Testausdokumentaatio:
Tietorakenteet ja algortimit harjoitustyö.

Kaikki testit on toteutettu Junit-testeinä ja ne voi suorittaa ajamalla projektin juuressa sijaitsevalla runtests.sh -skriptillä. Joskus pakkaajan ja purkajan testit saattavat kaatua, keskeytä tällöin testit ja koeta ajaa uudelleen.

Tässä dokumentissa käytettävästä termistöstä:

tavuleveys: (dataikkunan leveys) data pilkotaan tavuihin, tämä numero kertoo kuinka monta tavua yhdessä yksikössä on.

data: informaatio joka on taulussa.

monitavu: (MultiByte) yksikkö johon tallennetaan tavuleveyden verran dataa.

Jäännös: raakadataa luettaessa aina tavuleveyden verran saattaa viimeisten tavujen kohdalla jäädä tavuleveys vajaaksi, tätä alle tavuleveyden mittaista tavustoa lopussa kutsutaan jäännökseksi

pakkaaja: ohjelmakoodissa MultiByteEncoder

purkaja: ohjalmakoodissa MultiByteDecoder

prefix: pakkaajan luoma tavumuotoinen esikenttä, joka kertoo kuinka monta saman tavuleveyden omaavaa avainta seuraa, vaihtoehtoisesti kuinka monta tavuleveyttä pakkaamatonta dataa seuraa

status: määrämuotoinen (enumerator) arvo, jota pakkaaja ja purkaja käyttävät kertomaan sisäisestä tilasta;
pakkaajan statukset: NULL, DECODING, INTERRUPTED, DATAERROR, DONE;
purkajan statukset: NULL, BUILDING, ENCODING, INTERRUPTED, DATAERROR, DONE.

Dictionary: MultiByteHashedTableTest
setup: kullekin testille tehtiin setup jossa luotiin neljä tavutaulukkoa joissa kussakin eri data, riippuen testin tarkoituksesta:

incrementalData: sisältää tavuja numeraalisesti nousevassa järjestyksessä
sameData: sisältää vain yhtä tavu numeraalia
sortinData: sisältää viittä (5) numeraalisesti eriävää tavudataa tietyillä esiintymistaajuuksilla
randomData: sisältää 200 000 tavua pseudo-satunnaista dataa joka generoidaan javan sisäänrakennetulla random-funktiolla

-contains-metodi:

Käyttää incremental- ja sameData taulukoiden tietoja. Kullekin tehtiin seuraavaa:
Luotiin uusi hajautustaulu (MultiByteHashedTable) tavuleveyksillä 2 – 4.
kullakin tavuleveydellä lisätiin hajautustauluun dataa tavuleveyden verran niin kauan kun dataa riitti taulussa.
Kullakin tavuleveydellä testattiin, että hajautustaulu sisältää edellä lisätyn datan.

-ylimääräisen tiedon vapautus:

Käyttää randomData-taulukon tietoja.

Täytettiin hajautustaulu datalla käyttäen tavuleveyttä kaksi (2).
Luettiin hajautustaulun avainten määrä muistiin
suoritettiin hajautustaululle operaatio purgeAndClean
Luettiin hajaututstulun avainten määrä operaation jälkeen
Testattiin onko avainten määrä vähentynyt.

-hajautustaulun järjestäminen

Käyttää sortingData-taulukon tietoja

Täytettiin hajautustaulu käyttäen tavuleveyttä neljä (4).
Luettiin hajautustaululta järjestetty taulu monitavuja
Muodostettiin manuaalisesti oikein järjestetty taulukko monitavuista, jotka edustavat sortinData-taulun yksittäisiä monitavuja

Testattiin hajautustaululta saadulle taulukolle, että sen paikassa *i* on monitavu joka vastaa manuaalisesti järjestetyssä taulussa *i* olevaa monitavua

Entities: MultiByteTest (monitavu):
setup: Muodostettiin kolme tavudataa sisältävää talukkoa:
incrementalData: numeraalisesti nousevassa järjestyksessä olevaa tavudataa.
sameData: numeraalisesti samaa tavudataa sisältävä taulukko.
randomData: 2000 000 tavua sisältävä taulukko jonka data on muodostettu javan sisäänrakennetulla pseudo-satunnaisfunktiolla.

-satunnaisen datan hajautusavaimet:
Luotiin tavuleveyksille 2-4 kokonaislukutaulukot
kuhunkin taulukkoon laitettiin kyseisen tavuleveyden monitavun hajautusavain
testattiin iteratiivisesti taulukko läpi käymällä onko taulukossa samoja hajautusavaimia.
Huomio: tämä testi ei suunnitellusti mene koskaan läpi, ja se on käytössä vain jos sen erityisesti haluaa testata.

