1 Camins i algorismes

Sovint, quan utilitzem un graf per modelitzar quelcom, ens interessa poder-hi fer algunes operacions. Podem, per exemple, voler trobar un camí entre dos punts, recòrrer el graf sencer o trobar el camí més curt per anar d'un vèrtex a un altre. Per aquest motiu utilitzem els camins, que trobarem o generarem mitjnçant diversos algorismes. En aquesta secció mostraré diverses maneres de recòrrer un graf, torbant la manera més eficient per a cada cas.

1.1 Grafs ponderats i dirigits

Grafs ponderats

Els grafs ponderats són grafs on cada aresta e està asociada a un nombre $w(e)$ anomenat pes o cost, tal que $w(e) \in \mathbb{R}$. El pes pot representar diverses quantitats, segons el que es vulgui modelitzar. Moltes vegades s'utilitza per representar distàncies, però si per exemple modelitzem una xarxa de distribució d'aigua, ens pot interessar representar el cabal de les canonades, o en una xarxa de bus, la densitat de trànsit de cada tram.

Grafs dirigits

Els grafs dirigits són grafs les arestes dels quals només admeten una direcció. D'aquesta manera, una aresta $e_0 = (v_0, v_1) \neq e_1 = (v_1, v_0)$, contràriament als grafs no dirigits. DE fet, no necessàriament ha d'existir una aresta contrària a una altra. Aquest tipus de grafs poden ser útils per representar carreteres, o moviments vàlids en algun joc.

1.2 Camins

Un camí p és una seqüència finita i ordenada d'arestes que connecta una seqüència ordenada de vèrtexs. Un camí p de longitud k (expressat com a $l(p) = k$) entre el vèrtex inicial v_0 i el vèrtex final v_k sempre que $v_0 \neq v_k$) és una successió de k arestes i $k+1$ vèrtexs de la forma $\overline{v_0, v_1}, \overline{v_1, v_2}, \cdots, \overline{v_{k-1}, v_k}$. Per definició, també es pot representar un camí p entre v_0 i v_k com a successió de vèrtex $p = v_0 v_1 \cdots v_k$. En auqest cas,pot ser tractat com un graf elemental P_n. Un cas especial és quan el camí comença i acaba al mateix vèrtex ($v_0 = v_k$). Llavors el camí és un cicle, i és l'equivalent a un graf cicle C_n. Quan un camí té totes les arestes diferents, s'anomena simple, i si a més té tots els vèrtexs diferents, s'anomena elemental.

En els grafs, ponderats, la longitud d'un camí $c = v_0, v_1, \cdots, v_n$ no es defineix pel nombre d'arestes per on passa el camí, sinó fent el sumatori dels pesos de les arestes

$$\mathbf{longitud}_w(c) = \sum_{i=0}^{n-1} w(\overline{v_i, v_{i+1}})$$

La distància entre dos vèrtexs v i u, $d_w(v,u)$, és la que s'obté al agafar la menor longitud d'entre tots els camins elementals entre v i u. (adjunjtar exemple de distància)

1.3 Algorismes

Un algorisme és un conjunt d'instruccions precises i ben definides que, donada una entrada, calculen la sortida corresponent segons les instruccions que té. A continuació

1.3.1 BFS

Aquest algorisme serveix per examinar l'estructura d'un graf o fer-ne un recorregut sistemàtic. La recerca per amplada prioritàtia (*breadht first serch* en anglès, d'aquí **BFS**) fa l'exploració en paral·lel de de totes les alternatives posibles per nivells des del vèrtex inicial. A la següent imatge es pot veure com funcionaria aquest algorisme en un graf: (Adjuntar imatge de BFS)

Per programar aquest algorisme s'acotuma a utilitzar un contenidor de tipus cua, que només permet afegir elements al final de la cua i treure'n de l'inici, sense poder accedir a elements del mig de la cua.

Algorisme 1: BFS

Data: un graf G i un node inicial v
Result: Seqüència de de nodes visitats
nova cua Q;
marquem v com a visitat;
imprimim(v);
posem v a la cua;
nou node *auxiliar*;
nou node *segent*;
while *la cua no estigui buida* **do**
  auxiliar = primer element de Q;
  imprimeix(*auxiliar*);
  elimina(primer element de Q);
  while *hi hagi nodes adjacents a auxiliar i aquests no s'hagin visitat* **do**
    marca adjacent(*auxiliar*) com a visitat;
    posa adjacent(*auxiliar*) a la cua;
  end
end
foreach *node de G* **do**
  marca'l com a no visitat
end

1.3.2 DFS