30.03.2020

Ohjelmointi 4 Tietorakenteet ja Algoritmit

Harjoitustyö 1: Misse Meni?

Ohessa kuvat ohjelmassa käytetyistä tietorakenteista. Valitsin unordered_mapit, koska niitä käyttäen alkioita voidaan lisätä ja hakea tehokkaasti myös alkion määrien kasvaessa. Structit olivat hyödyllisiä, koska niiden avulla saadaan paljon tietoa tallennettua ja haettua helposti.

```
struct Stop {
    Name name;
    Coord coord;
    RegionID in_region = NO_REGION;
};
```

```
std::unordered_map<StopID, Stop> stops_ = {};
```

```
struct region_node {

    Name name;
    std::vector<RegionID> subregions = {};
    std::vector<StopID> stops = {};
    bool is_subregion = false;
    RegionID parentid = NO_REGION;

};
```

```
std::unordered_map<RegionID, region_node> regions_;
```

Ohjelmassa jouduttiin kuitenkin tekemään kompromissi funktioiden stops_alphabetically sekä stops_coord_order kanssa. Tämä johtuu siitä, että (unordered)mappia ei voida järjestää valueiden mukaan millään simppelillä tavalla. Tämän takia näiden kahden funktion tehokkuudeksi saatiin O (n * log n), mikä on kuitenkin mielestäni tyydyttävä, koska unordered_map mahdollisti usein kutsuttujen operaatioiden todella hyvän tehokkuuden. (Keskimäärin vakioaikainen kaikissa.)

Ohjelmassa käytetään monta kertaa seuraavaksi mainittua koodia:

```
if(stops_.find(id) == stops_.end()) { return false; }
```

Kyseisestä koodia hieman muokatessa pystytään testaamaan eri funktioissa stoppien ja regionien olemassaoloa, jotta ei tehdä operaatioita esimerkiksi pysäkille, jota ei ole tietorakenteessa olemassa. Tämän toteutuksen tehokkuus on huonoissa tapauksissa lineaarinen, mutta keskimäärin vakioaikainen, joten koin sen hyväksi tavaksi edellä mainittuun tarkasteluun. Perftestiä ajaessa huomasin, että lineaarista tapausta ei oikeastaan ollenkaan esiinny.

Olen myös luonut kaksi omaa rekursiivista funktiota, joita käytän toteutuksessa hyödyksi. Näiden funktioiden tehokkuusarviot on dokumentoitu samalla tavalla koodissa kuin muidenkin.

Joonas Pelttari 274830 joonas.pelttari@tuni.fi