Risinājumu algoritmizēšana un programmēšana   
Grupas projekts: Saspiešanas metodes

1. Grupas apraksts

Grupas nosaukums:

*Team1*

Grupas locekļi:

Bohdan Poliakov, Artems Smirnovs, Laura Dubrovska, Una Ivanova, Milena Miloša

Grupas vadītāja vārds un uzvārds:

Ralfs Ulmanis

Problēmu risināšanas metodes:

Paredzamās problemas var sadalīt dažās grupās pēc to dabas:

1. Grupasbiedrs netiek galā ar saviem pienākumiem

Mūsu grupa ir vienojusies, ka pēc pienākumu sadalīšanas nebūs tā, ka katrs pildīs savu un tikai savu daļu. Protams, prioritātē ir savu pienākumu pildīšana, tomēr, ja kādam radīsies grūtības, tad grupasbiedrs par to savlaicīgi paziņos pārējiem grupasbiedriem izmantojot Whatsapp saziņu. Šājā gadījumā grupas vadītājs (Ralfs) nodrošinās vai nu savu personīgo palīdzību grupasbiedram, kas netiek galā ar uzdevumu vai arī pieliks citu grupasbiedru palīdzēt. Sanāk, ka labākajā gadījumā viens otram palīdzēs un skaidros to, kas nebija saprotams.

Apskatot gadījumu, kad grupasbiedrs kādu iemeslu dēļ nav paziņojis par to, ka viņa darbs netiek veikts, tad, lai novērstu tādas situācijas ar problēmu noklusēšanu vai termiņu neievērošanu, lai tas nebūtu pārsteigums beigās, tā ir HR (Milenas) atbildība sekot līdzi grupasbiedru progresam un stāvoklim un ziņot grupas vecākajam, ja ir radušas problēmas.

1. Grupasbiedrs atsakās pildīt uzdevumus, kas viņam tika sākotnēji piedāvāti

Tā kā mūsu grupa ir noskaņota ļoti lojāli un saprotoši, tiks organizēta laipna saruna ar grupasbiedru, tad grupasbiedrs būs lūgts uzņemties citus pienākumus cerībā, ka viņš tiks galā ar kādu citu iespējams viņam patīkamāku uzdevumu.

1. Grupasbiedrs atsakās nest ieguldījumu projektā

Šajā gadījumā, kad jau tika organizēta laipna saruna ar grupasbiedru un tika pārdalīti pienākumi, jo grupasbiedrs netika galā ar piemajiem, grupasbiedrs tiek izslēgts no grupas.

1. Izvēlētās tehnoloģijas

Komunikācijas metodes:

Tika pieņemts lēmums, ka galvenā saziņas metode ir WhatsApp grupa, kas tika savlaicīgi izveidota, pievienojot visus grupas biedrus. Šajā grupā tiks izvietota visa svarīga informācija par projekta uzdevumiem, pienākumu sadalīšanu un tiks izmantota steidzīgo jautājumu risināšanai.

Sekundārā saziņas metode ir Discord 10. grupas mācību serveris, kur tiks apspriesti jautājumi, kas prasa verbālo saziņu.

Katrs projekta posms tiks sākts 7-10 dienas pirms iesniegšanas termiņa. Par posma sākumu tiks paziņots WhatsApp grupā, tajā pašā dienā tiek sadalīti uzdevumi starp grupasbiedriem un tiek paziņots par pirmo termiņu, līdz kuram kaut kāda daļa ir jāizdara. Pirmējais termiņš varētu būt 5 dienas pirms posma iesniegšanas datuma, otrējais termiņš 3 dienas pirms. Pēdējā pārbaude, ja tāda nepieciešama, tiks organizēta dienu pirms iesniegšanas termiņa.

Izstrādes vides un versiju kontroles sistēmas:

Kods tiks izstrādāts izmantojot VSC vai IntelliJ IDEA un tiks augšuplādēts GitHub.com portālā.

1. Saspiešanas metožu apraksts

### **Metodes nosaukums:** LZ77

**Vispārējās ziņas par algoritmu:**

LZ77, kas tika izlaists 1977. gadā, ir daudzu citu bezzudumu saspiešanas algoritmu pamatā. To var atrast ar dažādiem nosaukumiem, piemēram, Lempel Ziv 77 vai LZ1.

LZ77 algoritmu var pielietot teksta, attēlu, audio un video failiem. LZ77 algoritms pieder bezzaudējuma datu saspiešanas metodēm.

