Part I

IntroducciÃ^3*alateoriadegrafs*

Aquest primer apartat consisteix en la part mÃ©s teÃ^2*ricadeltreball.Explicar*©*breumentlahist*^{2}

1 HistÃ^2*riadelgrafs*

1.1 Els primers passos

Tot sovint, les noves branques de la matemÃ tica sorgeixen de cercar solucions a problemes. Problemes que no poden ser resolts ni demostrats amb el que coneixem i que forcen a desenvolupar nous mÃštodes i teories. La teoria de grafs no n'Ã©s una excepciÃ^3*itotseguitpresentar*©*elsproblemesqu*

La teoria de grafs neix a partir de la soluciÃ³d'unproblemacuriÃ³s : elproblemadelspontsdeK¶nigberg(l'actualKali

"El riu Pregel divideix KÃ¶nigsberg en quatre parts separades, i connectades per set ponts. Ãs possible caminar per la ciutat passant per tots els ponts tan sols una vegada?"

Els ciutadans de KÃ¶nigsberg sabien que no era possible, perÃ²mainingºhohaviademostratfinsque
"A mÃ©s d'aquella part de la geometria que s'ocupa de les quantitats, la qual sempre ha generat un interÃ©s preferent, hi ha una altra part -encara prÃ cticament desconeguda- que ja fou esmentada per Leibniz amb el nom de geometria de posiciÃ³.Aquestapartdelageometriaestudiatotall²quepot©sserdeterminatºnicamentperlaposiciÃ³,a comlespropietatsdelaposiciÃ³;enaquestcamphomnohauriadepreocupar−sedequantitatsnidecomobte les.Noobstantaix²,elstipusdeproblemesquepertanyenaaquestageometriadeposiciÃ³ielsmtodesusatsp doncs,quandarrerementse'mvaplantejarunproblemaquesemblavab sicamentgeomtric,per²quenod ,hedeciditexposarac,comamostradelageometriadeposiciÃ³,elmtodequehededuïtperaresoldreproblem

Per aconseguir demostrar que el problema no tenia solÂ·luciÃ³,*Eulervahaverderepresentarelpro sticad'Eulerdel'espaiielteoremadepoliedresd'Euler(teoremaquedespr©svautilitzarperdemostrar*

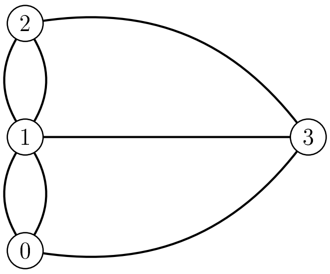


Figura 1: RepresentaciÃ³topol²gicadelspontsdeK¶nigsberg

Vandermonde i el tour del cavall d'escacs

A partir de l'article d'Euler, diversos matemÃ tics van comenÃ§ar a interessar-se pel camp de la topologia (o geometria de la posiciÃ³, comlideienenaquellmoment).Und'ellsfouAlexa Th©ophileVandermonde.Vandermondevatreballariestudiarelproblemadelscavalls, quepreguntaq

1.2 Les primeres descobertes i aplicacions

Un cop establert l'inici, Ã©s difÃcil veure com va continuar desesenvolupant-se la teoria, ja que en els primers treballs es volien resoldre problemes concrets sense establir relacions entre ells. Tot i aixÃ, durant el segle XIX es van plantejar una gran qÃŒantitat de problemes i es van desenvolupar molts teoremes referents els grafs.

Francis Guthrie

El 1852, Francis Guthrie, matemÃ tic britÃ nic, es planteja el segÃŒent problema mentre intenta pintar un mapa del Regne Unit:

> *"Ãs possible pintar qualsevol mapa de paÃ¯sos de tal manera que un paÃs tingui un color diferent al de tots els seus veÃ¯ns, utilitzant tan sols quatre colors?"*

D'aquest problema en surt el teorema a partir del qual s'sestableix que qualsevol mapa pot ser pintat Ãºnicament amb quatre colors diferents, de tal manera que dues regions adjacents (entenem com a adjacents dues regions que comparteixiin frontera, no tan sols un punt) no tinguin colors iguals. Aquest problema, que pot semblar tan trivial, no va ser demostrat fins l'any 1976. Va passar per mans de pioners com De Morgan, Hamilton, Cayley, Kempe (que va fer una demostraciÃ³publicadael1879), Heawood(quevaconclourequelademostraci³deKempenoeracorrecta) seenunprogramad'ordinador, novaacabardeconvncerlademostraci³.Aixdoncs, aquestproblemanova neundemostraci³.Enelstreballsd'AppeliHankenesvandefiniralgunsdelsconceptesifonamentsdel'c

Arthur Cayley

Arthur Cayley, matemÃ tic que treballava en la teoria de grups, topologia i combintÃ^2*ria, tamb©vaaportarunagranquantitatdeconeixementalateoriadegrafs.Vatreballaram*
que determina el nombre d'arbres expansius que tÃ© un graf complet de n vÃšrtex (veure apartat **??**). Una fÃ^3*rmulasemblantapareixiaentreballsdeCarlWilhelmBorchardt, enelsqu*

TambÃ© va treballar en el desenvolupament d'una representaciÃ^3*del'estructuraabstractad'ungr*
micsquecomparteixenf^3*rmulaocomposici*^{3}*per*^{2}*tenendiferentestructuramolecular), representantc*

William Hamilton i Thomas Kirkman

William Rowan Hamilton va plantejar un problema el 1859 que consistia a trobar un camÃ que passÃ©s pels 20 vÃšrtex d'un dodecÃ edre una sola vegada a travÃ©s de les seves arestes. Hamilton va comercialitzar el joc sota el nom de "The Icosian game" (Ã©s important dir que el nom de icosian no va ser degut a que utilitzÃ©s un icosaedre, sinÃ^3*quefeiareferenciaals20vrtexsdeldodecaedreperons'haviadepassar).E*

Gustav Kirchhoff

Gustav Kirchhoff, conegut majoritÃ riament en el camp de l'electrotÃšcnia per les lleis de Kirchhoff, tambÃ© va fer aportacions importants a la teoria de grafs. Les seves lleis, publicades el 1874, es basen en la teoria de grafs, perÃ^2*am©s, vaserelprimerd'utilitzarels*

1.3 Teoria de grafs moderna

Durant el segle XX, la teoria de grafs es va continuar desenvolupant. Amb les bases ja establertes durant el segle XIX, els matemÃ tics hi van comenÃ§ar a treballar i el 1936 DÃ©nes KÃ¶nig va escriure el primer llibre de teoria de grafs. Frank Harary va escriure un altre llibre el 1969, fent mÃ©s accessible la teoria de grafs. El desenvolupament de l'informÃ tica i les noves tÃšcniques de computaciÃ^3*vanpermetretreballarambgrafsamoltm©sgranescala, fentpossible, perexemple, laja*

Actualment la teoria de grafs Ã©s una part molt important de la matemÃ tica discreta i estÃ relacionada amb molts Ã mbits diferents, com per exemple la topologia, la combinatÃ^2*ria, lateoriadegrups, lageometriaalgebraica...Desdelseudesenvolupaments'har*
sica, laqumica, l'electr^2*nica, lestelecomunicacions, labiologia, lalogsticaifinsitotenl' mbitecon*^{2}*mic*

2 Principis bÃ sics

Un *graf* $G = (V, E)$ es defineix com un conjunt de *vÃšrtexs* (o nodes) $V = \{v_1, v_2, ..., v_n\}$ i un conjunt d'*arestes* $E = \{e_1, e_2, ..., e_m\}$, cadascuna de les quals uneix dos vÃšrtex de V. Si v_i i v_j, amb $v_i, v_j \in V$, estan units per l'aresta e_k

escriurem $e_k = \{v_i, v_j\}$. Quan volguem especificar el graf concret empararem les notacions $V(G)$ i $E(G)$. AixÃ doncs un graf estÃ format per un conjunt de punts i un conjunt d'arestes que uneixen alguns d'aquests punts. El nombre de vĂšrtexs d'un graf queda determinat pel nombre d'elements que hi ha en el conjunt V, per tant ens referirem a ell com a $|V|$ (cardinal de V). Amb les arestes passa el mateix, i tambĂ© utilitzarem $|E|$ per indicar el nombre d'arestes d'un graf. Definim tambĂ© que dos vĂšrtexs sÃ^3*nadjacentssiest nunitsperunaarestai, comaconseqncia, s*^{3}*nincidentsa*
Laideaintuitivadegrafconvidaautilitzarrepresentacionsgr fiques.Aix, enlafigura??*esmostraung*

