1. Harjoitustyön toinen vaihe

Työn tarkoituksena oli tehdä edelliseen harjoitus työn osaan lisäksi algoritmit, jotka käsittelevät reittejä erilaisilla alueilla. Kurssihenkilökunta on toteuttanut jälleen kaiken muun projektia varten paitsi tietorakenteen joihin datat tallennetaan tai algoritmit joilla niitä käsitellään. Opiskelijan tehtäväksi on jäänyt toteuttaa sopivat tietorakenteet ja algoritmit niiden käsittelyksi.

2. Valitut tietorakenteet

Valitsin tietorakenteeksi väylille unordered mapin sen nopeiden käsittely aikojen takia. Suurin osa tämän rakenteen nopeuksista ovat keskimäärin vakioita tai pahimmassa tapauksessa lineaarisia. Rakenteeseen olen vallinnut avaimeksi väylän id:n, joka on uniikki arvo jokaiselle väylälle. Toisena arvona karttaan tallennetaan vectorissa olevat koordinaatti tiedot.

3. Pakollinen osuus

Pääosin sain pakolliset osuuden funktiot toimimaan O(n) tehokkuudella hash map rakenteen vuoksi, mutta todellisuudessa funktiot ovat nopeampia, sillä niiden keskiverto käsittely aika on vakio. Route_any jäin huomattavasti hitaammaksi kuin olin ajatellut sen olevan, eli O(n^3). Funktio olisi muuten ollut O(n^2), mutta jouduin lisäämään yhden for-loopin, jonka avulla lasketaan etäisyys joka kuljetaan väylää pitkin. Toteutin suoraan route_any funktioon algoritmin, joka löytää lyhimmän reitin alkupisteestä loppu pisteeseen.

3.1 kompromissit

Sain palautetta edelliseen projektiin, että tuplea ei tulisi käyttää. Kuitenkin olin kerennyt kirjoittamaan tämän projektin funktiot jo ennen palautteen saamista, joten olin kerennyt kirjoittamaan funktiot tuplea käyttäen. Normaalisti olisin voinut kirjoittaa rakenteet uusiksi palautteen mukaan, mutta tällä kertaa jätän nuo tuplet tuonne, enkä korjaa niitä millään tähän projektiin.

Sen sijaan, että käyttäisin painotuksien tarkastelussa priority_queuta käytän set-rakennetta, koska se on suoraan aina nousevassa järjestyksessä.

4. Vapaaehtoinen osuus

Olin toteuttanut suoraan any_route funktioon nopeimmat reitin etsimisen, joten kutsun vain sitä lyhimmän reitin etsinnässä ja palautan sen palautteen. Huomasin, että algoritmi ei kuitenkaan välttämättä toimi, koska se palautti kinttulammi testeissä virheellistä palautetta, mutta en ole varma mistä se virheellinen palaute johtuu, sillä palaute tulee 2 viimeisellä väylällä.

Reitti, jossa pyritään etsimään vähiten risteyksiä on toteutettu samalla periaatteella kuin route_any, mutta painotukseksi on laitettu matkan sijaan risteyksien määrä.