**Informācijas avotu saraksts:**

1. https://dzone.com/articles/crunch-time-10-best-compression-algorithms

LZ77 izmanto "sliding window" metodi. Bīdāmā loga saspiešana izmanto meklēšanas atmiņas apgabalu, kuru var pārvietot, kas neļauj algoritmam būt pārāk resursietilpīgam. Šī algoritma mērķis ir kodēt buferī un vārdnīcā atrasto atkārtošanos pēc pozīcijas/izmēra kombinācijas.

LZ77 izmanto vārdnīcu, lai attēlotu:

Nobīde — attālums starp frāzes sākumu un faila sākumu.

Run-length — rakstzīmju skaits, kas veido frāzi.

Atkāpes rakstzīmes — marķieri, kas norāda uz jaunu frāzi. Tas ietver norādi, ka frāze ir vienāda ar sākotnējo frāzi un kuras rakstzīmes atšķiras.

As a file is parsed, the dictionary is dynamically updated to reflect the compressed data contents and size. For example, a file containing the the string "abbadabba" is compressed to a dictionary entry of "abb(0,1,'d')(0,3,'a')".

1. https://habr.com/ru/articles/132683/

Algoritma galvenā ideja ir aizstāt atkārtotu virknes gadījumu ar atsauci uz kādu no iepriekšējām gadījuma pozīcijām. Lai to izdarītu, izmantojiet bīdāmo logu metodi. Bīdāmo logu var uzskatīt par dinamisku datu struktūru, kas ir organizēta, lai atcerētos un nodrošinātu piekļuvi iepriekš “minētajai” informācijai.Tādējādi pats kompresijas kodēšanas process saskaņā ar LZ77 atgādina programmas rakstīšanu, kuras komandas ļauj piekļūt bīdāmā loga elementiem, un saspiestās secības vērtību vietā bīdāmajā logā ievietot atsauces uz šīm vērtībām. Standarta LZ77 algoritmā virkņu atbilstības tiek kodētas kā pāris: match length un offset vai distance.

### **Metodes nosaukums:** LZW

**Vispārējās ziņas par algoritmu:**

LZW (Lempel-Ziv-Welch) ir populārs saspiešanas algoritms, kas strādā ar vairākiem formātiem, tostarp GIF, TIFF, PostScript, PDF, Unix Compress un V.42bis. Tas ir balstīts uz LZ77 un LZ78, metodēm, kuras 1970. gados izstrādāja Ābrahams Lempels un Džeikobs Zivs, un vēlāk Terijs Velčs to pilnveidoja par LZW. Tas efektīvi saspiež atkārtotus datus un dara to ar minimālām skaitļošanas izmaksām.

LZW algoritmu var pielietot teksta, attēlu, audio, video failiem un arī arhīvu failiem.

LZW algoritms pieder bezzaudējuma datu saspiešanas metodēm.

**Informācijas avotu saraksts:**

1. https://kb.iu.edu/d/aghf#:~:text=LZW%20(Lempel-Ziv-Welch,42bis.

Kompresijas process izskatās šādi. Ievades straumes rakstzīmes tiek nolasītas secīgi un tiek pārbaudīts, vai izveidotajā virkņu tabulā šāda virkne pastāv. Ja šāda virkne pastāv, tiek nolasīta nākamā rakstzīme, un, ja virkne neeksistē, straumē tiek ievadīts kods iepriekšējai atrastajai virknei, virkne tiek ievadīta tabulā un meklēšana sākas no jauna. Piemēram, ja tiek saspiesti baitu dati (teksts), tad tabulā būs 256 rindas (no “0” līdz “255”). Ja tiek izmantots 10 bitu kods, tad rindu kodi paliek diapazonā no 256 līdz 1023. Jaunas rindas tabulu veido secīgi, t.i., rindas indeksu var uzskatīt par tās kodu. Dekodēšanas algoritmam kā ievade ir nepieciešams tikai kodēts teksts, jo tas var rekonstruēt atbilstošo tabulu tieši no kodētā teksta. Algoritms ģenerē unikāli dekodējamu kodu, pievienojot virkņu tabulai jaunu virkni katru reizi, kad tiek ģenerēts jauns kods. LZW pastāvīgi pārbauda, vai virkne jau ir zināma, un, ja tā, izvada esošo kodu, neģenerējot jaunu. Tādējādi katra rinda tiks saglabāta vienā eksemplārā, un tai būs savs unikālais numurs.

