Algolabra, kevät 2025, määrittelydokumentti
Toteuttaja: Pekka Linna
Opinto-ohjelma: Tietojenkäsittelytieteen kandidaatti (TKT)

Harjoitustyön aihe ja kieli

Harjoitustyön aiheena on yksinkertainen SAT solveri DPLL-algoritmilla, jossa toteutettuna yksikköpropagaatio ja puhtaan literaalin poisto. Työn toteutuskieli on Python. Jos aikaa jää, niin algoritmin toimintaa voidaan yrittää tehostaa käyttäen erilaisia optimointeja, esim. epäkronologinen taaksepäinhyppy.

Toiminta karkealla tasolla

Ohjelma lukee DIMACS-tiedoston, jonka pohjalta muodostetaan propositiologiikan kaava konjunktiivisessa normaalimuodossa. Tämän jälkeen suoritetaan itse algoritmi, joka palauttaa kaavan totuusjakelun, jos sellainen on olemassa tai tiedon siitä, että jakelua ei ole. Algoritmin toiminnan testaamiseksi toteutetaan yksikkö- ja suorituskykytestejä. Työn ydin on siis toteuttaa oikein toimiva ja riittävän suorituskykyinen DPLL-algoritmi. Tässä harjoitustyössä riittäväksi suorituskyvyksi on arvioitu toteutus, joka pystyy ratkaisemaan ~100 muuttujan kaavoja.

Tarvittavat tietorakenteet

Pythonin sisäänrakennetut tietorakenteet, kuten list, tuple, dict, set. Mitään monimutkaisempia ei todennäköisesti tarvitse rakentaa.

Riippuvuudet

- Testaus: pytest
- Testitapausten generointi: CNFGen

Kehitysympäristö:

- MacBook Pro M1 sirulla ja 16 GB muistilla
- Visual Studio Code
- Python 3.12.4
- Virtuaaliympäristö: Python venv

Dokumentaatio ja ohjelmointikäytännöt

Ohjelmakoodi ja kommentit englanniksi. Muu dokumentaatio suomeksi.

Aikavaativuus

DPLL-algoritmin aikavaativuus on pahimmassa tapauksessa eksponentiaalinen, sillä algoritmi käy läpi kaikki mahdolliset muuttujien totuusarvot, mutta käytännössä se voi olla huomattavasti nopeampi monien optimointien ansiosta.

1. **Pahimman tapauksen aikavaativuus**:
    - Ongelman totuusarvotilan koko on 2^n, missä n on muuttujien määrä.
    - Pahimmassa tapauksessa algoritmi käy läpi kaikki mahdolliset 2^n muuttujan totuusarvot, jolloin aikavaativuus on $O(2^n)$
2. **Käytännön optimoinnit**:
    - **Yksikköpropagaatio**: Vähentää ratkaistavien muuttujien määrää tunnistamalla yksikäsitteiset muuttujien arvot. Tämä pienentää tarvittavien haarautumisten määrää.
    - **Puhtaan literaalin poisto**: Poistaa joitakin muuttujia, jotka esiintyvät vain joko positiivisena tai negatiivisena. Tämä vähentää hakupuun kokoa.
    - Näiden optimointien ansiosta todellinen suorituskyky on usein huomattavasti parempi kuin pahimman tapauksen analyysi.
3. **Keskiarvoinen tapaus**:
    - Aikavaativuus on ongelmakohtainen, mutta optimointien ansiosta se voi olla lähempänä lineaarista monilla todellisilla ongelmilla.

Tilavaativuus

DPLL:n tilavaativuus määräytyy seuraavista tekijöistä:

1. **Rekursiivinen syvyys**:
    - Rekursiivinen kutsupino voi syventyä korkeintaan n tasolle, missä n on muuttujien määrä.
    - Tilavaativuus rekursiopinon osalta on O(n)
2. **Klausuulien kopiointi**:
    - Jokaisella rekursiotasolla algoritmi kopioi jäljellä olevan lausekejoukon ja muokkaa sitä. Tilavaativuus riippuu alkuperäisten lausekkeiden määrästä (m) ja muuttujien määrästä (n).
    - Jos lausekkeet ovat pitkiä, pahimmassa tapauksessa tilavaativuus on: O(m * n)
3. **Muuttujien totuusarvojen tallennus**:
    - Jokaiselle muuttujalle tallennetaan totuusarvot. Tämä vie lineaarisesti tilaa suhteessa muuttujien määrään: O(n)

Vertaisarvioinnit:

Pystyn vertaisarvioimaan ainakin seuraavilla kielillä toteutettuja projekteja:
Python, Java, C++, Typescript