Universitat de Barcelona

Grau en Enginyeria Informàtica

Tardor del Curs 2015-2016

Tresors en un Mapa

Primera pràctica d'Algorísmica Avançada

- grafs -

1 Introducció

Com ja s'ha vist a teoria, el recorregut més simple que es pot fer d'un graf no és un recorregut topològic, sinó un recorregut de l'estructura de dades que el representa. És a dir, suposant que el graf està representat en un vector de llistes d'adjacència, el recorregut més senzill seria un bucle de cada adjacència dins un bucle per cada vèrtex, o sigui, per cada element del vector. Això com és sabut, significa un esforç computacional de $\Theta(V + E)$, que vol dir del màxim entre el nombre de vèrtexs i el d'arestes, i per tant, si el graf és connex, $\Theta(E)$. Així doncs, no hi ha dubte que el recorregut de l'estructura és el més eficient a l'hora de recórrer un graf.

Ara bé, quan el recorregut ha de satisfer condicions específiques que depenen del contingut o propietats dels elements del graf (ja siguin vèrtexs o arestes), llavors la millor aproximació és utilitzar recorreguts topològics.

En molts problemes en els que es tracta de sotmetre estructures matricials a anàlisis específiques es transforma la matriu en qüestió a un graf. Per cada casella de la matriu s'afegeix un vèrtex al graf, i segons el nombre de veïns que tingui la casella, s'hi afegeixen les arestes corresponents. Posem per cas que per moure'ns en una matriu podem caminar d'una casella a les quatre veïnes immediates. Llavors cada vèrtex del graf corresponent tindrà grau 4, excepte els de les fronteres de la matriu que tindran grau 3, i els quatre extrems que tindran grau 2. Per problemes on es pogués anar d'una casella a qualsevol veïna en diagonal, els vèrtexs del centre tindrien grau 8.

2 Cerca de Tresors en un Mapa

Es disposa d'una matriu com la de la Figura 1.

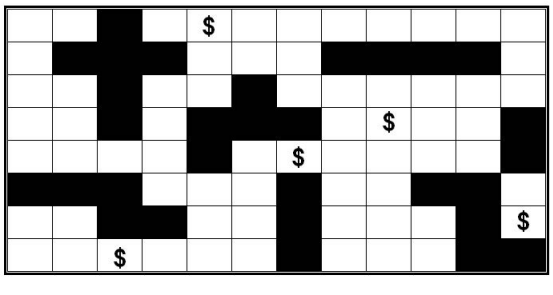

Figura 1: *Exemple de mapa amb tresors i obstacles*

El problema consisteix en codificar un algorisme que a partir de la informació continguda en aquesta matriu ens digui quants tresors hi ha, quants són accessibles des de la posició de dalt a l'esquerra, i quin és el mínim nombre de passos cal fer per recollir-los.

2.1 Entrada

Les dades d'entrada a l'algorisme són en format text, i pel cas de la Figura 1, l'entrada corresponent es descriu en el fitxer exemple1.dat:

```
8,12
..X.$.......
.XXX...XXXX.
..X..X......
..X.XXX.$..X
....X.$....X
XXX...X..XX.
..XX..X...X$
..$...X...XX
```

exemple1.dat

És a dir, a la primera línia de l'arxiu hi ha el nombre de files i columnes de la matriu. Després, hi ha una línia de text per cada fila, en la què un punt vol dir casella lliure, una ics vol dir casella prohibida, i un signe de dòlar vol dir tresor.

2.2 Sortida

La sortida ha d'indicar primer el nombre de tresors que conté el mapa. Després ha de llistar una línia per cada tresor, ordenats de més proper a més llunyà. Cada línia ha de mostrar la posició del tresor i el nombre de passes mínimes necessàries per arribar-hi, o "inaccessible"si no s'hi pot arribar. En el nostre exemple, la sortida seria:

```
Hi ha 5 tresors:
Tresor a la fila 1 columna 5 accessible amb 12 passes.
Tresor a la fila 5 columna 7 accessible amb 12 passes.
Tresor a la fila 8 columna 3 accessible amb 13 passes.
Tresor a la fila 4 columna 9 accessible amb 15 passes.
Tresor a la fila 7 columna 12 inaccessible.
```

sortida corresponent a l'entrada de la Figura 1

3 Desenvolupament

Heu d'implementar la pràctica en Python, en un únic fitxer que contingui tot el conjunt de funcions que heu implementat. Al campus virtual teniu un arxiu `plantilla.py` que heu d'utilitzar com a base per la vostra implementació, seguint les següents normes:

- El vostre fitxer s'ha d'anomenar

  `grafs_nom_cognom1_cognom2_G.py` ,

  on "G" és la lletra del grup de pràctiques al que pertanyeu.

- Assegureu-vos de posar el vostre nom al principi de l'arxiu, on està indicat que ho feu.

- Configureu el vostre editor per utilitzar codificació "UTF-8". D'aquesta manera podreu passar la pràctica entre Windows, Mac i Linux sense problemes amb els caràcters especials (accents, apòstrofs, etc.).

- Configureu el vostre editor per utilitzar 4 espais per al sagnat del codi. Així treballarem tots amb la mateixa configuració i evitarem barrejar espais amb tabuladors a l'hora de codificar.
- L'arxiu de plantilla està preparat per executar la pràctica amb una entrada per defecte, però podeu executar la pràctica amb un altre arxiu d'entrada també:

```
python grafs_nom_cognom1_cognom2_G.py <arxiu>
```

La presentació de les pràctiques d'aquesta assignatura consta tant del codi que implementa el problema, com de l'anàlisi d'eficiència. L'anàlisi el podeu efectuar comentant cada línia amb la seva eficiència, i/o comentant com l'heu calculada al comentari de cada funció implementada.

Podeu utilitzar la llibreria `networkx`[1] per definir els vostres grafs. Tot i així, **no es permet utilitzar funcions d'alt nivell que implementin recorreguts complets**, ni estructures de dades sofisticades de les que no en coneixem la implementació ni, per tant, la seva eficiència.

4 Lliurament

El lliurament s'ha de fer via campus virtual abans de la data límit d'entrega, el diumenge 15 de novembre de 2015, a les 23:55. Heu d'enviar únicament l'arxiu de la vostra pràctica `grafs_nom_cognom1_cognom2_G.py`.

5 Criteris d'avaluació

- El programa dóna un resultat correcte per totes les entrades possibles: 50% de la nota
- Ús adequat del llenguatge i bon estil de programació: 25% de la nota
- Anàlisi d'eficiència: 25% de la nota

[1] `https://networkx.github.io/`