1. Kodējot (saspiežot) ziņojumu, šis algoritms dinamiski izveido frāžu vārdnīcu: noteiktas rakstzīmju (frāžu) secības ir saistītas ar fiksēta garuma bitu (kodu) grupām (piemēram, 12 bitu, kā ierosināts Velča oriģinālā). pants [1]). Vārdnīca tiek inicializēta ar visām 1 rakstzīmes frāzēm (8 bitu rakstzīmju gadījumā tas ir 256 frāzes). Kodēšanas laikā algoritms skenē teksta rakstzīmi pēc rakstzīmes no kreisās puses uz labo. Kad algoritms nolasa nākamo rakstzīmi, šajā pozīcijā ir virkne W ar maksimālo garumu, kas atbilst kādai vārdnīcas frāzei. Pēc tam šīs frāzes kods tiek nosūtīts uz izvadi, un virkne WK, kur ievades ziņojumā K ir rakstzīme, kas seko W, tiek ievadīta vārdnīcā kā jauna frāze un piešķirts kods. K rakstzīme tiek izmantota kā nākamās frāzes sākums. Dekodēšanas algoritmam kā ievade ir nepieciešams tikai kodēts teksts: atbilstošo frāžu vārdnīcu var viegli izveidot, simulējot kodēšanas algoritma darbību.

### **Metodes nosaukums:** LZSS

**Vispārējās ziņas par algoritmu:**

LZSS (Lempel-Ziv-Storer-Szymanski) ir vārdnīcas saspiešanas algoritms, ko pirmo reizi ierosināja Džeimss A. Storers un Tomass G. Šimanskis 1982. gadā. Šī metode ir bezzudumu un balstās uz esošo LZ77 tehniku.

LZSS algoritms ir pielietojams teksta, attēlu, audio un video failiem. LZSS algoritms pieder bezzaudējuma datu saspiešanas metodēm.

**Informācijas avotu saraksts:**

1. https://moddingwiki.shikadi.net/wiki/LZSS\_compression

LZSS izmanto bīdāmo logu tāpat kā tā priekšgājējs, tomēr atšķiras galvenokārt ar selektīvāku pieeju aizstāšanai. Ja LZ77 vienkārši aizstāj katru apstrādāto virkni, LZSS aizstās virkni ar “length-distance” pāri tikai tad, ja virknes lielums bitos pārsniedz iepriekš iestatīto minimumu, atstājot virknes simbolus kā literāļus, ja ne. Tas neļauj pārim kādreiz aizstāt virkni, kuras izmērs ir mazāks par pašu, un tādējādi tiek novērsta nevajadzīga gala izvades lieluma palielināšana (problēma, kas var rasties LZ77). Kā minēts iepriekš, LZSS arī atceļ koda vārdu komponentu LZ77 trīskāršā "length-distance-codeword", atstājot tikai garumu un attālumu. Algoritms arī ievieš 1 bita karogu, kas katram koda elementam ir prefikss, lai norādītu, vai nākošais elements ir burtisks vai "length-distance" pāris.

1. https://studfile.net/preview/9721928/page:10/

1982. gadā Storer un Szimanski izstrādāja LZSS algoritmu, pamatojoties uz LZ77, kas atšķiras no LZ77 ar tā radītajiem kodiem. LZSS radītais kods sākas ar viena bita prefiksu, kas atšķir faktisko kodu no nekodētās rakstzīmes. Kods sastāv no pāra: nobīde un garums, tāds pats kā LZ77. LZSS logs tiek nobīdīts precīzi par atrastās apakšvirknes garumu vai par 1, ja vārdnīcā nav atrasta neviena apakšvirkne no bufera. Apakšvirknes garums LZSS vienmēr ir lielāks par nulli, tāpēc apakšvirknes garuma binārais garums ir bufera garuma binārais logaritms, kas noapaļots līdz lielākam veselam skaitlim. Koda garumu aprēķina šādi: apakšvirknes garums nevar būt lielāks par bufera lielumu, un nobīde nedrīkst būt lielāka par vārdnīcas lielumu -1. Tāpēc nobīdes binārais garums tiks noapaļots uz augšu n=log2 (vārdnīcas lielums), un apakšvirknes garuma binārais garums tiks noapaļots uz augšu m=log2 (bufera izmērs). Katra rakstzīme ir kodēta ar 8 bitiem (piemēram, ASCII+). Tas nozīmē, ka ir nepieciešami n+m+8 biti, lai kodētu katru sākotnējā ziņojuma apakšvirkni.