar putea fi data de testarea algoritmilor pe date de intrare care deja sunt supuse unui grad mare de compresie (imagini, fișiere PDF etc), moment în care ne așteptăm ca algoritmi să aibă un impact **nesemnificativ** (sau chiar unul rău - dimensiunea fișierului crește) asupra fișierului de intrare.

Pentru o parte din teste voi genera date **complet aleatoare** folosind comanda: `cat /dev/urandom`.

Scopul lucrării este acela de a analiza toate aceste situații și a observa în ce caz este mai bun un algoritm decât celălalt.

References

1. Autor, Gajendra Sharma, Analysis of Huffman Coding and Lempel-Ziv-Welch (LZW) Coding as Data Compression Techniques
2. <https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/>
3. <https://www.programiz.com/dsa/huffman-coding>
4. <http://www.cas.mcmaster.ca/cs2c03/2020/LN21-2020.pdf>
5. <https://www.youtube.com/watch?v=j2HSd3HCpDs&t=428s>
6. <https://www.youtube.com/watch?v=dM6us854Jk0>
7. <https://www.youtube.com/watch?v=JsTptu56GM8>