Recorreguts i connectivitat

PID_00254243

Joaquim Borges
Robert Clarisó
Ramon Masià
Jaume Pujol
Josep Rifà
Joan Vancells
Mercè Villanueva

Temps mínim de dedicació recomanat: 4 hores

Universitat Oberta de Catalunya

Índex

Introducció

Si haguéssim de descriure el present mòdul amb una paraula clau, aquesta seria la de *recorreguts* en un graf, és a dir, les maneres de "recórrer" un graf, visitant vèrtexs a través d'arestes. Així, doncs, en aquest mòdul analitzarem els diferents tipus de recorreguts que hi pot haver, l'accessibilitat entre vèrtexs i les distàncies. A més, presentarem les propietats estructurals que garanteixen que certs tipus de recorreguts sempre es puguin fer i que l'accessibilitat es conservi malgrat que algunes parts del graf (vèrtexs o arestes) desapareguin.

La teoria i els resultats que es presenten aquí constitueixen el marc teòric que és a la base d'importants aplicacions i modelitzacions per a xarxes de comunicacions i problemes d'optimització combinatòria.

1. Recorreguts

Un cop caracteritzats els grafs i presentats alguns dels més importants, convé estudiar les diverses maneres de recórrer-los. Per això s'introdueix el concepte de *recorregut*. És imprescindible enunciar alguns resultats fonamentals que hi fan referència.

Determinats grafs estan formats per "una sola peça", en el sentit que hi ha algun recorregut entre qualsevol parell de vèrtexs del graf. En canvi, n'hi ha d'altres que estan formats per "més d'una peça". Per les seves aplicacions en diversos camps, és important la noció de distància entre vèrtexs, que es pot definir mitjançant la connexió entre vèrtexs.

1.1. Concepte de recorregut

Definició 1

Un **recorregut** en un graf simple $G = (V,A)$ és una seqüència de vèrtexs $v_1,v_2,\ldots,v_k$ amb la propietat que dos vèrtexs consecutius en la seqüència són els extrems d'una aresta de G, és a dir, $\{v_i,v_{i+1}\} \in A$. Aquest recorregut d'extrems v_1, v_k s'anomena $v_1 - v_k$ **recorregut** o, també, recorregut entre v_1 i v_k.

Si $G = (V,A)$ és un graf dirigit, un **recorregut orientat** o **dirigit** és una seqüència de vèrtexs $w_1,w_2,\ldots,w_k$ amb la propietat que dos vèrtexs consecutius en la seqüència són els extrems d'un arc de G, és a dir, $(w_i,w_{i+1}) \in A$.

En la seqüència de vèrtexs no utilitzem la numeració dels vèrtexs, sinó la numeració corresponent al seu número d'ordre en la seqüència.

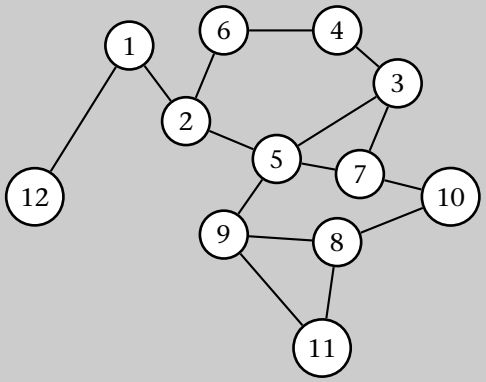
En un multigraf o un pseudograf és possible passar d'un vèrtex a un altre que hi sigui adjacent per més d'una aresta. Per això, cal indicar un recorregut com una seqüència d'arestes, $a_1,a_2,\ldots,a_{k-1},a_k$, amb la condició que dues arestes consecutives en la seqüència han de compartir un extrem.

Definició 2

- Un $u-v$ recorregut és **tancat** si els extrems coincideixen, és a dir, si $u = v$; en cas contrari es diu que és **obert**.
- La **longitud d'un recorregut** R, $\ell(R)$, és el nombre d'arestes que el componen. S'hi compten també les que hi pugui haver de repetides.
- Un **recorregut trivial** o de longitud 0 és el format per un únic vèrtex.

Exemple 1

Considerem el graf representat en la figura següent.



