Pràctica 1 GRAFS

Nom: Adrià Muro Gómez
NIU: 1665191

Nom: David Morillo Massagué
NIU: 1666540

1. **Quines proves has fet mentre feies les tres primeres tasques de la pràctica? Per a grafs de quin ordre i mida?**

   Hem creat un graf, modificant les funcions per a que només tinguin en compte les primeres 1000 línies de l'arxiu. Fent això, el graf resultant de la funció build_simplegraph() té un ordre de 476 (vèrtexs) i de mida 819 (arestes).

   Fent això, el temps de còmput per a crear i visualitzar el graf i digraf es reduïa, passant d'un minut a un parell de segons, i a l'hora oferint-nos suficient informació per a comprovar que la funció funcionava correctament.

2. **Quants components hi ha al conjunt de dades que ens han passat?**

   16706 arestes i 1005 vèrtexs que acaben essent 20 components del graf.

3. **En el cas general, creieu que components_BFS() és més ràpid que components_DFS()? A l'inrevés? Són aproximadament el mateix? Per què?**

   En el cas general els dos mètodes de búsqueda tsón igual de ràpids. No obstant, depén del graf en el qual s'aplica ja que, si el graf no és molt ample i les solucions son llunyanes (profunditat) DFS és millor. En el cas de que el graf sigui ample en solucions però aquestes no estiguin molt separades del node inicial, BFS és una bona opció.

4. **Per al nostre cas particular, quin és més ràpid? Com ho heu determinat?**

   En el nostre cas el mètode BFS ha resultat ser més ràpid. Ho hem determinat a partir de la llibreria time per veure el temps que tardava cada funció (DFS o BFS) en executar-se i acabar. De mitja BFS ens tarda 0,14 segons mentre que DFS 0,18 segons.

5. **Quins algorismes heu utilitzat per construir la funció how_many_degrees(G,a,b)? Per què?**

   Sobretot, en la funció how_many_degrees hem utilitzat l'algoritme de búsqueda BFS. Dona que necessitem comptar els salts desde el número inicial i el resultant escollits, el BFS era l'ideal ja que aquest revisa tots els veïns del primer nivell abans de passar al segon, així doncs, cada nivell sumava una iteració.

6. **Diem que el diàmetre d'un component d'un graf és la distància mínima màxima entre dos vèrtexs de la component. Quin és el diàmetre de cada component del graf que representa el conjunt de dades que ens han passar? (Construïu una funció diameters() que retorna una llista dels diàmetres.)**

   El diàmetre de cada component és: [7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

7. **Expliqueu detalladament com heu construït la funció diameters( ).**

   El que hem fet ha sigut calcular les distàncies per cada combinació (en parelles) de nodes, fent servir per això és la funció how_many_degrees().

   El que hem fet ha sigut, primer de tot, separar el graf en components fent servir la funció prèviament creada components _DFS(), encara que també podriem haver utilitzat la components _BFS().

   A continuació, fem el primer bucle for, que iterarà entre les components de la llista creada amb components _DFS().

   El que farem serà calcular cada distància basant-nos en una matriu triangular com aquesta:

0	**X**	**X**	**X**
/	**0**	**X**	**X**
/	**/**	**0**	**X**
/	**/**	**/**	**0**

   Fem servir files com a node 1 i columnes com node 2 (que son els que compararem amb el how_many_degrees()). Sabem que la diagonal és 0, ja que sabem que la distància entre un node i ell mateix és 0. Com que, en un graf simple, sabem que la distància entre dos nodes x, y és la mateixa que entre y, x, només calcularem una d'elles.

   El seguent bucle itera per cada vèrtex de la component actual. Aquest actúa com a fila de la matriu. Creem una variable “maxnode”, que guardarà el màxim nombre d'iteracions d’aquesta fila.

   A continuació fem un seguent for, que actuarà com a columnes de la matriu.

   El que fem és agafar el node de la fila i calcular (amb la funció how_many_degrees()) la distància mínima entre aquest i cadascun dels de la columna. Si aquesta és més gran que el de la variable “maxnode”, la variable canviarà el seu valor a la nova màxima.

Fem el mateix amb una variable anomenada “maxcomponent” que té la mateixa funció, però per cada component, en aquest cas. Aquesta la fem fora de l'últim for, ja que es queda amb el màxim de cada combinació entre parelles d’aquell node.

A continuació, treiem el primer node de la llista per a cumplir amb l'estructura de la matriu triangular, com ja hem dit.

Finalment, afegim a la llista de diàmetres a retornar, la variable “maxcomponent”.