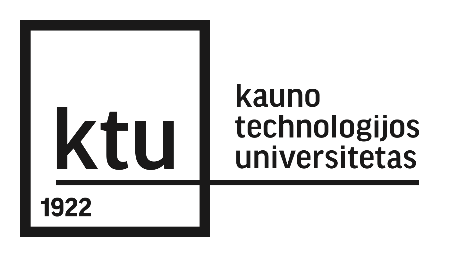
****

**Informatikos fakultetas**

**Duomenų struktūros**

Technologinis projektas

Traveling Salesperson Problem

**Atliko:** IFF-1/1 gr. stud. Paulavičius Karolis

**Priėmė:** dėst. Budnikas Aurelijus

**KAUNAS, 2022**

**Užduoties formuluotė**

Parašykite programą keliaujančio pirklio uždavinio apytiksliam sprendimui apskaičiuoti, sprendimą naudokite rasti trumpiausią maršrutą, jungiantį duotus taškus plokštumoje.

**Užduoties sprendimo algoritmo aprašymas**

Euristika *insertNearest()* naudojama taškams įterpti pagal artimiausią tašką. Tikrinama, ar yra pirmas taškas, su kuriuo galima palyginti. Jei taško nėra, pridedame jį. Tada pereiname per visus taškus ir lyginame atstumą tarp kiekvieno taško ir pridėtinio taško. Šį veiksmą tęsiame, kol visi taškai atsiranda maršrute. *insertNearest()* algoritmo asimptotinis sudėtingumas yra O(n2).

*insertSmallest()* algoritmo asimptotinis sudėtingumas yra O(n2). Šis metodas lygina taškus ir pasirenka, įterpimą kuris padarytų mažiausią atstumą. Šis procesas buvo atliekamas užtikrinant, kad būtų pridėti bent trys taškai. Jei taip nebuvo, taškai buvo pridėti. Toliau einame per visus taškus ir lyginame atstumą su dabartiniu maršrutu.

double d1 = p.distance(node.data);

double d2 = p.distance(node.next.data);

double d3 – node.data.distance(node.next.data);

double tempMin = d1+d2-d3;

Šiuo atveju imamas atstumas tarp dabartinio mazgo ir taško, bei atstumas tarp kito mazgo ir taško, atėmus pradinį atstumą. Toliau atstumas *tempMin* buvo lyginamas su kintamuoju, kuris turėjo mažiausią atstumą. Tai leido nustatyti geriausią sekantį tašką. Kai dabartinis mazgas buvo ties paskutiniu mazgu, *tempMin* buvo vėl lyginamas su mažiausiu atstumu. Jei jis buvo mažesnis, taškas būtų įrašomas į sąrašo pabaigą. Jei jis buvo didesnis, taškas buvo įterptas geriausioje įterpimo vietoje.

Šio kodo problema yra ta, kad kai jis iteruoja per dabartinį sąrašą, mes nepatikriname, ar taško pridėjimas į pabaigą yra mažesnis, nei dabartinė geriausia pozicija. Taip sumaišomi keli skaičiai.

**Duomenų struktūrų, naudojamų uždaviniui spręsti aprašymas:**

Šiam uždaviniui buvo pasirinktas žiedinis susietasis sąrašas. Šios duomenų struktūros pagalba, buvo galima sumažinti sudėtingumą įterpiant viršūnes ieškant kelią.

**Išvados**

Pavyko sukurti du veikiančius algoritmus keliaujančio pirklio problemai spręsti, tai pat sukūriau vizualizaciją aiškesniam algoritmų veikimo proceso pateikimui. Esant dideliam viršūnių kiekiui, niekas nežino efektyvaus metodo, kuriuo būtų galima rasti trumpiausią įmanomą maršrutą bet kokiai duotai taškų aibei. Tačiau ištirta daug metodų, kurie, atrodo, gerai veikia praktikoje, nors ir nėra garantuota, kad jie duos geriausią įmanomą maršrutą.