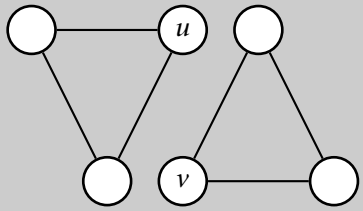
Un recorregut possible sobre aquest graf seria el corresponent a la seqüència de vèrtexs: 12,1,2,6,4,3,7,10,8,11; aquest seria un 12-11 recorregut de longitud 9.

Un recorregut tancat seria: 6,4,3,5,2,6, de longitud 5. Observeu que hi pot haver (depenent del graf) més d'un recorregut entre dos vèrtexs: per exemple, a continuació teniu dos 12-11 recorreguts: 12,1,2,5,7,10,8,9,11 i també 12,1,2,5,9,11.

Tot i que dos recorreguts tinguin globalment les mateixes arestes, poden ser diferents si les arestes s'utilitzen en ordres diferents. Per exemple, són recorreguts diferents els següents: 9,5,3,7,5,2 i 9,5,7,3,5,2.

Exemple 2

En el graf representat en la figura següent no existeix cap recorregut possible entre dos vèrtexs, en aquest cas u i v; els vèrtexs en qüestió no són mútuament accessibles en el graf $G = C_3 \cup C_3$.



Definició 3

Un recorregut és un **itinerari** si totes les arestes són diferents. Es poden destacar els tipus d'itinerari següents:

- Un **camí**, si no es repeteixen vèrtexs.
- Un **circuit**, si és tancat.
- Un **cicle** és, alhora, un camí (no repeteix vèrtexs) i un circuit (és tancat). Els grafs que no contenen cicles s'anomenen **acíclics**.

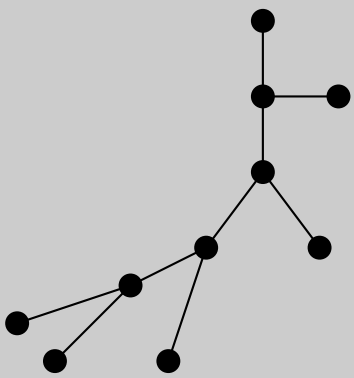
Exemple 3

En el graf de l'exemple 1 es té:

Cicle: 9,8,11,9
Cicle: 3,5,9,11,8,10,7,3
Circuit: 5,2,6,4,3,5,9,8,10,7,5
No itinerari: 12,1,2,5,3,7,5,2,1,12 (repeteix les arestes {1,2} i {2,5})
No camí: 1,2,5,7,3,5,9,8,11 (repeteix el vèrtex 5)

Exemple 4

Vegeu a continuació un exemple de graf acíclic.



Exemple 5

Un arbre, T, és un graf que compleix la propietat que entre dos vèrtexs qualssevol del graf, hi ha un únic camí que els connecta. Per exemple, el graf de l'exemple 4 és un arbre. És evident que un arbre no pot contenir cicles.

Exemple d'arbre

L'arbre genealògic d'una família és un exemple d'arbre, sempre que dos descendents de l'avantpassat comú no tinguin mai descendència comuna.

Vegeu també

Els arbres són grafs molt importants i, per això, s'estudiaran més endavant en el mòdul "Arbres".

Exercicis

1. Quina és la longitud mínima d'un cicle?

2. Un cicle de longitud n d'un graf d'ordre n és un subgraf d'aquest graf?

3. Indiqueu diversos cicles en el graf complet K_6.

4. Quina és la longitud del graf cicle C_n?

5. Indiqueu si són acíclics els grafs T_n, C_n, K_n, $K_{n,m}$, R_n, E_n.

6. La propietat d'aciclicitat s'hereta per part dels subgrafs d'un graf. Cert o fals?

7. L'eliminació d'arestes o de vèrtexs d'un graf acíclic produeix un nou graf acíclic?

8. Si en un recorregut no hi ha repetició d'arestes, hi pot haver repetició de vèrtexs?

Solucions

1. La longitud mínima és 3.

2. Cert (acabeu de justificar la resposta).

3. Té molts cicles: només cal prendre tres vèrtexs qualssevol i unir-los dos a dos amb arestes diferents.

4. Per al graf cicle C_n, $\ell(C_n) = n$.

5. Els acíclics són T_n, E_n, K_n ($n \leq 2$) i $K_{n,m}$ ($n = 1$ o $m = 1$).

6. Cert (acabeu de justificar la resposta).

7. Sí (acabeu de justificar la resposta).

8. Sí (acabeu de justificar la resposta).

1.2. Resultats fonamentals

Proposició 1

Donats dos vèrtexs diferents u, v d'un graf $G = (V,A)$, tot $u - v$ recorregut conté un $u-v$ camí, és a dir, un recorregut entre els vèrtexs sense repetició de vèrtexs.

