Harjoitustyödokumentti

Tässä dokumentissa käydään läpi suurpiirteisesti harjoitustyössä tehdyt valinnat ja syyt niille. Funktiokohtaiset tehokkuuden arvioinnit löytyvät lähdekoodista datastructures.hh-tiedostosta.

Sekä paikat, että alueet säilytetään unordered_map -tietorakenteissa. Paikat ovat omassa säiliössään, jossa hakuavain on paikan id ja arvo on struct, jossa on kentät paikan nimelle, tyypille ja koordinaatille (x,y). Alueet ovat vastaavassa säiliössä, mutta structissa on kentät myös alueen suoran ”ylialueen” id:lle sekä sekä kenttä alueen suorille alialueille, joiden id:t ovat vectorissa tässä structissa.

Näiden unordered_map -säiliöiden käyttöön päädyttiin, koska niihin lisäykset ja niistä hakemiset avainten perusteella ovat keskimäärin vakioaikaisia operaatioita. Huonona puolena on, että alkioiden järjesteleminen tietyn ominaisuuden mukaan on hitaampaa ja työläämpää. Yksi vaihtoehto olisi esimerkiksi ollut lisätä paikat ja alueet mappeihin, mutta tällöin ei oltaisi esimerkiksi lisäyksissä ja avainhauilla päästy keskimääräiseen vakioaikaisuuteen, vaan ne olisivat olleet logaritmisia operaatioita aikakompleksisuudeltaan. Lisäksi näitä mappeja olisi joutunut tekemään paikoillekin useampia, jos olisi haluttu useiden ominaisuuksien mukaan pitää järjestyksessä ja tämä olisi vaatinut usean säiliön päivittämistä joidenkin operaatioiden yhteydessä. Vaikutti järkevämmältä siis käyttää unordered_mappia, vaikka järjestely onkin sillä hitaampaa. Lisäksi alun perin oli tarkoitus hyödyntää osoittimia ainakin alueen alialueiden ja ylialueen merkkaamisessa, mutta ne osoittautuivat ongelmallisiksi ja haasteellisiksi, joten niistä luovuttiin ja päädyttiin käyttämään vain kopioita alueiden id-arvoista.

Monet ohjelman komennot ovat aikakompleksisuudeltaan keskimäärin Theta(1). Unordered_map on kuitenkin ”hash table” toteutukseltaan ja on olemassa mahdollisuus, että useat näistä komennoista olisivat huonoimmassa tapauksessa lineaarisia. Paikkojen aakkosjärjestyksen laittaminen ja koordinaattijärjestykseen laittaminen (etäisyyys origosta) ovat kuitenkin tehokkuudeltaan O(n log(n)), mutta tämä valinta tehtiin, koska näin ollen esimerkiksi paikkojen lisääminen saatiin keskimääräisessä tapauksessa vakioaikaiseksi. Funktiot, joissa alkioita säiliön elementtejä joudutaan käymään läpi yksi kerrallaan ovat niiltä osin lineaarisia, ja niitä ei mielestäni saa asymptoottisesti tehokkaammaksi, muuten kuin siten, että kyseiset operaatiot, tehdään jossain muussa funktiossa, jolloin yksittäisen funktio asymptoottinen tehokkuus voi parantua, mutta kokonaisuuden kannalta sillä ei ole väliä. Työläin jäsenfunktio *places_closest_to* on myös asymptoottiselta tehokkuudeltaan O(n log(n)). Jos siitä olisi

halunnut tehokkaamman, pitäisi varmaankin tehdä sitä varten oma säiliö, mutta vaihtuva vertailupiste tuottaisi ongelmia, ellei säiliötä tehtäisi joka kerta uusiksi, mikä vaikuttaa hyvin raskaalta ja on todennäköisesti tehottomampi loppujen lopuksi.