Matemàtica Discreta - Seminari 2 - Fonaments de Grafs

Funcionament dels seminaris

El primer i primordial que heu de saber és:

ÉS MOLT IMPORTANT LLEGIR BÉ ELS ENUNCIATS I CONTESTAR EL QUE ES DEMANA DE LA MANERA QUE ES DEMANA, NO UNA ALTRA COSA.

Aquest és el motiu més habitual de problemes tant als seminaris com als exàmens.

Durant aquests seminaris intentarem aplicar, d'una forma pràctica, aquells coneixements que anem adquirint a teoria i problemes de l'assignatura. Per seguir els seminaris necessiteu un ordinador, un editor de textos i algun tipus de IDE que us permeti programar i executar Python3.

Cada seminari es compondrà d'una sèrie de preguntes i d'algun exercici de programació. Per facilitar la feina, per a cada seminari tindreu disponible un esquelet de programa que heu d'omplir i d'un conjunt de testos per validar que el vostre programa funciona. Per tant, sempre disposareu d'una carpeta anomenada esquelet amb almenys tres fitxers. Un main per fer proves o crear objectes concrets, un o més fitxers que heu d'omplir i on tindreu pistes sobre el que heu de fer i un fitxer de testos.

Per a començar sempre heu de duplicar la carpeta esquelet i dir-li 'solucio' i en aquesta carpeta és on heu de fer els canvis. La carpeta esquelet no l'heu de tocar per res.

El fet que el programa passi el conjunt de tests donat, no vol dir que sigui correcte. Primer, perquè els testos donats no comproven tots els casos és a dir, es valorà la capacitat de tractar casos extrems o no contemplats, i a més, es valorarà que el codi segueixi els conceptes d'Eficiència, Escalabilitat i Llegibilitat. En canvi, no passar el conjunt de tests si que es considerarà un seminari no correcte.

Els lliuraments es compondran d'un pdf amb les respostes a les preguntes teòriques, d'un arxiu README.md amb les instruccions per a l'execució i els arxius .py necessaris per a la seva execució. Tot comprimit en un zip o un tar.gz.

Exercici 1 - Un whatsapp menys invasiu

Imaginem que volem crear una nova aplicació de missatgeria estil Whatsapp, però menys invasiu, de manera que dues persones només es poden enviar missatges si el telèfon de cada un d'ells està guardat com a contacte al telèfon del altre. D'aquesta manera ningú que no coneguis et pot enviar missatges o afegir-te a grups indesitjats.

Aquest tipus de relació es pot modelar amb un graf.

1. Imagineu que les següents persones baixen l'aplicació:

   - La Teresa té el contacte de la Jana, l'Ariadna, en Joan i en Pau.
   - La Jana té el contacte de la Teresa, i l'Ariadna
   - L'Ariadna té el contacte de la Teresa, la Jana i en Pau.
   - En Joan té el contacte d'en Pau, la Teresa i la Jana
   - En Pau té el contacte d'en Joan de l'Ariadna i de la Jana.

   Atenció! El model que esteu creant és el model de la aplicació. El que us heu de preguntar és, des del punt de vista de la aplicació de missatgeria que volen dir els nodes, i que volen dir les arestes? Per a l'aplicació, el fet que una persona A tingui el contacte d'una persona B sense que B el tingui de A, que implica? Poden tenir comunicació? Llavors en aquest cas hi ha d'haver o no una aresta?

