ANALIZA OMREŽIJ - SEMINARSKA NALOGA

Ker smo se doslej predvsem ukvarjali s ne-usmerjenimi omrežij, seminarska naloga bo orientirana k analizi usmerjenih omrežij. Ogromno realnih omrežij so v bistvu usmerjena: moja spletna stran lahko ima link na tvojo, obratno pa ne; jaz te lahko imam za prijatelja, ti mene pa ne; Slovenija mogoče ima veliki izvoz avtomobilov v Avstrijo, Avstrija v Slovenijo pa ne, in tako naprej.

Koda v Pythonu (napisana s uporabo NetworX-a), prebere usmerjeno omrežje podano kot seznam povezav (mu bomo rekli “input omrežje”). Ta seznam povezav je tokrat mišljen v usmerjenem smislu, kar pomeni, da npr naslednji seznam:

1 2

3 2

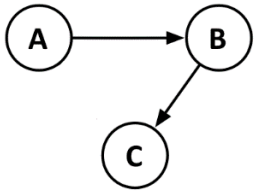
označuje omrežje kje imamo link s vozlišča 1 na vozlišče 2 (obratno pa ne), ter s vozlišča 3 na vozlišče 2 (obratno pa ne). Omrežje kot vedno lahko zapišemo v obliki matrike sosednosti, ki zdaj ne bo več simetrična. V zgornjem primeru bomo imeli A[1][2]=1, vendar A[2][1]=0, itn.

Input omrežje lahko predpostavimo da je povezano v šibkem smislu (to pomeni, da če ne uspoštevamo usmerjenost, lahko pridemo s vsakega vozlišča na vsako drugo). V input omrežju bomo s i,j,k označevali vozlišča (indeks vozlišča). Koda mora delovati za vsako usmerjeno input omrežje, za testiranje dela kode lahko pa uporabimo kakšnokoli generirano usmerjeno omrežje (najbolje da štartamo s majhnimi grafi).

Za podano input omrežje koda izračuna in izpiše naslednje lastnosti:

1. Izpis in-degree (vhodnih stopenj) in out-degree (izhodnih stopenj) za vsako vozlišče i. Tukaj se preprosto poračuna koliko povezav gre “v”, in koliko ih gre “van” vsakega vozlišča. Koda izpiše za vsako vozlišče i njegovo izhodno in vhodno stopnjo.

2. Na sliki spodaj vidimo en primer enostavnega usmerjenega omrežija s 3 vozlišča (usmerjena 3-veriga). Pozorni moramo biti na dejstvo, da niso ta 3 vozlišča topološki enaka: iz vozlišča A izhaja en link, v vozlišče C prihaja en link, med tem ko v B izhaja en in prihaja en link.



   Koda naj poračuna za vsako vozlišče i, koliko krat je udeleženo v usmerjeni 3-verigi kot vozlišče A, koliko krat kot vozlišče B, in koliko krat kot vozlišče C. Da bi za vozlišče i poračunali koliko krat je udeleženo v 3-verigi kot A, prvo pogledamo izhodne linke iz i, potem pogledamo kdo od izhodnih sosedov j ima link na neko tretje vozlišče k, takšno da i nima nobenih stikov s k. To je en primer

kje je i udeležen kot A v neki 3-verigi, zdaj treba pogledati koliko različnih situacij s različnimi 3-verigami najdemo za i. Na podoban način lahko izračunamo udeležbo vozlišča i v različnimi 3-verigami kot B in kot C. Za vsako vozlišče i, koda izpiše 3 števila: udeležba v 3-verigah kot A, kot B, in kot C.

3. Usmerjeno input omrežje je šibko povezano, kar pomeni, da če uspoštevamo usmerjenost povezav, lahko obstaja en del omrežja (eno pod-omrežje) iz katerega ni možno priti do vseh preostalih vozlišč, preprosto zato ker usmerjenost tega ne dovoli. Npr, na zgornji sliki, enkrat ko smo na vozlišču B, ni mogoče več priti v vozlišče A (v C pa vendar lahko pridemo). Koda naj detektira in izpiše v input omrežju takšno pod-omrežje (npr, na zgornji sliki se gre za pod-omrežje sestavljeno iz B in C, zato ker s teh vozlišč ni mogoče priti do vseh drugih vozlišč, namreč do A). To pod-omrežje je torej sestavljeno iz takšnih vozlišč i, za katere obstaja vsaj eno drugo vozlišče j, na katero ni mogoče priti iz i, če upoštevamo usmerjenost.

4. Za vsako vozlišče input omrežja koda naj poračuna PageRank. To pomeni, da rabimo statistiko obiskovanj vsakega vozlišča, za sprehajalca ki se naključno sprehaja po input omrežju in upošteva usmerjenost povezav. Začnemo tako da določimo parameter α, npr lahko damo α=0.1. Pri vsakem koraku, sprehajalec s verjenostjo 1-α skoči na enega naključno izbranega soseda (ne pozabiti na usmerjenost), in s preostalo verjetnostjo α se “teleportira” na eno naključno izbrano vozlišče kjerkoli v input omrežju (ne glede če je to sosed ali ne). Recimo da je treba narediti čim večje število takšnih korakov, vsaj 20 krat več kot je število vozlišč. Za vsako vozlišče input omrežja potem preštejemo koliko krat je bilo obiskano v tem procesu, in to število je PageRank. Koda izpiše PageRank za vsako vozlišče.

Pri izdelavi zgoraj opisanih nalog, ni dovoljeno uporabljati gotovih Pythonovih paketov ki avtomatično dajo rezultate, do rezultata je potrebno priti programerski. To vključuje že narejene funkcije za vhodno in izhodno stopnjo, pakete kot je network_motif_counter.py, ali pa funkcije kot so weakly_connected_components(), weakly_connected_component_subgraphs() in funkcije nx.pagerank(). Seveda, takšne gotove kode lahko pa uporabimo za preverjanje točnosti naših kod.