-pienen data-setin hajautusavainten testaus:
Luotiin tavuleveyksille 2-4 kokonaislukutaulukot
kuhinkin taulukkoon lisättiin sen tavuleveyden monitavujen hajautusavaimet
testattiin, ettei taulukossa ole yhtäkään samaa hajautusavainta.

-saman datan hajautusavainten testaus:
Luotiin tavuleveyksille 2-4 kokonaislukutaulukot
kuhunkin taulukkoon lisättiin sen tavuleveyden monitavujen hajautusavaimet
testattiin, että jokainen hajautusavain on samanlainen kussakin taulukossa.

-datan ylivuodon testaus:
iteratiivisenen testaus tavuleveyksille 2-10.
kullekin tavuleveydelle muodostettiin uusi monitavu
monitavulle lisättiin iteratiivisesti dataa 100 tavuun asti
kullakin lisäyksellä testattiin, että dataa ei sallittu lisäävän yli monitavun kapasiteetin

-equals-metodi:
Käyttää sameData-taulukon dataa.
iteratiivinen testaus tavuleveyksille 2-10
kullekin tavuleveydelle muodostettiin monitavu johon lisättiin sen kapasiteetin verran dataa
käytiin taulukon data läpi iteratiivisesti muodostaen aina uusi monitavu indeksin osoittamasta kohdasta
testattiin, että muodostetut monitavut olivat yhtäläisiä, eli niiden sisältämä data on sama.

Utilities: ArrayUtilitiesTest
setup: kullekin testille alustettiin javan sisäänrakennettu pseudo-satunnaisgeneraattori alkuluvulla.

-monitavutaulukon kutistus:
Luotiin 90-paikkainen monitavutaulukko.
Kutsuttiin ArrayUtilities-luokan staattista metodia contractMultiByteArray, arvolla 10
Testattiin, että paluuarvona saatu monitavutaulukko oli pituudeltaan 10.

-tavudatan kopioituminen:
Luotiin manuaalisesti tavudataa sisältävä taulukko
Luotiin uusi tyhjä taulukko joka oli pitempi kuin edellinen.
Kutsuttiin ArrayUtilities-luokan staattista metodia encodeIntoArray em taulukoilla indeksillä nolla (0)
testattiin, että uudessa, laajemmassa, taulukossa oli täsmälleen samat tavudatat alkaen indeksistä nolla (0), päättyen pienemmän taulukon pituuteen.

-tavutaulukon laajentaminen:
Luotiin uusi tavutaulukko jossa 10 paikkaa

Luotiin em taulukkoon pseudo-satunnaista dataa
Kutsuttiin ArrayUtilities-luokan staattista metodia expandByteArray
Otettiin talteen metodin paluuarvona saatu tavutaulukko
Testattiin, että uuden tavutaulukon pituus oli kaksinkertainen alkuperäiseen nähden
Testattiin iteratiivisesti, että uudessa taulukossa on täsmälleen sama data kuin alkuperäisessä

-monitavutaulukon laajentaminen:
Luotiin uusi monitavutaulukko
Kutsuttiin ArrayUtilities-luokan staattista metodia expandMultiByteArray
Testattiin, että metodin paluuarvona saatu taulukko oli kaksi kertaa alkuperäistä pitempi

-monitavun luominen:
Luotiin uusi tavutaulukko joka sisälti ennalta määrättyjä tavuja
Kutsuttiin ArrayUtilities-luokan staattista metodia makeMultiByte seuraavilla arvoilla:
alkaen indeksistä 0, em tavutaulukosta, tavuleveydellä kolme (3)
Testatiin iteratiivisesti, että paluuarvona saadun monitavun sisältämä data oli sama ja samassa järjestyksessä kuin taulukossa, josta se luotiin

-tavutaulukon nollien poistaminen indeksin perusteella:
Luotiin tavutaulukko jossa 80 paikkaa
luotiin taulukkoon pseudo-satunnaista tavudataa
Kutsuttiin ArrayUtilities-luokan staattista metodia removeTrailingZeros em taulukolla indeksillä 8
Testattiin, että paluuarvona saatu taulukko oli kahdeksan paikkainen
Testattiin iteratiivisesti, että paluuarvona saadun taulukon data oli samaa kuin alkuperäisessä taulukossa niillä paikoilla jotka olivat jäljellä

Utilities: IntegerConverterTest

-kokonaisluvun muunto yhden tavun mittaiseksi tavuksi
Kutsuttiin IntegerConverter-luokan staattista metodia IntegerToByte arvoilla 1, 1
Kutsuttiin IntegerConverter-luokan staattista metodia ByteToInteger em metodin paluuarvolla
testattiin, että kohdan 2 paluuarvo oli 1
testattiin, että kohdan 1 paluuarvon pituus oli 1 (yhden tavun mittainen)