$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$

$$E = \{\{v_1, v_2\}, \{v_1, v_4\}, \{v_1, v_5\}, \{v_2, v_5\}, \{v_3, v_4\}, \{v_3, v_5\}, \{v_3, v_6\}, \{v_4, v_6\}\}$$

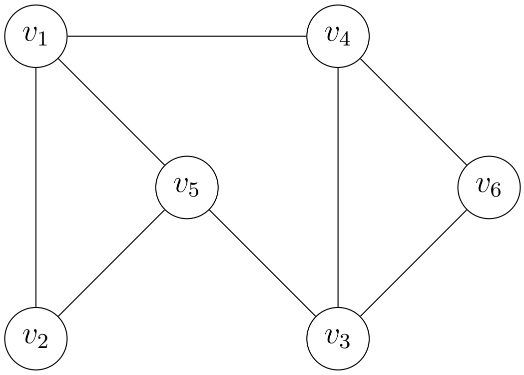


Figura 2: RepresentaciÃ^3*gr ficadelgraf*G=(V,E)*ques'hapresentat*

Si una aresta comenÃ§a i acaba en el mateix vĂšrtex (per exemple $e_m = \{v_i, v_i\}$ s'anomena llaÃ§ (figura **??**, cas (a)). TambĂ© pot ser que hi hagi dues arestes idĂšntiques, Ă©s a dir, dues arestes que uneixin v_i i v_j (figura **??**, cas (b)). En qualsevol d'aquests dos casos anteriors, el graf s'anomena *multigraf* o *pseudograf*. En cas contrari, el graf serÃ anomenat *simple*. Amb el que hem vist fins ara, podem dir que $e_1 = \{v_1, v_2\}$ Ă©s equivalent a $e_2 = \{v_2, v_1\}$ (ja que es tracta de parells no ordenats). Tanmateix, existeixen grafs en els quals les arestes han de ser recorregudes en una direcciÃ^3*determinada.S'anomenengrafsdirigitsi, enaquestcas, si*$e_1 =$ (v_1, v_2) i $e_2 = (v_2, v_1)$, $e_1 \neq e_2$, ja que es tracta de parells ordenats (figura **??**, cas (c)). En la segĂŒent imatge es mostren els grafs esmentats anteriorment:

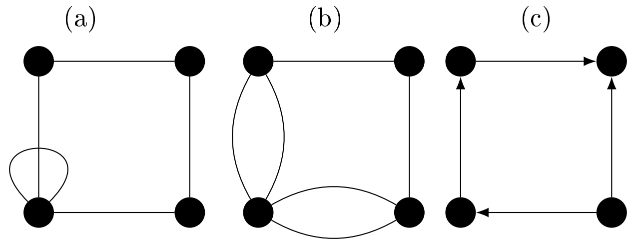


Figura 3: Graf amb un llaÃ§ (a), graf amb arestes mÃºltiples (b) i graf dirigit (c)

Nota de l'autor: a partir d'ara, i si no s'indica el contrari, quan es parli de grafs, s'exclouran els multigrafs i grafs dirigits.

Direm que un graf no dirigit $G = (V, E)$ Ã©s *connex* si per a qualsevol $v_i, v_j \in V$ $v_i \neq v_j$ existeix un *camÃ* (successiÃ³*d'arestes*)*queelsuneix.*
*Elnombred'arestesques*³*nincidentsaunvrtex*v(*comptantelsllaoscomaduesarestes*)*determinenelq*
*nimd'ungraf*G=(V,E)*quedadeterminatdelasegentmanera* $:\delta(G) = min\{g(v) : v \in V(G)\}$. De manera similar, el grau mÃ xim de G, $\Delta(G) = max\{g(v) : v \in V(G)\}$.

Si d'un graf connex en treiem una aresta o un node, en resulta un altre (o mÃ©s d'un) graf connex. Si $v \in V(G)$, indicarem per $G - v$ al graf que s'obtÃ© al suprimir el vÃšrtex v i totes les seves arestes incidents. De la mateixa manera, si $e \in E(G)$, $G - e$ indicarÃ el graf que s'obtÃ© a l'eliminiar la aresta e.

Amb tots aquests conceptes ja podem veure el teorema d'Euler, un dels primers teoremes en teoria de grafs i un dels mÃ©s importants.

Teorema 1 (Euler)

En tot graf $G = (V, E)$, la suma dels graus dels vÃšrtex Ã©s igual al doble del nombre d'arestes.

$$\sum_{v \in V(G)} g(v) = 2|E|(G)$$

*DemostraciÃ*³ : *NomÂ©scalveurequecadaarestatÂ©dosextremsiquesumadoseneltotaldelsgraus.Lad*

- *Si* $|E| = 0$, *no cal considerar el cas. Si* $|E| = 1$, *o* $|V| = 2$ *i cada vÃšrtex tÃ© grau 1 o la aresta Ã©s un llaÃ§ i hi ha un sol vÃšrtex de grau 2. En qualsevol d'aquests dos casos, el teorema es verifica.*

- *Ara suposem que el teorema estÃ demostrat per a* $|E| \leq k$ *i que* G *Ã©s un graf amb* $|E| = k + 1$. *Si* e *Ã©s una aresta de* G *prenem* $H = G - e$.

- *Llavors tots els vÃšrtexs de H tenen el mateix grau a H que a G excepte 2 que tenen un grau menys o un que tÃ© dos graus menys (nomÃ©s en el cas que e siguÃ©s un llaÃ§). En tots dos casos obtenim que:*

$$\sum_{v\in V(G)} g(v) = \sum_{v\in V(H)} g(v) + 2 = 2(|E|-1)+2 = 2|E|$$

D'aquesta demostraci$\tilde{A}^3 entreiemunaaltraafirmaci^3$:

En tot graf, el nombre de vÃšrtex amb grau imparell, Ã©s parell.

3 Tipus de grafs

Fins ara hem vist els conceptes bÃ sics en teoria de grafs i algunes de les propietats que compleixen tots els grafs. No obstant, existeixen diversos tipus de grafs que tenen propietats especials. A continuaci$\tilde{A}^3 se'npresentenalguns$

3.1 Grafs complementaris

El graf complementari del graf G Ã©s el graf H amb els mateixos vÃšrtexs que G, de manera que dos vÃšrtexs d'H seran adjacents si i nomÃ©s si a G no ho s$\tilde{A}^3 n.Formalment, si G=(V,E)©sungrafsimpleiK=E(K_n)$ *Ã©s el conjunt d'arestes derivades de totes les possibles combinacions de dos elements de V, essent $n = |V(G)|$, llavors $H = (V, K \setminus E)$[1]. Per indicar el complementari de G s'escriu $\bar{G}$ o G'.*

Propietat 1 *Per obtenir el graf complementari de G tan sols s'han de posar les arestes que falten per obtenir un graf complet i treure totes les que hi eren inicialment (ja que $|E(G)| + |E(\bar{G})| = |K|$). Veure figura* **??**.

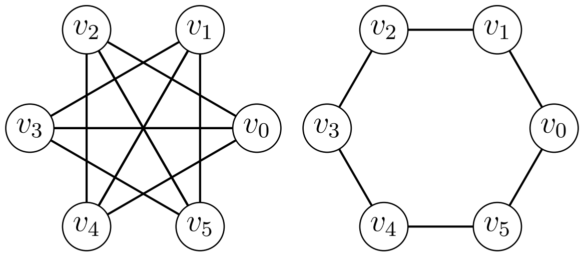


Figura 4: Un graf G i el seu complementari $\bar{G}$

[1]La notaci$\tilde{A}^3$K$\setminus E$ indica el conjunt dels elements de K que no s$\tilde{A}^3 na$E

3.2 Grafs regulars

Es diu que un graf $G = (V, E)$ Ã©s regular de grau r quan tots els seus vÃšrtexs tenen grau r. Formalment, un graf Ã©s r-regular quan $\Delta(G) = \delta(G) = r$. Un graf 0-regular Ã©s un graf nul [2]*, un graf 1-regular consisteix en arestes separades entre elles i un graf 2-regular consisteix en un o mÃ©s cicles* [3] *separats. A partir d'aquÃ, els grafs regulars mÃ©s importants tenen noms propis, com per exemple els 3-regulars, que sÃ³nelscÃºbics; els4−regulars, quÃ rtics; els7−regulars, grafs de Witt truncats dob*

Propietat 1 *No necessÃ riament existeix un Ãºnic graf r-regular, sinÃ³quesovints′enpodenferamb*

Propietat 2 *Com a conseqÃŒÃšncia del teorema **??**, per a tots els grafs r-regulars amb n vÃšrtexs es compleix que*

$$|E|(G) = \frac{1}{2}nr$$

on $|E|(G)$ Ã©s el nombre d'arestes.