Demostració: Sigui R un $u - v$ recorregut del graf G. Com que u i v són diferents podem suposar que R és obert, i sigui, per exemple, $R : u = w_0 w_1, \ldots, w_k = v$, on hi poden haver vèrtexs repetits. Si no n'hi ha cap de repetit, hem acabat, ja que R compleix la condició requerida. Suposem, per tant, que hi ha algun vèrtex repetit, i siguin $i < j$ tals que $w_i = w_j$; aleshores suprimim els vèrtexs $w_i, w_{i+1}, \ldots, w_{j-1}$ i en resulta un nou $u - v$ recorregut R_1 de longitud estrictament inferior amb una repetició de $w_i = w_j$ eliminada (tot i que encara poden quedar més repeticions del mateix vèrtex). Si R_1 és un camí, hem acabat; en cas contrari, el procés continua i necessàriament s'arriba a la conclusió indicada, ja que la longitud inicial és finita. ∎

Exemple 6

En el graf de l'exemple 1, a partir del recorregut 2,5,3,7,5,9,8,11 se'n pot extreure un camí: 2,5,9,8,11. Per a fer-ho només cal descartar la part del recorregut que hi ha entre les dues visites al vèrtex 5.

Proposició 2

Si en un graf $G = (V,A)$ hi ha dos camins diferents que connecten un parell de vèrtexs, aleshores el graf conté algun cicle.

Demostració: Considerem dos $u-v$ camins diferents: $R_1 : u = w_1 w_2 \ldots w_p = v$ i $R_2 : u = t_1 t_2 \ldots t_q = v$. Informalment, els camins poden anar compartint vèrtexs fins a arribar al primer vèrtex a partir del qual se separen, és a dir, comencen a ser diferents. Sigui, per tant, i el primer índex per al qual $w_i \neq t_i$. Tard o d'hora els camins han de confluir, finalment, al vèrtex terminal comú, que és v, i això ho faran en el mateix vèrtex o abans; però, en tot cas, a partir d'un cert lloc en endavant, els dos camins tornen a compartir vèrtexs (i arestes, ja que el graf és simple). Ara bé, en un cert vèrtex posterior a la bifurcació anterior han de tornar a coincidir per primera vegada. Siguin, doncs, j, k els mínims tals que $i < j \leq p$, $i < k \leq q$, $w_j = t_k$. Aleshores el camí $w_{i-1} w_i w_{i+1} \ldots w_j t_{k-1} t_{k-2} \ldots t_i w_{i-1}$ és un cicle. ∎

Teorema 3

Si tots els vèrtexs d'un graf $G = (V,A)$ són de grau superior o igual a 2, aleshores el graf conté algun cicle.

Demostració: Sigui n l'ordre del graf. Considerem un vèrtex qualsevol, que indiquem per w_1, de grau com a mínim 2 per hipòtesi; hi ha un vèrtex w_2 adjacent a w_1. Atès que $g(w_2) \geq 2$, hi ha un vèrtex w_3 adjacent a w_2 i diferent de w_1. D'aquesta manera podem construir una seqüència de vèrtexs encadenats. Sigui ara w_i diferent de tots el anteriors i adjacent a w_{i-1}. Si w_i és adjacent a algun dels anteriors (que no sigui w_{i-1}), tenim un cicle i hem acabat el problema. Si no és així, atès que és de grau superior o igual a 2, hi ha un nou vèrtex w_{i+1} diferent de tots els anteriors i adjacent a w_i. Per aquest procés, necessàriament finit, o bé es produeix en algun moment un cicle i hem acabat, o bé es van incorporant tots els vèrtexs en un camí que els conté tots, $w_1, w_2, \ldots w_n$, i es mantenen les propietats anteriors. Ara bé, atès que $g(w_1) \geq 2, \ldots, g(w_n) \geq 2$, han de ser necessàriament adjacents i, per tant, es forma cicle. ∎

