Proiect Huffman

Adnan Saadeddine

Contents

Tema proiectului

Sa se scrie un program care sa realizeze comprimarea unui fisier folosind metoda Huffman. Programul trebuie sa realizeze si decomprimarea unui fisier comprimat.

Limbajul de programare

Pentru implementarea proiectului s-a realizat o aplicatie web dezvoltata in Javascript folosind framework-ul React.

Pentru pornirea aplicatiei este nevoie de instalarea NodeJs (https://nodejs.org/en/). Apoi trebuie pornit un terminal in locatia proiectului si rulata comanda “npm start”.

Codul se regaseste pe Github: https://github.com/asaadedd/huffman

Modalitatea de rezolvare

Pentru realizarea proiectului trebuie implementate doua functionalitati: comprimarea si decomprimarea.

Comprimarea

Primul pas in realizarea unei comprimari Huffman este numararea fiecarui caracter din text.

Dupa ce avem aceasta lista trebuie sa facem un arbore, arborele Huffman, fiecare nod din acest arbore contine doi copii, stanga si dreapta. Frunzele acestui arbore sunt caracterele din text. Pentru realizarea arborelui se porneste de la cele mai putin utilizate caractere si se creaza un nod parinte care are drept copii cele mai putin utilizate caractere. Acest nou nod se adauga in lista de caractere avand ca numar de utilizari suma dintre numarul de utilizari ale copiilor. Se realizeaza acest pas pana cand avem un arbore complet.

Avand acest arbore Huffman, putem sa comprimam textul caracter cu caracter. Pentru a avea codul binar pentru fiecare caracter trebuie sa parcurgem arborele de sus in jos adaugand 0 de fiecare data cand facem stanga si 1 cand facem dreapta, pana cand ajungem la caracterul dorit.

Aplicand logica de mai sus, pentru fiecare caracter avem textul comprimat, dar pentru a putea decomprima textul pe viitor mai avem nevoie de arborele Huffman. Pentru a

putea salva arborele trebuie encodat pentru a minimiza memoria utilizata. Encodarea arborelui se poate face parcurgand arborele de sus in jos si adaugand 1 daca nodul nu este frunza si 0 daca nodul este frunza. Dupa fiecare 1 urmeaza 16 biti reprezentand caracterul encodat folosind UTF-16.

Textul final in binar este format din: primii 32 de biti fiind reprezentarea in binar a numarului de biti din care este compus arborele Huffman encodat, urmeaza arborele Huffman encodat, iar la final este textul comprimat.

Decomprimarea

Pentru a decomprima un text comprimat trebuie sa inversam procesul de comprimare.

Citim primii 32 de biti pentru a determina numarul de biti folositi pentru a encoda arborele Huffman.

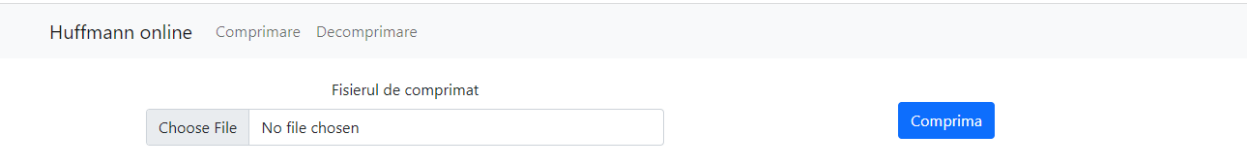
Decodarea arborelui se realizeaza citind textul encodat, bit cu bit, si daca se gaseste bitul 0 nodul nu este frunza si daca se gaseste bitul 1 nodul este frunza, apoi se decodeaza urmatorii 16 biti, folosind metoda UTF-16, pentru a afla caracterul.

Pentru decomprimarea textului se urmareste textul bit cu bit parcurgand arborele Huffman la fel ca la encodare pentru a ajunge la caracterul comprimat. Aplicam aceasta regula pana nu mai avem biti neutilizati si avem textul decomprimat.