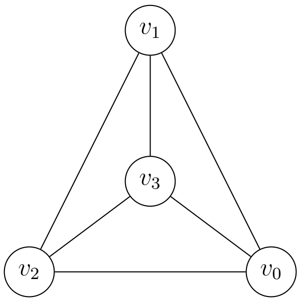


Figura 5: Graf cÃºbic (3-regular)

3.3 Graf nul i grafs buits

Els grafs buits sÃ³ngrafssensearestes, ©sadir, conjuntsd′nv©rtexs.S³nelscomplementarisdelsgr [4]*, i per tant la seva nomenclatura Ã©s $\bar{K}_n$ o, simplement, N_n. Estrictament s'anomena graf nul a N_0 i buits a la resta, perÃ²comquenormalmentnos′utilitzaN$_0$, convencionalment es diuen nuls tots els elements del conjunt dels buits.*

[2] Veure apartat **??**
[3] Veure apartat **??**
[4] Veure apartat **??**

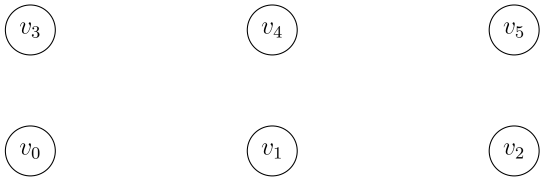


Figura 6: Graf buit N_6

3.4 Grafs complets

Un graf complet Ã©s un graf on cada vÃšrtex estÃ unit a tots els altres una sola vegada. Un graf complet amb n nodes Ã©s un graf simple i $(n-1)$-regular i la seva nomenclatura Ã©s K_n.

Propietat 1 *Els grafs complets tenen $\binom{n}{2} = \frac{n(n-1)}{2}$ arestes.*

Propietat 2 *El nombre de cicles que contÃ© un graf complet queda determinat per la segÃŒent igualtat:*

$$C_n = \sum_{K=3}^{n} \frac{1}{2} \binom{n}{k} (k-1)!$$

On:

$\frac{1}{2}$ *es multiplica perquÃš es contes dues vegades els cicles, ja que no sÃ³ndirigits.©selnombredegrups*

$(k\binom{n}{k} 1)!$ *Ã©s el nombre de permutacions circulars de k elements.*

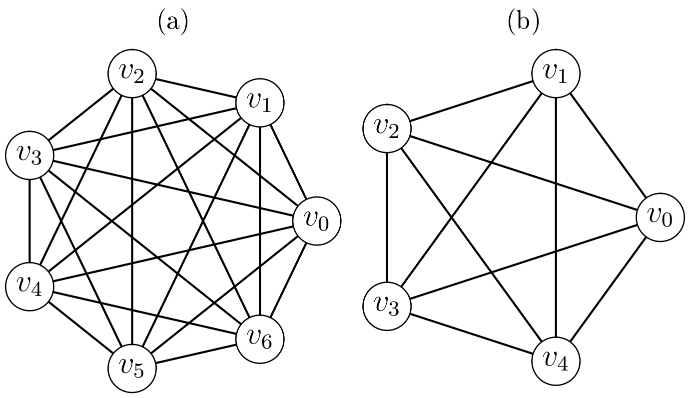


Figura 7: Grafs complets K_7(a) i K_5(b)

3.5 Cicles

Els cicles $s\tilde{A}^3ngrafs2 - regularsambnvrtexsinarestes, is'anomenenC_n.$

Propietat 1 *El graf lineal* [5] *d'un cicle Ã©s ell mateix.*

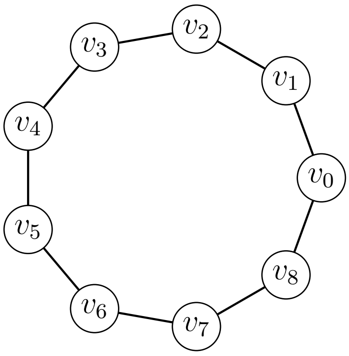


Figura 8: Cicle C_9

3.6 Grafs bipartits

Els grafs bipartits $s\tilde{A}^3naquellsenelsqualselsvrtexsespodensepararendosconjuntsdisjunts\,U i\,W($©$s$

Propietat 1 *Tots els grafs* C_n *amb* n *parell,* $s\tilde{A}^3ntamb$©$grafsbipartits.$

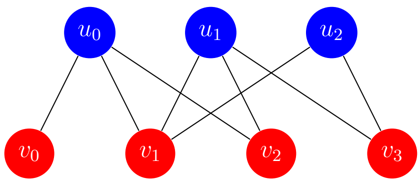


Figura 9: Graf bipartit

3.7 Grafs bipartits complets

Els grafs bipartits complets $s\tilde{A}^3ngrafsbipartits^6enelsqualscadaelementdelconjunt\,Uest\;unitatotsels$
on $m = |U|$ *i* $n = |W|$.

Propietat 1 *En els grafs bipartits* $K_{m,n} = K_{n,m}$, $|V| = n + m$ *i* $|E| = mn$.

---

[5] Veure apartat**??**
[6] Veure apartat **??**

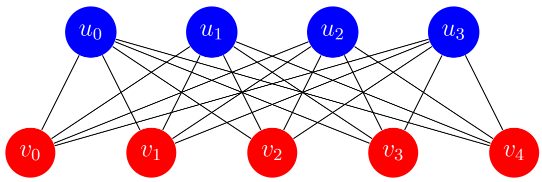


Figura 10: Graf bipartit complet $K_{4,5}$

3.8 Grafs estrella

Un graf estrella de grau n Ã©s aquell que contÃ© un vÃšrtex amb grau $n-1$ i els $n-1$ vÃšrtexs restants de grau 1. La seva nomenclatura Ã©s S_n.

Propietat 1 *SÃ³nestrellestotselsgrafsbipartitsdelaforma$K_{1,n-1}$ o $K_{n-1,1}$.*

Propietat 2 *El graf lineal de S_n Ã©s K_{n-1}.*

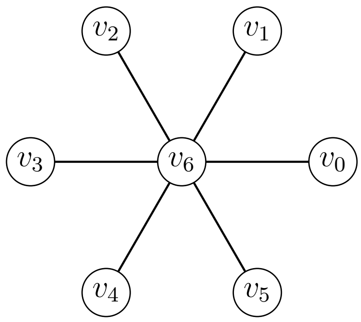


Figura 11: Graf estrella S_7

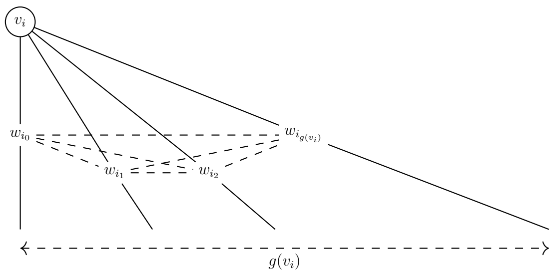
3.9 Graf lineal

Un graf lineal $L(G)$ d'un graf G Ã©s un graf que representa les adjacÃšncies entre les arestes de G. Formalment, donat un graf G, el graf lineal $L(G)$ Ã©s aquell en el qual cada vÃšrtex correspon a una aresta de G i dos vÃšrtexs sÃ³nadjacentsnom©ssilesarestescorresponentsaGcomparteixenunvrtex.

