KRATKO POROČILO

(če bo potrebno, ga dava v LaTeX, končno poročilo zagotovo)

Navodila:

Skupina 6: Wiener inverse interval problem (2 students)

In the Wiener inverse interval problems, we usually look for trees whose Wiener index value cover a large interval of integers. In most cases, all these trees can be obtained one from another by small changes. For the purposes of this project, we would like to deal with the following two subproblems:

P1. For a fixed number of vertices n and a tree T, let Tn+1 be the set of all trees on n+1 vertices obtained from T by adding a leaf to a vertex of T. Let W(Tn+1 ) be the set of values of the Wiener index for trees from Tn+1 . Find a tree T on n vertices such that |W(Tn+1 )| is the largest (smallest) possible.

P2. For a fixed number of vertices n, find a tree T with the largest possible diameter such that there is a leaf u in T appended to a vertex v and a vertex w in T such that deleting the edge uv and adding the edge uw changes the value of Wiener index by 1.

In order to get the answer to both problems for very small values of n first, apply an exhaustive search, and next, for larger n, apply a genetic algorithm or any other metaheuristic. Verify for how large n your exhaustive search and your genetic algorithm implementations are efficient.

P1:

S pomočjo funkcije nonisomophic\_trees iz knjižnice *networkx* najprej generiramo vse neizomorfne grafe z n vozlišči in jih dodamo v seznam (v najinem primeru poimenovan *sez*). Iz tega seznama s pomočjo funkcije random.shuffle iz knjižnice *numpy* dobimo naključno drevo z n vozlišči, ki ga bomo uporabili v nadaljevanju. Tukaj ob vsakem zagonu programa dobimo le eno drevo, za pisanje genetskega algoritma pa bo seveda potrebno generirati veliko število tovrstnih grafov. Kodo bo potrebno tako spremeniti, da bo vzorec dovolj velik, da se bo iz njega že videlo, kakšne vrste grafi so najbolj verjetni kandidati za iskana drevesa z nizkim Wienerjevim indeksom.

Funkcija seznam\_sosedov nam vrne seznam vseh vozlišč (oštevilčenih z naravnimi števili od 0 do n-1) in njihovih sosedov. Na podlagi tega seznama potem funkcija listi iz vseh vozlišč izloči samo tiste, ki so listi, torej imajo zgolj enega soseda. Ta funkcija je sicer uporabna le pri P2.

Nato naredimo kopije našega drevesa (za vsako vozlišče eno) in na vsaki kopiji na drugo vozlišče povežemo n+1-tega. V sprva prazen seznam M nato dodamo vsakega od teh grafov z n+1 vozlišči. S pomočjo funkcije wiener\_index iz knjižnice *networkx* še določimo Wienerjev indeks vsakega od grafov iz M in ven izberemo tistega z najmanjšim.