PROJEKT PRI FINANČNEM PRAKTIKUMU

ISKANJE BLIŽNJICE V DREVESU

Avtorja: Katarina Kromar in Žiga Kodrič

ORIGINALEN OPIS PROBLEMA

Take an arbitrary tree T and assume that all the edge weights are 1. We want to add an edge e ("a shortcut") to T that minimizes the average distance between the vertices. This means that we want to choose the edge e such that the sum of the distances in T+e between all the pairs of vertices is minimized. We can find such an edge e by trying all possible edges. Check several random trees and see how much the average distance between vertices decreases when inserting the best shortcut. You may want to check also for two shortcuts (for small trees.)

OPIS PROBLEMA

Vzamemo drevo T in predpostavimo, da imajo vse povezave utež 1. Radi bi dodali neko povezavo e (bližnjico) na drevo T, ki bi minimizirala povprečno razdaljo med vozlišči. Torej bi radi izbrali tako povezavo e, da bo vsota razdalj v T+e med vsemi pari vozlišč minimizirana. Lahko poskusimo najti tako povezavo e tako, da poskusimo vse možne povezave.

Ta postopek bova poskusila na nekaterih naključnih drevesih, kjer bova kodo za generiranje takih drevesih uporabila kar z interneta. Poskusila bova ugotoviti, koliko se povprečna razdalja med vozlišči zmanjša, ko vključimo v drevo najboljšo bližnjico. Poskusila bova tudi z vključitvijo dveh bližnjic, kar pa bo seveda zaradi zahtevnosti programa potrebno poskusiti na manjših drevesih.

PREDSTAVITEV GRAFOV NA RAČUNALNIKU IN GENERIRANJE PODATKOV

Grafe lahko predstavimo z matriko sosednosti ali pa s seznami sosednosti. Pri projektu bova uporabljala sezname, ker so bolj optimalni, saj zasedejo manj pomnilnika. Za vsako točko imamo torej seznam naslednikov in predhodnikov. Pri tem porabimo le pomnilnika. Če pregledamo vse sosede vozlišča porabimo časa, za pregled vseh povezav pa .

Pri najinem projektu potrebujeva naključna drevesa. Napisala (oziroma poiskala na internetu) bova program, ki bo vrnil naključno drevo na n točkah v obliki seznama sosedov.

OSNOVNA IDEJA ALGORITMA

Algoritem sva si zamislila na naslednji način. Najprej bova zgenerirala naključno drevo na n točkah. Algoritem si bo nato izbral prvo točko. Tej točki bo dodal bližnjico in preko algoritma BFS izračunal vse možne razdalje, ter vrnil povprečje le teh. Nato bo dodal naslednjo bližnjico (in prejšnjo seveda izbrisal), ter ponovno izračunal povprečje razdalj. V primeru, da bo povprečje manjše, bo shranil to bližnjico, v nasprotnem primeru pa jo bo ovrgel. Tako bo nadaljeval z računanjem dokler ne bo preveril vseh možnih bližnjic. Ko bo preveril vse možne bližnjice, bo nadaljeval na naslednjo točko in ponovno preveril vse možnosti. Algoritem se bo končal, ko bo preveril vse možnosti na vseh točkah.

Program si bo hkrati zapisoval katere bližnjice je že pregledal. Če bo torej izračunal razdalje ko bo dodal bližnjico , bližnjice ne bo ponovno pregledal.

Programirala bova v programskem jeziku Python.

BFS (BREADTH - FIRST SEARCH ALI PREGLED V ŠIRINO)

BFS se uporablja za neutežene grafe. Za vhod vzamemo neutežen graf G, pa je izvor/ začetno vozlišče. Za izhod dobimo drevo najkrajših poti v G do s.

Ideja:

* , ,
* konstruiramo in ko najdemo , lahko končamo

Algoritem BFS izračuna drevo najkrajših poti v neuteženem grafu, ki je lahko usmerjen ali neusmerjen, v linearnem času O().

Najin algoritem za pregled v širino bo porabil O()+ O()= O() operacij, torej bova morala paziti, da ne bova uporabila algoritma na prevelikih drevesih. Ko vključimo še eno bližnjico, porabimo dodatno še O() operacij, torej skupno porabi O() operacij (potrebno bo uporabiti še manjša drevesa, zaradi časovne zahtevnosti).