# Risinājumu algoritmizēšana un programmēšana

Grupas projekts: Saspiešanas metodes

## Posms: “izpildītāju grupas organizācija un saspiešanas metodes izvēle un apraksts”

### Uzdevums:

Sagatavot un augšupielādēt ORTUSā pārskatu, kas satur ziņas par izveidotu grupu un projekta izstrādes plāniem. Pārskatam ir jābūt pdf formātā un jāsatur sekojošas nodaļas:

### Grupas apraksts:

* Grupas nosaukums:
  + Ultra Mega Programmers
* Grupas locekļi:
  + Oto Ģigulis - 211RDB013
  + Jānis Aveniņš - 211RDB020
  + Līva Puķīte - 211RDB036
  + Anastasija Šarakova - 211RDB093
* Grupas vadītāja vārds un uzvārds:
  + Jānis Aveniņš
* Problēmu risināšanas metodes:
  + Darba uzdevumi tiks pārdalīti un kolēģis tiks izslēgts no grupas darba (Kolēģa iepriekšēji izpildītie uzdevumi tiks atstāti noteiktā kolēģa vārdā).

### Izvēlētās tehnoloģijas:

* Komunikācijas metodes:
  + WhatsApp (vismaz divas reizes nedēļā)
  + Discord (problēmu risināšanas laikā)
* Izstrādes vides un versiju kontroles sistēmas
  + Repl.it

## Saspiešanas metožu apraksts:

* Deflate [Oto Ģigulis]
  + Vispārējās ziņas par algoritmu:
    - Bezzudumu saspiešanas algoritms
    - Deflate algoritms ir LZ77 Algoritma un Huffman algoritma apvienojums, kuri pieder attiecīgi “Dictionary” un “Statiskajai” saspiešanas metožu saimei
    - Algoritmu var pielietot jebkāda veida failu saspiešanai
  + Informācijas avotu saraksts:
    - https://www.zlib.net/feldspar.html

Šajā rakstā ir vienkārši izskaidrota deflate metodes darbība, izskaidrojot, to atsevišķās metodes kas tiek apvienotas, t.i. LZ77 un Huffman coding metodes. Huffman coding metode sakārto datus tā, ka visbiežākās iespējamās datu vērtības tiek apzīmētas ar īsāko kodu(ja tiek izmantots koka attēlojums, biežākās vērtības atrodas “augstāk” kokā. LZ77 metode izmanto slīdošā loga principu, kas nozīmē, ka katrā vietā noteiktajā failā ir informācija par to kādas vērtības bija iepriekš. To zinot, saspiešanas algoritms var nepārrakstīt šīs vērtības, bet gan atzīmēt, cik sen šīs pašas vērtības tika redzētas un cik daudz vērtību garumā tās sakrīt. Beigās deflate algoritmam var būt 3 dažādas iespējas: nesaspiest datus(ja tas nedod nekādu guvumu), vispirms izmantot LZ77 metodi un tad pielietot jau sagatavotu Huffman coding koku vai arī izmantot vispirms LZ77 metodi un tad Huffman coding koku, kas tiek izveidots atbilstoši nolasītajiem datiem.

* + - https://www.euccas.me/zlib/

Šajā rakstā aprakstīta zlib saspiešanas programma, kas izmanto deflate algoritmu. Tajā par Huffman coding izskaidrots, kāpēc katram simbolam nepieciešams unikāls prefiksa kods. Tas ir paredzēts, lai nebūtu nekādu pārpratumu dekodējot datus, teiksim, ja burtam a izmanto kodu “0”, tad burtam b nedrīkst izmantot kodu “01” , jo tas var tikt interpretēts gan kā kods, kas satur 2 vērtības ar kodu “0” un “1” vai arī 1 vērtību ar kodu “01”, kas traucē datu nepārprotamībai.

* Lempel – Ziv - Markov chain Algorithm (LZMA) [Jānis Aveniņš]
  + Vispārējās ziņas par algoritmu:
    - Bezzudumu saspiešanas algoritms
    - Algoritms pieder “Dictionary” saspiešanas metožu saimei
    - Algoritmu var pielietot jebkāda veida failu saspiešanai
  + Informācijas avotu saraksts:
    - https://en.wikipedia.org/wiki/Lempel%E2%80%93Ziv%E2%80%93Markov\_chain\_algorithm:

Kopš 1996. vai 1998. gada veidotais algoritms ir līdzīgs LZ77 algoritmam, kas tika publicēts 1977. gadā. Šim algoritmam ir liela saspiešanas spēja, lai gan uztur oriģinālo failu atgriešanas ātrumu līdzīgu citiem algoritmiem. Pirms šī algoritma pārsvarā visi saspiešanas modeļi tika balstīti uz baitiem, bet LZMA galvenā inovācija ir tas, ka tā vietā tiek izmantoti bitu laukumi. Šis ir gandrīz tik pat prasti kā parastā bitu balstītā modelī, bet šis dod iespēju labāk saspiest failus, jo šādi izvairās no vairāku atšķirīgu bitu saplūšanas.

