Toteutusdokumentti

Ohjelman yleisrakenne

Ohjelma koostuu java-koodista sekä sanasto-tiedostosta, joka annetaan xml-muotoisena. Ohjelma suoritetaan ajamalla crossword.crosswordmaker-pakkauksessa oleva Main-metodi. Käyttöliittymän avulla käyttäjä voi valita generoitavan krypton koon sekä aloitussanan. Ohjelman koodi on jaettu kolmeen pakkaukseen:

- crossword.crosswordmaker
    - Pakkauksessa on Main-metodin lisäksi käyttöliittymä.
- crossword.datastructures
    - Itse toteutetut tietorakenteet CustomArrayList ja SearchTree ovat täällä.
- crossword.lexicon
    - Xml-muotoisen sanasto-tiedoston käsittely tapahtuu täällä.
- crossword.logic
    - Kaikki krypton generointiin tarvittava koodi löytyy tästä paketista.

Yksikkötestit ovat vastaavissa pakkauksissa.

Pakkauksessa crossword.logic on seuraavat luokat:

- BoardOfWords
    - huolehtii kryptolaudan piirtämisestä sekä sanojen piirtämisestä laudalle
- CrossWordMaker
    - jos aloitussanaa ei anneta, hakee satunnaisen sanan aloitussanaksi
    - kutsuu kaikkia niitä metodeita, joita tarvitaan krypton generointiin
- WordFinder
    - Sanojen etsiminen kryptolaudalle eli ohjelman varsinainen ydin on tässä luokassa
- WordPosition
- WordPositionFinder
    - Etsii valmiiksi piirretyltä kryptolaudalta ne paikat, joihin tulee löytää sana

Saavutetut aika- ja tilavaativuudet

Ohjelman ydin on metodi, joka etsii jokaiseen krypton sanapaikkaan sanan. Tämä on toteutettu rekursion avulla niin, että jokaiseen sanapaikkaan kokeillaan jokaista siihen sopivaa sanaa (varatut kirjaimet huomioiden) kunnes päädytään lopputulokseen, jossa krypton jokainen sanapaikka on täytetty. Pseudokoodina:

positions on globaali muuttuja

```
layWords(index, boardOfWords)
  if index == positions.size
    return boardOfWords
```

```
  position = positions[index]
  fittingWords = findWords(position)
  for each word in fittingWords
    copyOfBoard = copy(boardOfWords)
    copyOfBoard.drawWord(word)
    solution = layWords(index + 1, copyOfBoard)
    if (solution ≠ null) return solution
  return null

findWords(position)
  mask = makeMask(position)
  fittingWords = new ArrayList
  return searchWordTree(mask, root of searchTree, 0, fittingWords)

searchWordTree(mask, node, level, fittingWords)
   if mask.charAt(level) == node.getKey
    if level == maskin pituus
      if nodeen on talletettu sana
        add sana to fittingwords
      return fittingWords
    childNode = node.getChild
    while childnode NOT null
      searchWordTree(mask, childNode, level + 1, fittingwords)
      childNode = childNode.getNext
  return fittingWords
```

Koska kryptolauta kopioidaan jokaisessa rekursiokutsussa, muistissa voi kerrallaan olla puun syvyyden verran kryptolautoja eli niin monta kuin laudalla on sanapaikkoja. Jos merkitään, että s = sanapaikkojen määrä, l = kryptolaudan leveys ja k = laudan korkeus, niin tilavaativuus on $O(s \cdot l \cdot k)$. Kryptolaudan lisäksi rekursiokutsuissa kuljetetaan mukana listaa laudalle jo piirretyistä sanoista, minkä tilavaativuus on $O(s!)$.

Aikavaativuus on hyvin suuri. Aikaa kuluu pääasiassa sopivien sanojen etsimiseen hakupuusta sekä sopivien sanojen sovittamiseen laudalle.

Sopivien sanojen hakeminen hakupuusta: Hakupuun solmulla voi olla korkeintaan suomalaisten aakkosten verran eli 29 solmua. Jos maskissa ei ole kirjaimia rajoittamassa haun laajuutta, hakupuussa mennään syvyyssuunnassa korkeintaan maskin pituuden verran solmuja, eli pahimmassa tapauksessa käydään läpi 29^p solmua, missä p = sanan pituus. Käytännössä pahin tapaus ei koskaan toteudu, koska kaikki kirjaimet eivät esiinny peräkkäin.

Sanojen sovittaminen laudalle: Jos jokaiseen sanapaikkaan pystyisi sijoittamaan n määrän sanoja ja kryptoon sijoitettavien sanojen määrää merkittäisiin s:llä, aikavaativuus olisi n^s, mutta käytännössä n pienenee hyvin nopeasti laudalle jo asetettujen sanojen rajoittaessa seuraavaksi lisättävien mahdollisten sanojen määrää. Lisäksi jokaisessa rekursiokutsussa

käytettyjen sanojen lista (usedWords) käydään läpi ja siinä olevat sanat etsitään listalta fittingWords. Alussa fittingWords sisältää paljon sanoja, mutta usedWords vain vähän. Pidemmälle mentäessä fittingWords-lista lyhenee ja usedWords-lista kasvaa.