Exercicis

9. Un graf acíclic sense vèrtexs aïllats conté vèrtexs de grau 1. Cert o fals?

10. Si un graf és acíclic aleshores entre cada parell de vèrtexs diferents hi ha, com a màxim un camí. Cert o fals?

11. Demostreu que tot graf r-regular ($r \geq 2$) conté un cicle.

12. Doneu un exemple de graf acíclic tal que el seu complementari també sigui acíclic.

13. Demostreu que si en un graf d'ordre n el grau màxim dels vèrtexs és menor que $n-2$ aleshores el graf complementari conté un cicle.

Solucions

9. Cert. Si no tingués cap vèrtex de grau 1 aleshores tots els vèrtexs tindrien grau més gran o igual a 2 i hauria de contenir un cicle, pel teorema 3.

10. Cert. Si hi hagués més d'un camí, aleshores contindria algun cicle, per la proposició 2.

11. Si el graf és r-regular ($r \geq 2$) aleshores tots els vèrtexs són de grau superior o igual a 2 i el graf contindrà un cicle, pel teorema 3.

12. Per exemple, el graf T_3.

13. Si G^c és el graf complementari, aleshores $g_{G^c}(v) = n - 1 - g_G(v) \geq 2$ per a tot $v \in V$ i, per tant, conté un cicle, pel teorema 3.

2. Algorismes d'exploració de grafs

Per a determinar propietats d'un graf, sovint cal explorar cadascun dels vèrtexs i de les arestes del graf en un ordre determinat. Des d'un punt de vista algorísmic hi ha, bàsicament, dos procediments per a fer-ho: l'**algorisme de cerca primàriament en profunditat** (**DFS** o *Depth First Search*) i l'**algorisme de cerca primàriament en amplada** (**BFS** o *Breadth First Search*).

L'algorisme de cerca primàriament en profunditat s'aplica a diversos problemes de connectivitat (cercar components connexos o biconnexos) i també pot comprovar si un graf és acíclic. Altres algorismes com el de Dijkstra i el de Prim utilitzen idees similars al DFS.

L'algorisme de cerca primàriament en amplada s'aplica a diversos problemes de coloració (coloració de grafs bipartits), en la cerca del cicle més curt d'un graf o en el càlcul de distàncies.

2.1. Algorisme DFS

En l'algorisme DFS s'inicia l'exploració del graf per un vèrtex prefixat i a partir d'aquest s'exploren tots per mitjà de les arestes, sense repetició, d'una manera determinada. Essencialment, es tracta d'avançar el més profundament possible, sense explorar totes les possibilitats que es donen a partir del vèrtex actual. D'aquesta manera, del vèrtex actual es passa a un vèrtex adjacent no visitat, d'aquest a un tercer encara no visitat, i així successivament. Si s'arriba a un vèrtex des del qual no es pot avançar, cal tornar enrere (retrocés o *backtracking*) i eliminar-lo. Aquest procés es repeteix fins a tornar al punt inicial, on s'haurà acabat l'exploració.

Per a resoldre problemes és útil conèixer la formulació de l'algorisme bàsic que explora tots els vèrtexs sense repetició. És clar que es pot completar amb accions addicionals i formular variants noves.

2.1.1. Formulació de l'algorisme DFS

Estructures necessàries per a la formulació de l'algorisme:

- Un graf $G = (V,A)$ representat mitjançant una llista d'adjacències.

- Un conjunt P dels vèrtexs que s'han visitat, en l'ordre en què s'ha fet, que permeti de fer marxa enrere. L'estructura de dades adequada és la d'una pila amb les operacions habituals: *empilar*(P,v), *desempilar*(P), *cim*(P).
- Una taula de vèrtexs (*estat*) que registra els vèrtexs que es van visitant.
- Una llista R que conté els vèrtexs visitats fins ara (i que finalment coincidirà amb el conjunt de tots els vèrtexs).

Estructura de pila

En l'estructura de pila els elements es desempilen en ordre invers al qual s'empilen, i el cim és el darrer element empilat.

Quan és possible visitar més d'un vèrtex, escollim sempre el d'índex mínim en l'ordenació dels vèrtexs disponibles, segons l'ordenació general dels vèrtexs del graf (per exemple, si els vèrtexs es representen per lletres, aleshores es fa la tria del primer disponible per ordre alfabètic).

