I. GRUPAS APRAKSTS

Grupas nosaukums

POGramētāji

Grupas locekļi

- Tomass Kristiāns Šterns, 5.grupa, 231RDB190
- Petr Gabuniia, 6.grupa, 231RDB331
- Valentīns Koposovs, 10.grupa, 231RDB008

Grupas vadītāja vārds un uzvārds

Tomass Kristiāns Šterns

Problēmu risināšanas metodes

Ja kāds no grupas locekļiem neizpildīs savus pienākumus pārskata izveidē, notiks pārrunas un, iespējams, tiks nolemts, vai šim cilvēkam būs jāuzņemas kādi papilduzdevumi, atskaišu un/vai prezentācijas veidošana. Ja kāds no grupas locekļiem neizpildīs savus pienākumus programmas (*.java) izveidē un/vai prezentācijas izveidē, tiks nolemts, vai šim cilvēkam būs jāuzņemas kādi papilduzdevumi un/vai prezentēšana. Ja grupas vadītājs neiesniedz nepieciešamos failus un/vai nepilda savus pienākumus, tiks nolemts, vai grupas vadītājam tiks atņemts viņa statuss. Kā arī katra grupas locekļa pienākumu nepildīšana var tikt uzskatīts kā iemesls viņa atskaitīšanai no grupas.

II. IZVĒLĒTĀS TEHNOLOĢIJAS

Komunikāciju metodes

Komunikācija notiek klātienē, ja tas ir iespējams, vai sazinoties grupas Whatsapp sarakstē. Atskaites notiek sazinoties Whatsapp-ā, GitHub-ā un klātienē, aprakstot izdarīto.

Izstrādes vides un versiju kontroles sistēmas

Izstrādes vides projektā ir VSCode un IntelliJ IDEA. Grupas projekta versiju kontroli nodrošina GitHub vide, izmantojot saite https://github.com/tomasskristianssterns/dip-107-3-projekts/ , kas būs publiska projekta beigās. Šajā saitē tiek pievienoti visi projekta saistītie faili, izmantojot Git programmu, kur ir piekļuve visiem grupas dalībniekiem.

III. SASPIEŠANAS METOŽU APRAKSTS

Metodes nosaukums nr.1

Huffman Coding

Vispārējas ziņas par algoritmu

Huffman kodēšana ir bezzudumu datu saspiešanas algoritms. Huffman pieder pie *variable-length codes* saspiešanas metožu saimes. Tas vislabāk darbojas uz teksta failiem, bet to tehniski var izmantot uz jebkura tipa failiem.

Huffman ideja ir piešķirt mainīga garuma kodus rakstzīmēm, piešķirto kodu garumi ir balstīti uz atbilstošo rakstzīmju biežumu, kur visbiezākām rakstzīmēm ir īsākie kodi un visretākām – garākie. Mainīga garuma kodi, kas piešķirti ievades rakstzīmēm, ir prefiksu kodi, kas nozīmē, ka kodi (bitu secības) tiek piešķirti tā, ka vienai rakstzīmei piešķirtais kods nav koda prefikss, kas piešķirts kādai citai rakstzīmei. Tādā veidā Huffman Coding nodrošina, ka, dekodējot ģenerēto bitu straumi, nav neskaidrību. Huffman algoritma laikā tiek izveidots binārais koks, pēc kura nosaka katrai rakstzīmei piešķirto kodu, kuru pēc tam lieto, lai aizkodētu tekstu, tādā veidā samazinot faila izmēru. Dekodējot Huffman algoritma kompresēto kodu, ir nepieciešama informācija par rakstzīmēm un tiem piešķirtajiem kodiem.

Informācijas avotu saraksts

https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/

Informācija avotā satur īsi aprakstītu Huffman algoritma aprakstu un detalizēti aprakstītu tā darbību. Avotā ir informācija par Huffman algoritma darbības soļiem – par datiem, kas tiek doti algoritmam, un par to apstrādi, kuras posmā tiek veidots binārais koks. Avotā ir arī pseidokods, koda piemērs C, C++, C#, Python3, Java un Javascript valodā un informācija par algoritma laika un atmiņas sarežģītību, kā arī kuros gadījumos šo algoritmu varētu pielietot un kuros gadījumos tas ir visefektīvākais.

https://en.wikipedia.org/wiki/Huffman_coding

Avotā ir detalizēta informācija par Huffman algoritma aprakstu un vēsturi. Ir detalizēti aprakstīta Huffman algoritma darbība pa soļiem, iekļaujot kompresijas un dekompresijas soļus un minot piemērus. Avotā ir arī informācija par optimilitāti, Huffman algoritma veidiem, kur katrs ir salīdzinoši detalizēti aprakstīts. Ir arī informācija par Huffman algoritma pielietojumiem, kur tas darbojas visefektīvāk un kur to vispār var pielietot. Avotā ir daudz

saites uz citiem avotiem, lai varētu vēl detalizētāk atrast informāciju par kādu konkrētu tekstu, variāciju vai terminoloģiju.