-kokonaisluvun muunto kahden tavun mittaiseksi tavuksi
Kutsuttiin IntegerConverter-luokan staattista metodia IntegerToByte arvoilla 300, 2
Kutsuttiin IntegerConverter-luokan staattista metodia ByteToInteger em metodin paluuarvolla
testattiin, että kohdan 2 paluuarvo oli 300
testattiin, että kohdan 1 paluuarvon pituus oli 2 (kahden tavun mittainen)

-kokonaisluvun muunto monen tavun mittaiseksi tavuksi
Kutsuttiin IntegerConverter-luokan staattista metodia IntegerToByte arvoilla 90 000, 3
Kutsuttiin IntegerConverter-luokan staattista metodia ByteToInteger em metodin paluuarvolla
testattiin, että kohdan 2 paluuarvo oli 90 000
testattiin, että kohdan 1 paluuarvon pituus oli 3 (kolmen tavun mittainen)

-tavun muuntaminen kokonaisluvuksi
Luotiin tavutaulukko ennaltamäärätyillä tavuilla
Kutsuttiin IntegerConverter-luokan staattista metodia ByteToInteger ko taulukolla
Kutsuttiin IntegerConverter-luokan staattista metodia IntegerToByte em metodin paluuarvolla
testattiin, että kohdan 2 paluuarvona saatu tavutaulukko sisälsi täsmälleen samat tavut samassa järjestyksessä kuin alkuperäinen taulukko

Encoding: MultiByteDecorerTest
Tämän luokan testit testaavat myös pakkaajan toiminnan oikeellisuutta, siksi samoja testejä ei ole kirjoitettu pakkaajan testeiksi. Testeissä käytetään neljää erityyppistä dataa, kaikki taulukoituja määrämuotoisia tavuja:
encodedDataType1: simuloi pakattua dataa

encodingDataType1: simuloi pakkaamatonta dataa, ensimmäinen monitavu raakadataksi
testataa, että ensimmäine prefix muodostuu oikein (ja tätä seuraavat sen mukaan) kun ensimmäinen tavuleveys on pakkaamatonta dataa ja sitä seuraa pakattua dataa.

encodingDataType2: simuloi pakkaamatonta dataa, ensimmäiset monitavut pakatuiksi
testaa, että ensimmäinen prefix muodostuu oikein (ja tätä seuraavat) kun ensimmäiseen prefixiin tulee maksimimäärä avaimen tavuleveyttä (9 kpl)

encodingDataType3: simuloi pakkaamatonta dataa, ensimmäiset monitavut raakadataksi
testaa, että ensimmäinen prefix muodostuu oikein (ja tätä seuraavat) kun ensimmäiseen prefixiin tulee merkitä useamman tavuleveyden verran pakkaamatonta dataa

encodingDataType4: simuloi pakkaamatonta dataa, jossa viimeiset tavut jäännökseksi
testaa, että jäännös koodataan oikein ennen tasan tavuleveyden mittaista dataa.

-manuaalisesti luodun pakatun simulaatiodatan purku
alustettiin simuloitu pakattu data
luotiin uusi instanssi purkajasta
luotiin uusi säie johon kapseloitiin juuri luotu purkaja
testattiin, että purkajan status on NULL, eli mitään operaatioita ei ole aloitettu
käynnistettiin säie

kutsuttiin säikeen metodia isAlive
testattiin iteratiivisesti onko säie elossa niin kauan kunnes purkajan tila muuttui statukseksi DONE
kutsuttiin purkajan metodia getDecodedData
testattiin paluuarvon indeksi nolla (0), että se sisältää numeraalisen arvon 2.

-dekooderin keskeytyksen testaus
alustettiin simuloitu pakattu data
luotiin instanssi purkajasta pakatulla datalla
luotiin uusi säie johon purkaja kapseloitiin
käynnistettiin säie
kutsuttiin purkajan metodia interrupt
pysäytettiin testisäie ajaksi 100 ms
testattiin, että purkajan status oli INTERRUPTED

-määrämuotoisen pakkaamattoman simulaatoidatan pakkaus ja purku, tyyppi 1
alustettiin encodingDataType1
luotiin uusi pakkaaja em datalla
luotiin uusi säie, johon pakkaaja kapseloitiin
käynnistettiin säie
odotettiin kunnes pakkaajan status oli DONE
tallennettiin pakkaajan metodin getCombinedDataAndKeys kutsusta palautunut tavudata

luotiin uusi purkaja
luotiin uusi säie, johon purkaja kapseloitiin
käynnistettiin purkaja-säie
odotettiin kunnes purkajan status oli DONE
tallennettiin purkajan metodin getDecodedData kutsusta palautunut tavudata
testattiin, että palautunut tavudata oli saman pituinen kuin alkuperäinen data
testattiin iteratiivisesti, että palautettu data oli täsmälleen sama kuin alkuperäinen data