Propietat 1 *El graf lineal d'un graf amb n nodes, e arestes i amb vÃšrtexs de graus $g(v_i)$ tÃ© $n' = e$ nodes i*

$$e' = \frac{1}{2}\sum_{i=1}^{n} g(v_i)^2 - e$$

arestes.
DemostraciÃ³ : *Cadanode*v_i amb grau $g(v_i)$ del graf original generarÃ un graf complet de $g(v_i)$ nodes ($K_{g(v_i)}$).



Tal com s'ha vist a l'apartat **??**, un graf complet tÃ© $\binom{n}{k} = \frac{n(n-1)}{2}$, per tant en aquest cas se'n generen $\frac{g(v_i)(g(v_i)-1)}{2}$. Per$\tilde{A}^2 aix^2 escompleixperacadavrtex, illavorspodemescriure \sum_i^n$
$1) = \frac{1}{2}\sum_{i=1}^{n}(g(v_i)^2 - g(v_i)) = \frac{1}{2}\sum_{i=1}^{n} g(v_i)^2 - \underbrace{\frac{1}{2}\underbrace{\sum_{i=0}^{n} g(v_i)}_{2|E|}}_{|E|} = \frac{1}{2}\sum_{i=1}^{n} g(v_i)^2 - e$

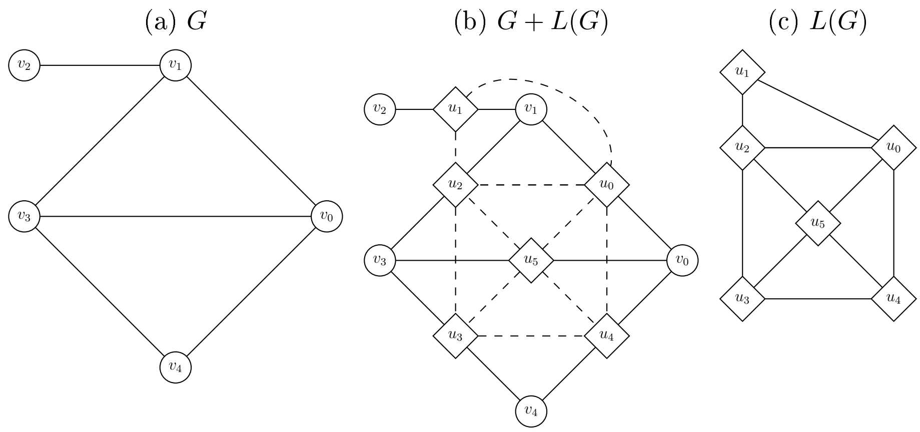


Figura 12: ProcÃ©s de construcci$\tilde{A}^3 d'ungraflineal$

3.10 Rodes

Un *graf roda* amb n vÃšrtex Ãšs un graf que contÃ© un cicle de longitud $n-1$, on tots els vÃšrtex del cicle estan connectats a un vÃšrtex fora del cicle, anomenat node central. S'escriu W_n, i a vegades simplement s'estudia com a $C_{n-1} + K_1$.

Propietat 1 El node central tÃ© grau $n-1$, i la resta de nodes tenen grau 3.

Propietat 2 El nombre de cicles que contÃ© un graf roda amb n vÃštrexs estÃ determinat per $n^2 - 3n + 3$.
$Demostraci\tilde{A}^3 : Elnombredeciclesquecont©ungrafroda©slasumadelnombredeciclesdelongituddes$

Amb l'exepciÃ^{3}deC_{n-1}, el nombre de cicles per a cada i Ã©s $n-1$. En el cas de C_{n-1} n'hi ha n de possibles, perÃ^{2}horepresentemdelamaneran-1+1,pertalquesiguim©sc^{2}modeoper 2longitudspossibles.D'aquestamanerapodemescriure(n-2)(n-1)+1= $\boxed{n^2-3n+3}$.

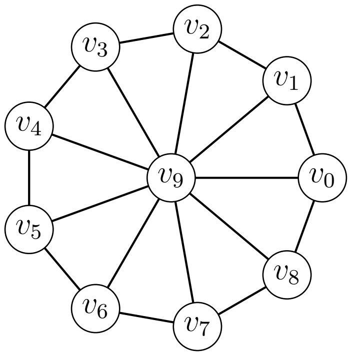


Figura 13: Roda W_{10}

3.11 Xarxes

Els *grafs xarxes* bidimensionsals $G_{m,n}$ sÃ^3*ngrafsbipartitsqueformenunentramatenformadecuadr-culademnvrtexs.*

Propietat 1 Es pot greneralitzar per a xarxes de mÃ©s dimensions com a $G_{m,n,o,\dots}$.

Propietat 2 Un graf xarxa tÃ© mn vÃšrtexs i $(m-1)n+(n-1)m$ arestes.

DemostraciÃ^3 : Talcomespotveurealafigura??, *ungrafxarxat©*$(n-2)(m-2)+2(m-2)+2(n-2)+4 = nm-\cancel{2m}-\cancel{2n}+\cancel{4}+\cancel{2m}-\cancel{4}+\cancel{2n}-\cancel{4}+\cancel{4} = mn$vÃšrtexs. D'altra banda, la suma dels graus de tots els vÃšrtexs Ã©s

$$\frac{4(m-2)(n-2)+3\times 2(m-2+n-2)+4\times 2}{2} = 2(m-2)(n-2)+3(m-2)+3(n-2)+4$$

Si es desenvolpua, s'obtÃ©

$$(n-2)(m-2)+(n-2)(m-2)+2(m-2)+(m-2)+2(n-2)+(n-2)+4$$

on la suma dels elements marcats equival a mn. Si es continua desenvolupant tinguent en compte aquest detall, s'obtÃ©

$$mn-2m-2n+4+m-2+n-2+mn = 2mn-m-n = \boxed{m(n-1)+n(m-1)}$$

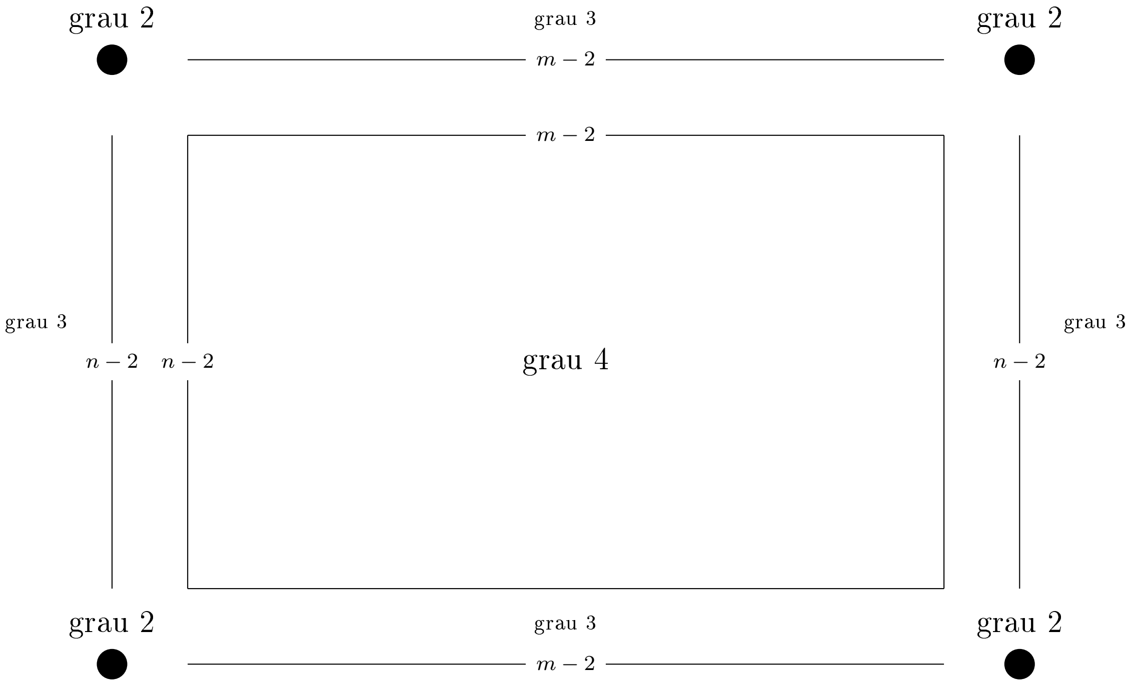


Figura 14: RepresentaciÃ³*delsgrausdelsdiferentsnodesd′ungrauxarxa*

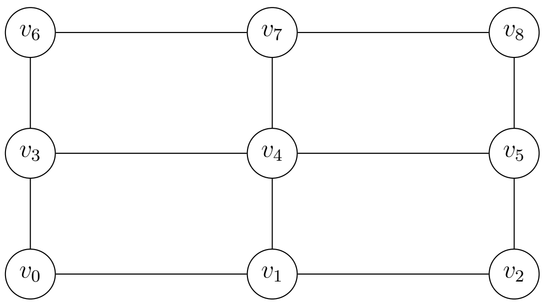


Figura 15: Graf xarxa $G_{3,3}$

3.12 Arbres

Els *arbres* sÃ³*nuntipusmoltimportantdegrafs* : *s*³*ngrafsconnexossensecicles, demaneraqueexiste*
entredosvrtexs.