Metodes nosaukums nr.2

PNG

Vispārējas ziņas par algoritmu

PNG (*Portable Network Graphics*) ir rastrgrafikas faila formāts, kas atbalsta bezzudumu datu saspiešanu. Tas paredzēts attēlu saspiešanai, saspiešanai izmanto DEFLATE metodi. PNG fails sākas ar 8 baitu galveni, kas specificē turpmāko faila struktūru. Pēc galvenes teksts ir sadalīts vairākās datgrupās, kas sastāv no 4 daļām: 4 baitu izmēra garums, 4 baitu izmēra datgrupas nosaukums, garuma skaita baitu izmēra datgrupas dati un 4 baitu izmēra cikliskā redundances pārbaude/kontrolsumma.

Faila kompresēšana un dekompresēšana saglabā visu informāciju, nezaudējot liekus datus. Algoritms spēj attēlot dažādu krāsu (melnbaltu, krāsaini u.c.) attēlus, saspiest datus pēc ne pārāk sarežģītiem soļiem. Ir iespējama arī sākotnējo datu filtrēšana, lai atvieglotu saspiešanas algoritma darbību.

Informācijas avotu saraksts

https://en.wikipedia.org/wiki/PNG

Avots satur plašu aprakstu par algoritmu. Ir aprakstīta tā rašanās, vēsture, faila formāts, pikseļu formāts, attēla caurspīdīgums, saspiešana, priekšrocības, salīdzinājums ar citiem failu formātiem, kā arī citām noderīgām tēmām. Faila formāta sadaļā ir detalizēti aprakstīts no kā sastāv faila galvene, kā notiek datu filtrēšana. Priekšrocību sadaļā ir aprakstītas šī algoritma priekšrocības, kas atšķiras no citiem algoritmiem. Salīdzināšanas sadaļā ir aprakstīts kā atšķiras PNG no tādiem failu formātiem kā GIF, JPEG, JPEG-LS un TIFF.

https://medium.com/@duhroach/how-png-works-f1174e3cc7b7

Avots apraksta PNG darbību. Tajā ir aprakstīti algoritma divi soļi: filtrēšana, kur tiek noskaidrots, kā blakus esošie pikseļi savā starpā korelē, un saspiešanas metode, pielietojot DEFLATE metodi. Aprakstīts arī pikseļu saspiešana un tā nozīmīgums, vizualizējot kurās vietās ir vairāk saspiestu pikseļu un kur mazāk. Aprakstīts arī ir faila PNG formāts, līdzīgi kā iepriekšējā avotā, kā arī ir norādīti daži piemēri attēlu saspiešanas izmēri, kas iekļauj un neiekļauj metadatus PNG faila struktūrā.

Metodes nosaukums nr.3

LZ77

Vispārējas ziņas par algoritmu

Jakoba Ziva (Jacob Ziv) un Abrahama Lempela (Abraham Lempel), kuri tās pirmo reizi ierosināja 1977. un 1978. gadā. Šīs shēmas ir izplatītas un bieži izmantotas datu saspiešanas tehnikas, ko lieto modernajā datorzinātnē un datoru datu pārsūtīšanā. LZ77 algoritms izmanto slīdošo logu, kas sastāv no meklēšanas bufera un priekšskatīšanas bufera, lai kodētu sekvenci. Meklēšanas buferī tiek meklēta garākā atbilstība priekšskatīšanas buferī, un tad tiek izveidots triplets <o, l, c>, kur o ir nobīde, l ir atbilstības garums, un c ir simbols no priekšskatīšanas bufera pēc atbilstības. Dekodēšanas procesā šie tripleti tiek atkodēti, un oriģinālais saturs tiek atgūts, izmantojot izvēlēto nobīdi un atbilstošo garumu. Tas ļauj efektīvi saspiest datus, saglabājot tos atjaunojamus.