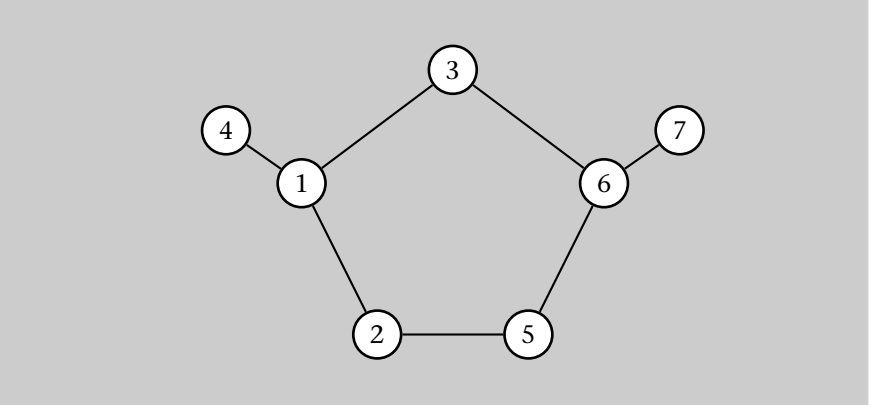
La pila P s'inicialitza amb el vèrtex de partida i es van empilant els vèrtexs que són visitats. Quan s'arriba a un vèrtex des del qual no es pot continuar, es van desempilant vèrtexs fins a trobar-ne un a partir del qual ja es pot tornar a avançar, i es reinicia, així, el procés. Si la pila queda buida s'atura el procés.

```
Entrada  : G = (V,A), v ∈ V
Sortida  : R, vèrtexs visitats
algorisme DFS(G,v)
  inici
    P ← ∅
    R ← [v]
    per w ∈ V
        estat[w] ← 0
    fiper
    estat[v] ← 1
    empilar(P,v)
    mentre P ≠ ∅
              w ← cim(P)
              si w és adjacent a u amb estat[u] = 0
                 aleshores empilar(P,u)
                           estat[u] ← 1
                           afegir(R,u)
                      si_no desempilar(P)
              fisi
    fimentre
    retorn (R)
  fi
```

2.1.2. Simulació de l'algorisme

Exemple 7

Considerem el graf representat en la figura següent.



Aquesta taula enregistra el funcionament de l'algorisme d'exploració en profunditat (DFS) per a aquest graf, amb vèrtex d'inici $v = 1$.

P	Vèrtex afegit	Vèrtex eliminat	R
1	1	-	[1]
12	2	-	[1,2]
125	5	-	[1,2,5]
1256	6	-	[1,2,5,6]
12563	3	-	[1,2,5,6,3]
1256	-	3	[1,2,5,6,3]
12567	7	-	[1,2,5,6,3,7]
1256	-	7	[1,2,5,6,3,7]
125	-	6	[1,2,5,6,3,7]
12	-	5	[1,2,5,6,3,7]
1	-	2	[1,2,5,6,3,7]
14	4	-	[1,2,5,6,3,7,4]
1	-	4	[1,2,5,6,3,7,4]
$\emptyset$	-	1	[1,2,5,6,3,7,4]

2.1.3. Anàlisi de l'algorisme DFS

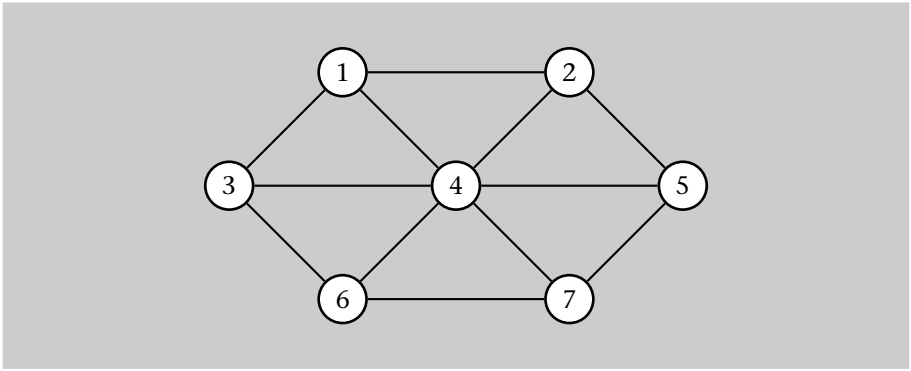
Si el graf G té ordre n i mida m, podem calcular, de manera aproximada, la complexitat d'aquest algorisme observant que cada vèrtex és visitat dues vegades (una quan s'afegeix a la pila i una altra quan s'elimina). A més, quan accedim a un vèrtex de la pila analitzem tots els que hi són adjacents per a cercar els vèrtexs encara no visitats. En total, ho haurem de fer tantes vegades com arestes hi ha a la llista d'adjacències del graf.