Propietat 1 Si a un arbre se li afegeix una aresta, es genera un cicle, i se s'en treu una, el graf deixa de ser connex.

Propietat 2 Hi ha un tipus especial d'arbres anomenats *elementals o camins*, que sÃ³*nelsarbresamb*|V|=1,|V|=2*iengeneraltotsaquellson*$\delta = 1$ i $\Delta = 2$. S'anomenen P_n, on $n = |V|$. TambÃ© es pot pensar en grafs elementals com a $G_{n,1}$.

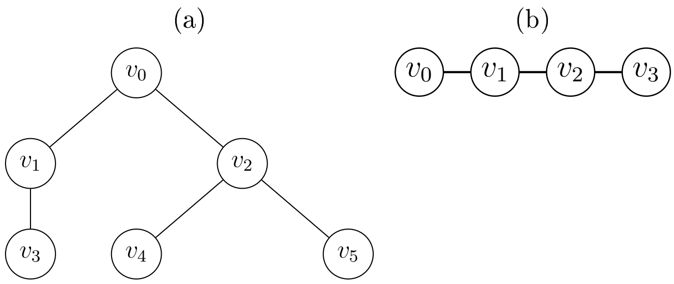


Figura 16: Arbre (a) i graf elemental P_4 (b)

Teorema 2

Un arbre amb n nodes tÃ© $n - 1$ arestes.

DemostraciÃ³ : SicomprovemelcasonunarbreTtÃ©n=1vrtexs, veiemquenotÃ©caparesta, pertantelte
i T_2. Per hipÃ²tesi, T_1 tindrÃ $v_1 = |V|(T_1)$ vÃšrtexs i $n_1 - 1 = |E(T_1)|$ arestes, i T_2 tindrÃ $n_2 = |V(T_2)|$ vÃšrtexs i $n_2 - 1 = |E|(T_2)$ arestes. Llavors el nombre d'arestes de T Ã©s $(n_1 - 1) + (n_2 - 1) + 1 = (n_1 + n_2) - 1$. DeduÃ¯m llavors que el nombre de nodes de T Ã©s $n_1 + n_2$. Dit d'una altra manera, $|E(T)| = |V(T)| - 1$.

Com a conseqÃŒÃšncia d'aquest teorema, en podem arribar a un altre:

Teorema 3

En un arbre T amb $|V| \geq 2$, hi ha com a mÃnim dos vÃšrtexs de grau 1 (anomenats fulles).

DemostraciÃ³ : *ComaconseqnciadelTeorema1iTeorema2podemdirqueenunarbre* $\sum_{v \in V} g(v) = 2|E| = 2|V| - 2$TambÃ© podem deduÃ¯r que si tots els vÃšrtexs tinguessin un grau $g(v_i) > 1$, llavors

$$\sum_{v \in V} g(v) \geq \sum_{v \in V} 2 = 2|V|$$

Sabem que aixÃ²*noÃ©scorrecte, jaqueestemdientque*2|V|-2$\geq 2|V|$. Amb aixÃ²*japodemveurequenec*
*nimdosgraus, per*²*demostremtambÃ©queambunsolnodedegrau1tampocÃ©ssuficient.Suposemaraqu*
nimgrau2.Llavors$2|V| - 2 = \sum_{v \in V} g(v) = 1 + \sum_{\substack{v \in V \\ \text{si } g(v) \neq 1}} g(v) \geq 1 + \sum_{\substack{v \in V \\ \text{si } g(v) \neq 1}} 2 = 1 + 2(|V| - 1) = 2|V| - 1$Ara estem dient que $2|V| - 2 \geq 2|V| - 1$, que torna a ser una contradicciÃ³.*SabemquesitreiemungraumÃ©s, elteoremaescomplir , ihemdemostratqueelm-nimnombredevrtexsdegrau1Ã©s2.*

Hi ha un altre teorema que relaciona el nombre de fulles amb el grau mÃ xim d'un arbre:

Teorema 4

El nombre de fulles d'un arbre T Ã©s mÃ©s gran o igual a Δ.

DemostraciÃ³ : SieliminemelnodedegrauΔ *de l'arbre, juntament amb totes les seves arestes incidents, obtenim un conjunt de* Δ *grafs. Si alguns d'aquests grafs consisteixen en tan sols un node, vol dir que abans eren adjacents al node que hem eliminat, per tant, nomÃ©s tenien grau 1. Si, pel contrari, formen nous arbres, pel teorema 3 podem dir que hi haurÃ com a mÃnim dues fulles. Encara que hi ha la possibilitat que un dels nodes amb grau 1 siguÃ©s l'adjacent a el node que hem tret, sempre podem garantir que hi ha com a mÃnim una fulla. Per tant, tambÃ© podem garantir que que hi haurÃ com a mÃnim* Δ *fulles.*

Part II
Camins i algorismes

Sovint, quan utilitzem un graf per modelitzar quelcom, ens interessa poder-hi fer algunes operacions. Podem, per exemple, voler trobar un camÃ entre dos punts, recÃ²rrerelgrafsencerotrobarelcamm©scurtperanard'unvrtexaunaltre.Peraquestmotiuutilitzem

En aquesta secciÃ³mostrar©diversesmaneresderec²rrerungraf,torbantlamaneram©seficien

3.13 Grafs ponderats i dirigits

Grafs ponderats

Els grafs ponderats sÃ³ngrafsoncadaarestaeest asociadaaunnombrew(e)anomenatpesocost,talque
$\mathbb{R}$. *El pes pot representar diverses quantitats, segons el que es vulgui modelitzar.*
Moltes vegades s'utilitza per representar distÃ ncies, perÃ²siperexemplemodelitzemunaxarxadedist

Grafs dirigits

Els grafs dirigits sÃ³ngrafslesarestesdelsqualsnom©sadmetenunadirecci³.D'aquestamanera,una $(v_0, v_1) \neq e_1 = (v_1, v_0)$. *De fet, no necessÃ riament ha d'existir una aresta contrÃ ria a una altra.*

Aquest tipus de grafs poden ser Ãºtils per representar carreteres, o moviments vÃ lids en algun joc.

3.14 Camins

Un camÃ p Ã©s una seqÃŒÃšncia finita i ordenada d'arestes que connecta una seqÃŒÃšncia ordenada de vÃšrtexs. Un camÃ p de longitud k (expressat com a $l(p) = k$) entre el vÃšrtex inicial v_0 i el vÃšrtex final v_k sempre que $v_0 \neq v_k$) Ã©s una successiÃ³dekarestesik+1vrtexsdelaforma$\overline{v_0, v_1}, \overline{v_1, v_2}, \cdots, \overline{v_{k-1}, v_k}$
. Per definiciÃ³, tamb©espotrepresentaruncampentrev_0 *i* v_k *com a successiÃ³devrtexp=*$v_0 v_1 \cdots v_k$.
En aquest cas, pot ser tractat com un graf elemental P_n.

Un cas especial Ã©s quan el camÃ comenÃ§a i acaba al mateix vÃšrtex ($v_0 = v_k$)*. Llavors el camÃ Ã©s un cicle, i Ã©s l'equivalent a un graf cicle* C_n.

Quan un camÃ tÃ© totes les arestes diferents, s'anomena simple, i si a mÃ©s tÃ© tots els vÃšrtexs diferents, s'anomena elemental.