* + - lifewire.com/lzma-file-2621951:

Saitē ir izskaidroti .lzma faili. Galvenokārt šie faili ir redzami Unix balstītās operatorsistēmās. Tiek izskaidrota atšķirība starp LZMA un LZMA2. LZMA2 algoritms izmanto gan LZMA, gan ne-saspiestus datus. LZMA ir pilnīgi pietiekams arhīviem zem 256 MB, bet, ja ir nepieciešams saspiest lielākus failus vai failus, kas jau ir saspiesti, tad LZMA2 algoritms var nodrošināt ātrāku un labāku saspiešanu. Šim algoritmam ir nepieciešams lielāks operatīvās atmiņas daudzums. Līdzīgi saspiešanas algoritmi ir LZ77 un LZ78, uz kuriem ir balstīts pats LZMA.

* LZ77 [Līva Puķīte]
  + Vispārējās ziņas par algoritmu:
    - Bezzudumu sasppiešanas algoritms
    - Algoritms pieder “Dictionary” saspiešanas metožu saimei
    - Algoritmu var pielietot failiem ar dažādām entropijām
    - Algoritmu var pārveidot par “run-length encoding” metodes algoritmu
  + Informācijas avotu saraksts:
    - https://docs.microsoft.com/en-us/openspecs/windows\_protocols/ms-wusp/fb98aa28-5cd7-407f-8869-a6cef1ff1ccb:

Vietne, kurā ir izskaidrota visa terminoloģija saistībā ar algoritmu. Sākumā, protams, arī aprakstīts pats algoritms un kādā veidā mēs ar to strādājam. Šeit var atrast arī piecus vienkāršus soļus, kas palīdz programmētājiem viegli uzrakstīt kodu šim algoritmam. Tāpat šajā mājaslapā var izlasīt kā notiek pats saspiešanas process, un papildus ir ievietotas tabulas, kurās ir piemēri un septiņi soļi kā tiek saspiesta datu plūsma. Līdzīgas tabulas ir izveidotas arī, lai atrādītu kā darbojas “decompression” process.

* + - http://www.stringology.org/DataCompression/lz77/index\_en.html:

Šajā vietnē ir aprakstīta īsa vēsture par algoritmu – autori, gads un metode kādā algoritms tiek izpildīts(slīdoša loga princips). Mājaslapā var arī lasīt aprakstu un secīgus soļus kā notiek datu saspiešana, kā tiek meklēti simboli un ar kādu pierakstu tie tiek saglabāti atmiņā. Sākumā ir rakstīts kā “logu” sadala divos buferos, no kuriem viens, kas ir meklēšanas buferis parasti ir 8192 bitu liels un otrs buferis 10-20 bitu lielumā. Pēc tam var aplūkot arī pseidokodu, kurš izpilda šo LZ77 algoritmu.

* LZSS [Anastasija Šarakova]
  + Vispārējās ziņas par algoritmu:
    - Bezzudumu sasppiešanas algoritms
    - Algoritms pieder “Dictionary” saspiešanas metožu saimei
    - Šī metode mēģina aizstāt simbolu virkni ar atsauci uz tās pašas virknes vietu vārdnīcā. Atšķirība starp LZ77 un LZSS ir tas, ka LZ77 vārdnīcas atsauce var būt garāka par virkni, kuru tā aizstāja
  + Informācijas avotu saraksts:
    - https://go-compression.github.io/algorithms/lzss/:

Šajā rakstā ir parādīti piemēri kā, izmantojot LZSS algoritmu, var implementēt kodētāju. Rakstā tiek analizēts kā strādā kodētājs: Kodētājs strādā character by character, meklēšanas buferis ir kodētāja atmiņa, katru simbolu kuru kodē, pievieno meklēšanas buferim tā lai tas to “atceras”. Process turpinās, kodējot tokens un pievienojot tekstu meklēšanas buferim pa simbolam, līdz tas ir pabeidzis visu kodēšanu. Kā arī rakstā ir piemēri kā var implementēt dekodētāju. Secinājumos tiek parādīti LZSS saspiešanas algoritma rezultāti: LZSS samazināja faila izmēru par 45%.

* + - https://ieeexplore.ieee.org/document/6413553:s

Šī raksta ievadā stāsta kur var tikt pielietots LZSS algoritms: var izmantot, lai samazinātu krātuves prasības, kā arī uzlabotu komunikācijas veiktspēju, samazinot datu apjomu, kas jāpārsūta pa I/O kanālu. Rakstā tiek aprakstīti testi, ar kuriem pārbaudīja algoritma veiktspēju un nodaļā rezultāti un analīze tiek attēloti un aprakstīti rezultāti. Šī raksta autori izmantoja LZSS algoritmu, lai uzlabotu GPU veiktspēju, kas rezultāta tika uzlabots par 46,65%. Kā arī parādīja dažādus LZSS algoritma pielietojumus.

### Izvēlētās metodes apraksts:

* Deflate
* Metodes algoritma apraksts:
  + Aprakstīt vispārējā veidā kompresora un dekompresora algoritmus izmantojot dabīgo valodu, pseidokodu, vai grafisko algoritma shēmu.