-määrämuotoisen pakkaamattoman simulaatoidatan pakkaus ja purku, tyyppi 2
käytännössä sama kuin tyypin 1 testi, mutta datana toimi encodingDataType2

-määrämuotoisen pakkaamattoman simulaatoidatan pakkaus ja purku, tyyppi 3
käytännössä sama kuin tyypin 1 testi, mutta datana toimi encodingDataType3

-määrämuotoisen pakkaamattoman simulaatoidatan pakkaus ja purku, tyyppi 4
käytännössä sama kuin tyypin 1 testi, mutta datana toimi encodingDataType4

Encoding: MultiByteEncoderTest
Testeissä käytetään viittä erilaista tavumuotoista dataa:
encodingDataType1 – 4: samat kuin pakkaajan testeissä
randomData: simuloi pseudo-satunnaista dataa

-avainten lukumäärä, data ja yhdistetty pakattu data
alustettiin encodingDataType1
luotiin instanssi pakkaajasta em datalla, tavuleveydellä 4
luotiin säie johon pakkaaja kapseloitiin
testattiin, että pakkaajan status oli NULL
käynnistettiin säie
odotettiin kunnes pakkaajan status oli DONE
tallennettiin pakkaajan metodin getEncodedKey kutsusta paluuarvona saatu data (avaimet)
testattiin, että avainten lukumäärä oli suurempi kuin 0
tallennettiin pakkaajan metodin getEncodedData kutsusta paluuarvona saatu data (pakattu data)
testattiin, että pakatun datan koko oli suurempi kuin 0
tallennettiin pakkaajan metodin getCombinedKeysAndData kutsusta paluuarvona saatu data
(yhdistetyt avaimet ja data)
testattiin, että yhdisteen koko oli suurempi kuin 0

-yksinkertainen datan oikeellisuuden testaus
alustettiin encodingDataType1
luotiin uusi pakkaaja em datalla, tavuleveydellä 4
luotiin säie johon pakkaaja kapseloitiin
käynnistettiin säie
odotettiin kunnes pakkaajan status oli DONE
tallennettiin pakkaajan metodin getEncodedKeys kutsusta paluuarvona saatu data (avaimet)
testattiin, että avainten lukumäärä oli 16 (neljä (4) avainta, neljä (4) tavua/avain)
tallennettiin pakkaajan metodin getEncodedData kutsusta paluuarvona saatu data (pakattu data)
testattiin, että pakatun datan koko oli pienempi kuin alkuperäinen data.

-pakkaajan keskeytyksen testaus
alustettiin satunnaisdata
luotiin instanssi pakkaajasta em datalla, tavuleveydellä 4
luotiin säie johon pakkaaja kapseloitiin
käynnistettiin säie
tallennettiin pakkaajan status
kun pakkaajan status oli BUILDING tai ENCODING kutsuttiin pakkaajan metodia interrupt
testattiin, että säie ei ollut käynnissä

-pakatun datan oikeellisuuden testit
pakatun datan oikeellisuus testattiin neljällä (4) erityyppisellä datalla: encodingDataType1 – 4. Kaikki neljä testiä on manuaalisesti kirjoitettu ja ne testaavat, että pakattu data muodostuu määrämuotoiseksi.
Menetelmä:
alustettiin data
luotiin pakkaaja datalla, tavuleveydellä 4
luotiin säie johon pakkaaja kapseloitiin
käynnistettiin säie
odotettiin kunnes pakkaajan status oli DONE
tallennettiin pakkaajan metodin getConbinedDataAndKeys kutsusta paluuarvona saatu data

testattiin, että määrätyissä paikoissa pakatussa datassa oli tietyt ennalta lasketut arvot

-ristiintestaus purkajan kanssa
alustettiin encodingDataType1
luotiin pakkaaja datalla, tavuleveydellä 3
luotiin säie johon pakkaaja kapseloitiin
käynnistettiin säie
odotettiin kunnes pakkaajan status oli DONE
tallennettiin pakkaajan metodin getCombinedDataAndKeys kutstusta paluuarvona saatu data

luotiin purkaja pakatulla datalla
luotiin säie johon purkaja kapseloitiin
käynnistettiin purkajan säie
odotettiin kunnes purkajan status oli DONE
tallennettiin purkajan metodin getDecodedData kutsusta paluuarvona saatu purettu data
testattiin iteratiivisesti, että purettu data oli täsmälleen sama kuin alkuperäinen data.