Структуре података и алгоритми 2024/25 – Други домаћи задатак

Никола Марковић, 1230/23

За израду је одабран језик C++ и уз њега један строго објектно-оријентисан приступ. Главне структуре су имплементиране као класе, док су помоћне имплементиране користећи *struct*-ове и STL шаблоне.

- Структуре:

Граф је, према спецификацији, имплементиран матрицом сусједности која се у меморији репрезентује као динамички низ са два нивоа индирекције, тј. 2D низ у неконтинуалном комаду меморије. **Стабло бинарног претраживања** (даље – *BST*) је имплементирано стандардно, као бинарно стабло динамичких *BSTNode* објеката који, поред показивача на лијево и десно дијете, додатно имају показивач „унапријед“ односно подршку за више чворова са истим кључем примјеном механизма једноструко уланчане листе. Овакав приступ је одабран како би се ријешио неминовни проблем појаве више возила на истој адреси (и самим тим на истој удаљености од корисника).

- Алгоритми:

Алгоритми за рад са BST-ом су највећим дијелом стандардни, уз неке додатке које омогућавају несметан рад у контексту задатка. Неки су имплементирани рекурзивно, а неки итеративно. **Дајкстрин** алгоритам за најмање удаљености је имплементиран једноставним приступом угнијеждених петљи временске сложености $O(V^2)$ и проширењем (које је типично Флојд-Воршаловом алгоритму) за реконструкцију путање. Све остало ради искључиво уз помоћ STL шаблона, најчешће вектора. **Проток времена** је реализован на начин да сваки корисник на реду који чека да се ослободи возило изазове временски „скок“ у трајању од преосталог времена вожње заузетог возила најближег својој коначној дестинацији, а управо то возило се датом кориснику потом и додјељује. Овај скок се односи на свако активно возило, али и на укупно вријеме чекања сваког корисника који није тренутно у возилу.

- Оптимизација:

Иако се Дајкстрин алгоритам може имплементирати на другачији начин (коришћењем приоритетних редова, мин. хипова итд) за бољу временску сложеност, још једна честа оптимизација јесте заустављање након што се добије жељена веза између заданог и траженог чвора, умјесто да се вриједности купе тек када се алгоритам потпуно изврши. Додатно, сама структура задатка доводи до потребе да се минималне везе између чворова заједно са путањама траже стотинама пута, самим тим позивајући да се алгоритам изврши над комплетним графом стотинама пута. Ови проблеми се рјешавају уз једну методу која се позива прије почетка рада са графом. Она позива извршавање Дајкстриног алгоритма над сваким чвором графа и чува резултате у меморији (као приватне атрибуте) у виду минималних удаљености и стрингова путања. Ове вриједности се потом просто дохватају по потреби без поновних позива самог алгоритма, ограничивши број његових извршавања на тачно V пута. Додатно, ово рјешава потребу за раније наведеном дорадом алгоритма да се заустави када се оствари жељена веза два чвора.

- Улаз и излаз:

Улазне *.csv* и *.txt* датотеке се систематски обрађују уз претпоставку да су синтаксички тачне. Постоје функционална ограничења (нпр. 0 возила, 0 корисничких захтјева, празан граф, упити/возила ван опсега графа итд) за које се бацају, хватају и исписују одговарајући изузеци. Уколико су улазни подаци исправни, на стандардни излаз се исписује хијерархија примања захтијева, путовања возила од почетне локације, па до корисника и коначно до корисничке дестинације. На крају свега се исписује одговарајућа статистика за кориснике и возила респективно. Захтјеви се примају и обрађују редом.