2. Quin tipus de graf heu dibuixat?

3. Ara programarem aquesta xarxa social. Copieu la carpeta 'esquelet', l'enganxeu a la mateixa ubicació que 'esquelet' i l'anomeneu 'solució'. Obriu la carpeta que heu creat anomenada 'solució'

Us donem alguna ajuda:

- graph.py: Aquí trobareu dues classes (agrupacions de funcions i variables). Per una banda, teniu la classe Node, que implementa un node d'un graf. Cada node està compost d'un nom, una posició en x i y (que servirà per dibuixar-lo) i una llista d'altres nodes que representen les arestes. Les funcions que necessiteu omplir estan marcades al comentari de cada una d'elles. **Pistes:**
  - Tot el que està dins de la classe (tabulat) son funcions de la classe.
  - Les classes són com una variable però complexa (conté altres variables i funcions) així, la classe Node conté un seguit de funcions que son les que estan a dins de la tabulació.
  - A més, la funció init declara i inicialitza les variables propies de la classe.
  - La paraula clau "self.[es] refereix a l'objecte mateix. Així, si la classe Node té una variable edges, per accedir a la varaible DES DE DINS de la classe (és a dir, des de dins d'alguna de les funcions de la classe) s'utilitzarà self.edges. Si volem accedir des de fora, primer cal instanciar una variable concreta (ex. node1 = Node("nom",0,0)) i llavors accedir a la seva variable edges de la següent manera: node1.edges.
  - Per crear una aresta podeu afegir el node amb què us voleu connectar a la llista d'arestes mitjançant la funció edges.append(node). Recordeu si esteu dins de classe llavors serà self.edges.append(node). Mireu-vos les llistes en Python si no hi esteu familiaritzats.
  - Per crear una aresta simple cal que els dos nodes tinguin a l'altre en la llista
  - Si us trobeu un paràmetre d'aquest estil f(parametre=False) vol dir que si no poseu res en la crida f(), el paràmetre tindrà un valor per defecte de False.
  - Per crear un graf, primer heu de crear els nodes i un cop els tingueu, afegir-los al graf amb la funció add_node(node).
  - Per a dibuixar el graf cal cridar la funció draw del graf que ja ve donada. La funció draw del node és una funció que no dibuixarà directament. Dit d'una altra manera, el que es dibuixa és el graf, no els nodes per separat.
- main.py: És el vostre main. Des de la funció main podeu fer tantes proves com vulgueu. A més hi ha dues funcions més, una per a l'exercici 1 i l'altra per a l'exercici 2.

Què heu de fer:

- Omplir aquelles funcions de graf.py que indiquin que siguin de l'exercici 1
- Crear el graf descrit en aquest exercici a la funció nou_whatsapp() que crea, dibuixa i finalment retorna el graf de l'exercici.

Exercici 2 - Instagram

Ara programarem l'Instagram. A l'Instagram les relacions són una mica diferents, perquè un usuari pot seguir a un altre sense que aquest també el segueixi a ell.

Aquest tipus de relació es pot modelar amb un graf.

1. Imagineu que les següents persones baixen l'aplicació:
   - La Teresa segueix a la Jana, a l'Ariadna, a en Joan i a en Pau.
   - La Jana segueix a la Teresa, i a l'Ariadna
   - L'Ariadna segueix a la Teresa, la Jana i en Pau.
   - En Joan segueix en Pau, a la Teresa i a la Jana
   - En Pau segueix en Joan, l'Ariadna i la Jana.
2. Quin tipus de graf heu dibuixat?
3. Ara programeu-ho. Ompliu aquelles funcions de graf.py que indiquin que siguin de l'exercici 2. Crear el graf descrit en aquest exercici a la funció instagram() que crea, dibuixa i finalment retorna el graf de l'exercici.

Exercici 3 - Havel-Hakimi (opcional)

Hem vist com treure la seqüència gràfica d'un graf, però, donada una seqüència qualsevol com per exemple [4,3,2,1,1], com podem saber si és gràfica? Programeu una funció a part de les classes donades (al fitxer havelkhakimi.py) que implementi l'algorisme de havel-hakimi. La seva signatura és: havel_hakimi(seq) on seq és una llista d'enters eventualment no ordenada.

Lliurament: Heu d'entregar, per grups, un tar.gz o zip amb les preguntes respostes i el codi al CV, tot abans del 16 d'Abril a les 23:59.