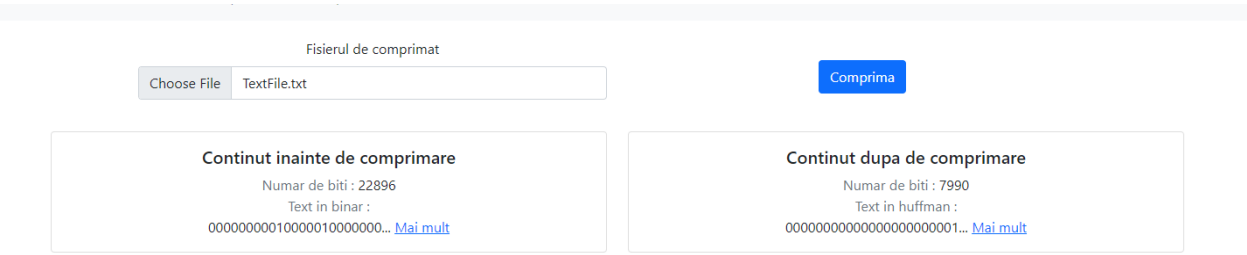
Aplicatia

Comprimarea

Pasul 1: Adaugarea fisierului care urmeaza a fi comprimat

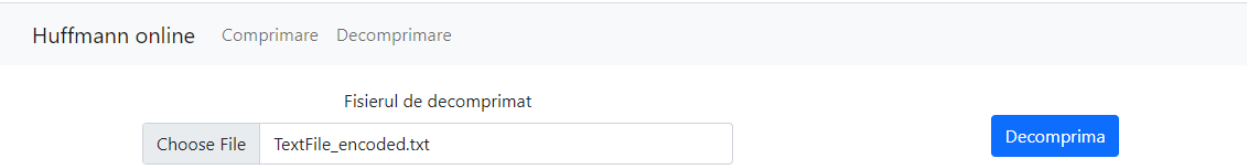


Pasul 2: Comprimarea si salvarea fisierul comprimat

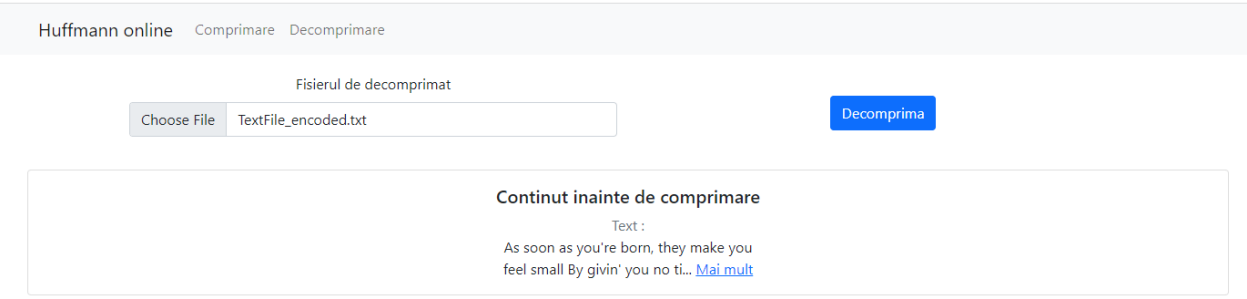


Decomprimarea

Pasul 1: Adaugarea fisierului care urmeaza a fi decomprimat



Pasul 2: Decomprimarea si salvarea fisierul initial



Referinte

- https://www.youtube.com/watch?v=JsTptu56GM8
- http://tau-cs1001-py.wdfiles.com/local--files/lecture-presentations-2014a/lec20.pdf
- https://engineering.purdue.edu/ece264/17au/hw/HW13?alt=huffman#:~:text=To%20store%20the%20tree%20at,tree%2C%20we%20write%20another%200.
- https://stackoverflow.com/questions/759707/efficient-way-of-storing-huffman-tree#:~:text=There%20are%20two%20main%20ways,lengths%20of%20the%20huffman%20codes.