En els grafs, ponderats, la longitud d'un camÃ $c = v_0, v_1, \cdots, v_n$ *no es defineix pel nombre d'arestes per on passa el camÃ, sinÃ³fentelsumatoridelspesosdelesarestes**longitud***$_w(c)$
$\sum_{i=0}^{n-1} w(\overline{v_i, v_{i+1}})$

La *distÃ ncia* entre dos vÃšrtexs v i u, $d_w(v, u)$, Ã©s la que s'obtÃ© al agafar la menor longitud d'entre tots els camins elementals entre v i u. (adjunjtar exemple de distÃ ncia)

4 Estructures de dades dels grafs

En els grafs, normalement s'ha de tractar una gran quantitat d'informaciÃ³ :
nodes, arestes, pesos, sentits...Totaaquestainformaci³no©sdifcildegestionarsielgrafques'estudic
laenestructuresdedades.Lesestructuresdedadessolenserutilitzadesenlaprogramaci³, per²enaquest
nefinsitotquannoestreballaambinform tica.

Matrius

Matriu d'adjacÃšncia

La matriu d'adjacÃšncia d'un graf Ã©s una matriu quadrada que contÃ© informaciÃ³*respecteelnor*

Si G Ã©s un graf amb n nodes, $A(G) = (a_{i,j})_{i,j} = 1, ..., n$ Ã©s la seva matriu d'adjacÃšncia de n x n, on $a_{i,j}$ correspon al nombre d'arestes que uneixen els nodes i, j, contant com a 2 els llaÃ§os, que sempre es trobarÃ n a la diagonal. La matriu serÃ simÃštrica si el graf ho Ãšs, i podrem conÃšixer el grau d'un node i fent el sumatori de les caselles de la i-Ãšsima fila. A vegades, quan s'utilitzen grafs ponderats, les matrius d'adjacÃšncia s'omplen amb els pesos de les arestes, per no haver d'utilitzar altres estructures per emmagatzemar la resta d'informaciÃ³.*Enaquesttipusdegrafs, elslla§osnonecess riamentvaldr n2, per²espodr ndiferenciar*
espais, i en grafs poc densos, la majoria d'espais estan ocupats per 0, de tal manera

que s'estÃ utilitzant molta memÃ^2*riaquenocont©informaci*^{3o}*til.Delamateixamanera, siafegimu*
1espaisalamatriu.

Matriu d'incidÃšncia

La matriu d'incidÃšncia d'un graf G sense llaÃ§os, $I(G) = (b_{i,j})_{i=1,\ldots,|V|(G),j=1,\ldots,|E|(G)}$, Ã©s la matriu binaria de $|V|(G) \times |E|(G)$ on $b_{i,j}$ indica si la aresta j Ã©s incident al node i.

Llistes d'adjacÃšncia

Aquesta estructura de dades Ã©s molt utilitzada per tractar grafs, ja que ocupa menys memÃ^2*riainom©sincloul'informaci*^{3}*necess ria.Lesllistesd'adjacnciaconsisteixenenuncon*
moltm©ssenzillferiteracions.Ambl'esquemasegentespotveurem©sclarament :

(afegir esquema de les llistes d'adjacÃšncia)

4.1 Algorismes

Un algorisme Ã©s un conjunt d'instruccions precises i ben definides que, donada una entrada, calculen la sortida corresponent segons les instruccions que tÃ©. A continuaciÃ^3*s'enmostrenunsquantsd'importants.*

4.1.1 BFS

Aquest algorisme serveix per examinar l'estructura d'un graf o fer-ne un recorregut sistemÃ tic. La recerca per amplada prioritÃ tia (*breadht-first serch* en anglÃšs, d'aquÃ **BFS**) fa l'exploraciÃ^3*enparal*Δ*leldedetoteslesalternativesposiblespernivellsdesdelvrtexini*
(AdjuntarimatgedeBFS)

Per programar aquest algorisme s'acotuma a utilitzar un contenidor de tipus cua, que nomÃ©s permet afegir elements al final de la cua i treure'n de l'inici, sense poder accedir a elements del mig de la cua. El que farÃ aixÃ^2©*sb sicamentimprimirperpantallalas*

Aquesta Ã©s una manera bastant usual de programar el BFS, i encara que Ãšs eficient, s'estÃ desaprofitant propietats de l'algorime. Amb BFS es pot saber a quina distÃ ncia del punt inicial estÃ cada node, el camÃ mÃ©s curt per anar del node inicial a qualsevol altre i fins i tot es pot generar un arbre expansiu mÃnim, agafant les arestes per on passa el BFS. El segÃŒent algorisme tÃ© en compte aquests detalls. EstÃ pensat per ser implementat en el llengÃŒatge Python, i per aquest motiu utilitza diccionaris (llistes on cada element tÃ© un nom i una clau), perÃ^2*enllenguatgesbasatsenC, espodenutilitzarmapsdelamateixamanera.*

Encara que aquest algorisme sembli molt senzill, ens pot aportar informaciÃ^3*important, ifinsitot*
m©scurtentredosnodes.Aquestalgorismes'utilitzatamb©peroperacionsm©scomplexes, comlesse

Algorisme 1: BFS

Data: Un graf G i un node inicial v
Result: SeqÃŒÃšncia de de nodes visitats
nova cua Q
marca v com a visitat
imprimeix(v)
afegeix v a la cua Q
nou node *auxiliar*
nou node *següent*
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
        afegeix adjacent(*auxiliar*) a la cua
    end
end
foreach *node de G* **do**
    marca'l com a no visitat
end

Google l'utilitza per indexar pÃ gines web noves al seu buscador. Amb BFS pot recÃ²*rrertotalaxarxad′internetsencera, i, sicadap ginaweb©sunnodeicadaenlla§©sunac hoest , l′algorismetrobar elnounode).Lesxarxessocialsl′utilitzenpersuggeriramistats.AmbB*

Es pot silÂ·lucionar un cub de rubik amb aquest algorisme. Si s'aconsegueix generar un graf on cada node sigui un estat diferent del cub i les arestes siguin un moviment d'una cara, donat un estat inicial, amb BFS arribes a l'estat resolt amb els mÃnims moviments possibles.

4.1.2 DFS

La recerca per profunditat prioritÃ ria (*depth-first search* en anglÃšs, d'aquÃ **DFS**) Ã©s un algorime que utilitza uns principis semblants al BFS, perÃ²*enllocdecobrirtotal′ampla (adjuntarimatgedeDFS)TalcomenelBFS, tamb©hihadiversesmaneresdeferl′algorisme, ienpres*
Aquest mÃ©tode, perÃ², *t©unproblema, i©squenom©sfuncionaperagrafsnodirigits.Hihala*

```
Data: Un graf G i un node inicial v
Result: SeqÃŒÃšncia de nodes visitats, distÃ ncia de cada node resperce v
nou diccionari dist
dist[v] = 0
nou diccionari anterior
anterior[v] = Nul
i = 0
nova llista frontera afegeix v a frontera
imprimeix(v)
while frontera no estigui buida do
    nova llista següent
    foreach node x de frontera do
        /* A cada iteraciÃ³,xagafar unvalordiferentdefrontera        */
        foreach node y adjacent a x do
            if y no existeix dins dist then
                dist[y] = i
                anterior[y] = x
                afegeix y a següent
                imprimeix(y)
            end
        end
        end
        frontera = següent
        i = i + 1
    end
    imprimeix(dist)
```

```
Data: Un graf G
Result: SeqÃŒÃšncia de nodes visitats des de cada node
nou diccionari anterior
nova llista ordre
foreach node u del graf do
    if u no existeix dins anterior then
        imprimeix(u)
        anterior[u] = Nul
        DFSrecursiu(G, u)
    end
end
inverteix ordre
imprimeix(ordre)

/* La funciÃ³DFSrecursiuquedadeterminadapelsegentalgorisme :        */
FunciÃ³ DFSrecursiu(G, v)
    foreach node x adjacent a v do
        if x no existeix a anterior then
            imprimeix(x)
            anterior[x] = Nul
            DFSrecursiu(G, x)
        end
    end
    /* NomÃ©s si es vol obtenir la seqÃŒÃšncia de
       recursiÃ³oordenaci³topol²gicaperagrafsdirigitsacclics,        */
```