LZ77 algoritma kodēšanas un atkodēšanas procesi ir cieši saistīti ar priekšskatīšanas un meklēšanas buferiem, kas nodrošina efektīvu datu pārveidi. Meklēšanas buferī tiek veikta ilgākā atbilstība, izmantojot slīdošo logu princips, kas ir atslēga šī algoritma darbībai. Kad atbilstība ir noteikta, tiek izveidots triplets, kas satur informāciju par atbilstības nobīdi, garumu un nākamo simbolu. Šis process tiek atkārtots, līdz priekšskatīšanas buferis ir iztukšots, nodrošinot efektīvu datu saspiešanu.

Tāpat kā citām datu saspiešanas metodēm, LZ77 un LZ78 ļauj iegūt mazāku datu apjomu, nezaudējot būtisko informāciju. Šīs algoritmas ir plaši izmantoti datorzinātnē, tīklu tehnoloģijās un failu pārsūtīšanā, kur efektīva datu pārraide ir būtiska. LZ77 algoritms, jo īpaši, ir ieguvis plašu popularitāti ar tā efektīvo darbību un relatīvi vienkāršo implementāciju, kas padara to par bieži izmantotu risinājumu datu saspiešanā un atkodēšanā.

Informācijas avotu saraksts

LZ77:
https://archive.ph/20130107232302/http://oldwww.rasip.fer.hr/research/compress/algorithms/fund/lz/lz77.html

Avots dod izpratni par LZ77 algoritma izstrādāšanas procesu. Pirmkārt, ir jāizstrādā algoritma principi, piemēram, izmantojot LZ77 metodi, kas meklē garāko sakritību starp jau apstrādāto datu logu un nākamajiem ievades simboliem. Tas ļauj efektīvi kodēt datus, atrodot sakritības un izmantojot pointerus un simbolus. Turklāt, ir jāņem vērā, ka laiks, kas veltīts

kodēšanai, var būt liels, tāpēc algoritmu jāoptimizē, lai samazinātu salīdzinājumu skaitu. Savukārt dekodēšana jāveic vienkārši un ātri, pamatojoties uz iepriekšējo kodēšanas procesu. Lai sasniegtu labu saspiešanas attiecību, var izmantot nelielu atmiņas apjomu, galvenokārt saglabājot tikai logu ar iepriekšējiem dati. Tādējādi, izstrādājot saspiešanas algoritmu, svarīgi ir ņemt vērā šos pamatprincipus, lai nodrošinātu efektīvu un efektīvu datu saspiešanu.

https://cs.stanford.edu/people/eroberts/courses/soco/projects/data-compression/lossless/lz77/index.htm

Šajā tekstā ir detalizēti aprakstīts Lempela-Ziva 77 (LZ77) algoritms, kas izmanto vārdnīcas bāzētu metodi datu saspiešanai. Algoritms izmanto slīdošo logu, lai atrastu un kodētu garākās sakritības starp iepriekšēji kodētu sekvenci un nākamo sekvenci, ko vēlas saspiest. Galvenie koncepti ietver atbilstošu offseta un sakritību garuma noteikšanu, kā arī izmantotās simbola kodu pārsūtīšanu. Šī informācija ir noderīga saspiešanas algoritma izstrādei, jo tā sniedz skaidru priekšstatu par to, kā darbojas vārdnīcas bāzēta saspiešana, un var kalpot kā pamats jaunā algoritma izveidei, pielāgojot to konkrētajiem datiem un prasībām.

IV. IZVĒLĒTĀS METODES APRAKSTS

-Metodes nosaukums

DEFLATE

-Metodes algoritma apraksts

DEFLATE ir bezzudumu datu saspiešanas algoritms, kas izmanto LZ77 un Huffman coding algoritmus. Kompresora algoritms aizvieto vienādus sekojošus simbolus ar datu pāri, kas satur simbolu un to skaitu. Otrajā posmā visbiežāk sastaptie simboli tiek aizvietoti ar īsāko bitu kodu un retāk sastaptie ar garāku bitu kodu. Kompresijas stadijā kompresors, pielietojot zlib/gzip metodi, veic kompresiju atbilstoši vārdu daudzumam. Dekompresors pielieto saspiesto kompresora rezultātu un atgriež oriģinālo failu, izmantojot pretējo algoritmu INFLATE.