Algorisme 2: DFS

Data: Un graf G i un node inicial v
Result: SeqÃŒÃšncia de nodes visitats
nova pila S
nou node *següent*
marca v com a visitat
imprimeix(v)
afegeix v a la pila S
while *la pila no estigui buida* **do**
  següent = node adjacent no visitat de l'element superior de S
  /* En cas que no n'hi hagi cap, *següent* = *Nul* */
  if *següent* = *Nul* **then**
    elimina(element superior de S)
  else
    marca *següent* com a visitat
    imprimeix(*següent*)
    afegeix *següent* a S
  end
end

Aquest algorisme no tÃ© tantes utilitats prÃ ctiques com el BFS, perÃ²*tamb©t©propuietatsuti*
Es pot saber si un graf tÃ© cicles, comprovant si quan estem a l'iteraciÃ³*d′unnode*(*encaranon′*

Es pot dur a terme una ordenaciÃ³*topol*²*gica*, *siestractad′ungrafdirigitsencecicles*.*Unexempl*
neunadeterminada, *calhaver*−*nefetprimerunaaltra*.*AmbelDFS*, *podemobtenirunadelesseqn*

Resulta que, a la secciÃ³**??**, *quanexplicavauntipusconcretdegraf*, *sovintutilitzavaaltrestipusdegra*

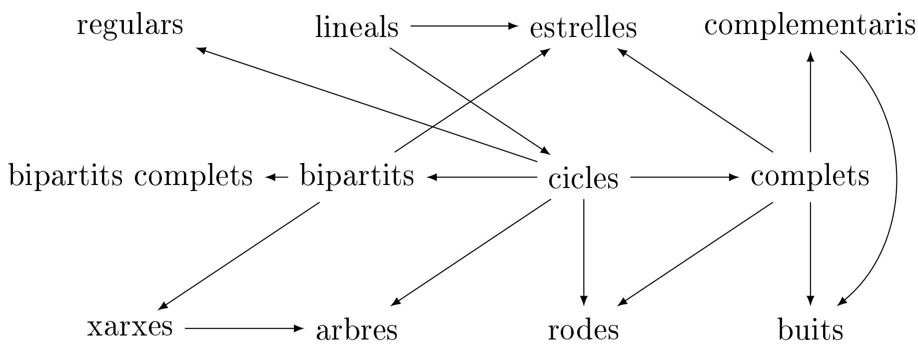


Figura 17: Graf de les dependÃšncies de cada apartat

4.1.3 Dijkstra

Aquest algorisme, desdenvolupat per Edsger W. Dijkstra el 1956, serveix per trobar el camÃ mÃ©s curt entre dos nodes d'un gaf ponderat. De fet, aixÃ²©*selquefeialavariantoriginal, trobaelcamm©scurtentreunnodeinicialitotalarestadenodesdelsgraf.Totiaix²espotmodificarlleug m©scurtentredosnodesespecificats.Elquefaelprograma©ssuposarquiness³nlesdist nciesm-nimesdesdelnodeinicialfinslaresta,ivadescobrintelgraffinsquepotassegurarelcam-m©scurt.Alprincipi,suposaqueelmovimentambmenyscost©snomoure's(aix²ser certsielgrafnoco nimesentrel'inicialiaquests©ssimplementl'arestaqueelsuneix.Espotassegurarqueser certpelnode ,aquestseriam©sllarg,jaqueperfor§aalgunadelesarestesambm©spesques'handescartathadefor alternatiu,ilasumaser sempresuperior).Uncophihaelprimernodeambmnimpesassegurat(ipertant),esbusquenelsseusadjacents,elpesdelsqualsinicialment©sinfinit.Lasuposici³delpesequivaldr ae nim)iestornaamirarelsadjacentsiassignarpesos.Quantotselsnodeshaginestatvisitats,elpesdecada nimaques'haderec²rrerperanardelnodeinicialfinsaaquest.*

(Adjuntar esquema de procediment de Dijkstra)

Aquest algorisme, encara que no pot treballar amb pesos negatius Ã©s molt Ãºtil tÃ© una gran quantitat d'aplicacions prÃ ctiques:

- Navegadors GPS, on les arestes sÃ³*ncarrersicarreteres,elsnodescruïlesielspesosdist ncies.S'*
- Els routers utilitzen l'algorisme per portar-te a travÃ©s d'internet al servidor desitjat amb la menor cantitat de passos possibles.
- En robÃ²*ticas'utilitzaperferlaplanificaci³demovimentdelrobot.Cadanode©sunaunitatd'esp m©s²ptimperarribaralaposici³desitjada.Enepidemologiaespotutilitzarpermodelitzarungru*

Algorisme 3: Dijkstra

Data: Un graf ponderat G i un node inicial s

Result: DistÃ ncia mÃnima entre s i la resta de nodes del graf, arbre expensiu mÃnim

nou diccionari *dist*
nou diccionari Q
foreach *node v de G* **do**
  $Q[v] = \infty$
  $dist[v] = \infty$
end
$Q[s] = 0$
while Q *no estigui buit* **do**
  $u = minvalordeQ$
  $dist[u] = Q[u]$
  foreach *node v adjacent a u* **do**
    if *v existeix dins Q* **then**
      if $Q[v] > Q[u] + w(u, v)$ **then**
        /* $w(u, v)$ Ã©s el pes de l'aresta $\{u, v\}$ */
        $Q[v] = Q[u] + w(u, v)$
      end
    end
  end
  elimina($Q[u]$)
end
imprimeix(*dist*)

4.1.4 Bellman-Ford

Aquest algorisme tÃ© un funcionament i utilitats molt seblants a les de l'algorisme de Dijkstra, perÃ^2*t©laparticularitatdepodertractarsenseproblemeslesarestesambpesosnegatius, n*
m©scurt, per^{2}ambpesosnegatiusnoespotsuposarqueladesigualtattriangularescompleixi.Am©s, p
entredosnodescont©unciledepesnegatiu, noespottrobaruncammnimentreaquestsdosnodes.Aix^{2}es
, illavorselmnimpossibleseriade-∞.

Algorisme 4: Bellman-Ford

```
Data: Un graf ponderat G i un node inicia s
Result: DistÃ ncia mÃnima entre s i la resta de nodes del graf, arbre
        expansiu mÃnim
nou diccionari dist;
nou diccionari anterior;
foreach node v de G do
    dist[v] = ∞;
    anterior[v] = Nul;
end
dist[s] = 0;
for i in range(0, len(Adj)-1) do
    foreach u dins Adj do
        foreach v dins Adj[u] do
            if dist[v] > dist[u] + w(u, v) then
                /* w(u, v) Ã©s el pes de l'aresta {u, v}              */
                dist[v] = dist[u] + w(u, v);
            end
        end
    end
end
foreach u dins Adj do
    foreach v dins Adj[u] do
        if dist[v] > dist[u] + w(u, v) then
            imprimeix("Hi ha cicles de pesos negatius");
        end
    end
end
imprimeix(dist);
imprimeix(anterior);
```

4.1.5 Kruskal

L'algorisme de Kruskal serveix per trobar un arbre expansiu mÃnim. Aquest algorisme utilitza una estructura de dades especial, anomenada union-find. Aquesta estructura permet fer tres operacions diferents: crear conjunts (Make Set), determinar a quin conjunt estÃ un element (Find) i unir dos subconjunts en un de nou (Union). L'Ãºs d'aquesta estructura especialitzada fa que en grafs petits o poc densos l'algorisme sigui molt rÃ pid, perÃ²*engrafsm©sgransesrelantitzielproc©s, siguentm©seficie*
nim, iposaenelmateixconjuntelsnodesqueunia.Despr©sagafalasegentarestairepeteixelprocedime

Data: Un graf G i un node inicial s
Result: Arbre expansiu mÃnim de G
nova estructura union-find $subgraf$
nova llista $arbre$ ordena G per ordre creixent de pesos
foreach *node u de G* **do**
    foreach *node v adjacent a u* **do**
        if $subgraf[u]! = subgarf[v]$ **then**
            afegeix (u, v) a $arbre$
            $union(subgraf[u], subgraf[v])$
        end
    end
end
imprimeix($arbre$)

4.1.6 Prim

L'algorisme de Prim, tal com del de Kruskal, serveix per trobar l'arbre expansiu mÃnim d'un graf ponderat no dirigit. Aquest algorisme funciona amb diccionaris, estructures de dades mÃ©s normalitzades que el Union-Find. Com a conseqÃŒÃšncia, Prim Ã©s mÃ©s lent en grafs petits, perÃ^2*m©sr pidengrafsmoltdensos.Primdivi*

Algorisme 5: Prim

Data: Un graf G
Result: Arbre expansiu mÃnim de G
nou diccionari *anterior*
nou diccionari Q
foreach *node v de G* **do**
    $Q[v] = \infty$
end
$Q[0] = 0$
while Q *no estigui buit* **do**
    $u = min\{$valor de $Q\}$
    foreach *node v adjacent a u* **do**
        if *v existeix dins Q* **then**
            if $Q[v] > w(u, v)$ **then**
                $Q[v] = w(u, v)$
                $anterior[v] = 0$
            end
        end
    end
    elimina($Q[u]$)
end
imprimeix(*anterior*)

Tant l'algorisme de Kruskal com del de Prim tenen aplicacions semblants, perÃ^2*segonslamidadelgraf©sm©sconvenientutilitzar−neunoaltre.Entrelesaplicacionsd′aquest.*

- S'utilitzen per dissenyar xarxes de telÃšfon, aigua, gas, internet... En aquestes xarxes s'ha d'arribar a tots els punts on s'ha de fer la distribuciÃ^3, *iambl′arbreexpansiumnimespotassegurarquelaxarxa©selm©scurtapossible.Elsarbresexpansiusmnimsespodenutilitzarpergenerarlaberints.*

  S'utilitzen com a subrutines (o funcions) d'algorismes mÃ©s complexes.

4.1.7 Floyd-Warshall

L'algorisme de Floyd-Warshall Ã©s un algorisme que permet calcular les distÃ ncies entre tots els nodes d'un graf ponderat. Per fer-ho, compara els pesos de tots els camins possibles. A cada iteraciÃ^3, *esdefineixelconjuntdenodesquepottenircadacamisijas′hatrobatuncamambelsmateixosextremsescomparaelpestotald′ambd*^{3}*s.Elconjunt©sdelafor* i a cada iteraciÃ^3*esvaincrememntant*k(*al′inici*k=0)*.Elresultatd′aquestalgorisme©sunamatriuquc* $w(i, j)$.

Aquest algorisme es pot utilitzar per qualsevol de les aplicacions en que s'utilitzaria Dijkstra en mÃ©s d'un node. S'utilitza sobretot quan es vol mantenir una base de dades de pesos precalculats, per no haver d'executar Dijkstra en cada cas concret. A part d'aquesta aplicaciÃ^3, $s'utilitzatamb$©$d'altresmaneres$:

Per detectar cicles de pes negatiu, cosa que passarÃ quan $D_{i,i} < 0$, quan a la diagonal de la matriu hi hagi un valor negatiu.

Estudiar la clausura transitiva d'un graf, Ã©s a dir, veure quins nodes sÃ^3$naccessiblesdesdecadanode.Aix^2espotveurealamatriuresultant, onelsvalors\infty$ indiquen que no espot accedir a el node concret.

Algorisme 6: Floyd-Warshall

Data: un graf G
Result: una matriu quadrada amb les distÃ ncies entre tots els nodes
nova matriu *dist* de $|V|x|V|$
for i *in range(*$0, |V|$*)* **do**
  for j *in range(*$0, |V|$*)* **do**
    if $i = j$ **then**
      $dist[i][j] = 0$
    else
      $dist[i][j] = \infty$
    end
  end
end
foreach *node u de G* **do**
  foreach *node v adjacent a u* **do**
    $dist[u][v] = w(u, v)$
  end
end
for x *in range(*$0, |V|$*)* **do**
  for u *in range(*$0, |V|$*)* **do**
    for v *in range(*$0, |V|$*)* **do**
      if $dist[u][v] > dist[u][x] + dist[x][v]$ **then**
        $dist[u][v] = dist[u][x] + dist[x][v]$
      end
    end
  end
end

Part III
Disseny de grafs

Fins ara, s'han mostrat i estudiat grafs que ja estan definits i que s'ha de fer alguna sobre ells. Ara bÃ©, en aplicacions reals de teoria de grafs, sovint no hi ha un graf determinat, sinÃ³*ques'hadegenerar*.

5 Punt de Fermat i l'arbre de Steiner

Normalment, els nodes ja estaran determinats i el problema consistirÃ en trobar les arestes. Si aquest Ã©s aixÃ, segurament es podrÃ n afegir altres nodes. Un exemple d'aquest cas consistiria en haver d'unir tres ciutats amb carreteres de tal manera que des d'una es pugui arribar directament a les altres dues. La primera solÂ·luciÃ³*enquesolpensarunapersona*©*sferuntriangleenelquallesciutatssiguinelsvrtexs*(*l'equiv*
Aquesta solÂ·luciÃ³©*s*²*ptimasiesvolanard'unaciutatal'altreamblamnimadist nciapossible*, *per*²*sie*
El punt de Fermat, tambÃ© anomenat $X(13)$, Ã©s el punt del triangle tal que la suma de les distÃ ncies des de cada vÃšrtex fins a aquest punt Ã©s la mÃnima. El punt de Fermat s'aconsegueix amb el procediment segÃŒent:

- Donat un triangle T, es generen dos triangles equlÃ ters a partir de dos costats arbitraris de T.
- S'uneixen els nous vÃšrtexs dels triangles equilÃ ters amb els vÃšrtexs oposats a T.

- L'interseccÃ³*d'aquestsdossegmentsdonalaposici*³*delpuntdeFermat*. (adjuntar imatge de punt de fermat) Aquest procediment Ã©s vÃ lid per a triangles amb angles menors a 120º, perÃ²*encascontrari*, *elpuntdeFermatser l'anglequeelssuperi*.

El punt de Fermat es pot generalitzar per a altres polÃgons convexos a partir de triangulacions d'aquests. Un exemple senzill Ã©s el del quadrilÃ ter, on en fer les diagonals ja es generen quatre triangles. En aquest cas, trobant els punts en dos dels triangles i unint-los ja s'uneixen tots els punts amb la menor distÃ ncia possible. (adjuntar imatge del punt de fermat en quadrilÃ ters i superiors)

El problema de connectar amb la mÃnima distÃ ncia un nombre determinat de punts siguent possible aferigir nodes Ã©s conegut com el problema de l'arbre de Steiner. Per aquest problema s'ha demostrat que en l'arbre Ã²*ptimt*©*comam xim*n-2*nodesafegits*(*siguent*n*elnombreinicialdenodes*), *queaquestsstenensempregrau*3, *iqueformensemp*

6 Arbres expansius

Quan hi ha una situaciÃ^3*onnoespotafegircapnode, elmillorsolserutilitzarelsalgorismespertrobara*

Part IV

Topologia

La topologia Ã©s la branca de les matemÃ tiques que estudia les propietats de l'espai i si aquestes es conserven desprÃ©s de deformacions. Des d'un punt de vista teÃ^2*ric*,

7 Isomorfismes

Es diu que dos grafs sÃ^3*nisomorfssiexisteixunafunci*^{3}*bijectiva*φ entre els seus vÃšrtexs que conservi les arestes. Formalment dos grafs G i H sÃ^3*nisomorfssi*

$$\exists\varphi : V(G) \to V(H) \text{ tal que } \overline{\varphi(v)\varphi(u)} \in E(H) \text{ si i nomÃ©s si } \overline{vu} \in E(G)$$

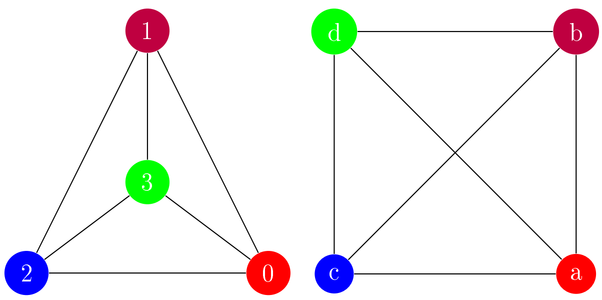


Figura 18: Grafs isomorfs