Així, el nombre total d'operacions serà proporcional a $2n + 2m$ i l'algorisme tindrà una complexitat $O(n + m)$.

Exercicis

14. Trobeu la llista dels vèrtexs visitats per l'algorisme DFS en el graf de l'exemple 7, però començant pel vèrtex 6. Escriviu, també, la taula que registra el funcionament de l'algorisme.

15. Trobeu la llista dels vèrtexs visitats per l'algorisme DFS en el graf següent començant pel vèrtex 1. Escriviu, també, la taula que registra el funcionament de l'algorisme.



16. L'algorisme DFS, al mateix temps que visita tots els vèrtexs d'un graf, també visita les arestes incidents en aquests vèrtexs. Modifiqueu l'algorisme perquè retorni la llista d'arestes visitades pel DFS. En lloc de la llista R serà una llista S d'arestes visitades.

17. Utilitzeu l'algorisme de l'exercici anterior per a obtenir les arestes visitades en el graf de l'exemple 7. Descriviu també la taula de funcionament de l'algorisme amb la capçalera següent:

P	Aresta afegida	S

Solucions

14. La llista de vèrtexs visitats és $R = [6,3,1,2,5,4,7]$.

15. La llista de vèrtexs visitats és $R = [1,2,4,3,6,7,5]$.

16. La versió del DFS per a les arestes serà:

```
Entrada : G = (V,A), v ∈ V
Sortida : S, arestes visitades
algorisme DFS(G,v)
  inici
    P ← ∅
    S ← ∅
    per w ∈ V
        estat[w] ← 0
    fiper
    estat[v] ← 1
    empilar(P,v)
    mentre P ≠ ∅
              w ← cim(P)
              si w és adjacent a u amb estat[u] = 0
                 aleshores empilar(P,u)
                           estat[u] ← 1
                           afegir(S,{w,u})
                      si_no desempilar(P)
              fisi
    fimentre
    retorn (S)
  fi
```

17. La taula de funcionament de l'algorisme serà:

P	Aresta afegida	S
1	-	$\emptyset$
12	{1,2}	[{1,2}]
125	{2,5}	[{1,2},{2,5}]
1256	{5,6}	[{1,2},{2,5},{5,6}]
12563	{6,3}	[{1,2},{2,5},{5,6},{6,3}]
1256	-	[{1,2},{2,5},{5,6},{6,3}]
12567	{6,7}	[{1,2},{2,5},{5,6},{6,3},{6,7}]
1256	-	[{1,2},{2,5},{5,6},{6,3},{6,7}]
125	-	[{1,2},{2,5},{5,6},{6,3},{6,7}]
12	-	[{1,2},{2,5},{5,6},{6,3},{6,7}]
1	-	[{1,2},{2,5},{5,6},{6,3},{6,7}]
14	{1,4}	[{1,2},{2,5},{5,6},{6,3},{6,7},{1,4}]
1	-	[{1,2},{2,5},{5,6},{6,3},{6,7},{1,4}]
$\emptyset$	-	[{1,2},{2,5},{5,6},{6,3},{6,7},{1,4}]

Observació

El graf (V,S) sempre és un arbre.

2.2. Algorisme BFS

L'algorisme BFS és similar al DFS, amb la diferència important que quan arribem al vèrtex actual que visitem, primer es fa una visita sistemàtica a tots els seus vèrtexs adjacents que encara no hagin estat visitats, escollits per ordre segons l'ordenació que hi hagi.

És útil conèixer la formulació de l'algorisme bàsic, que es pot completar amb accions addicionals per tal de formular-ne noves variants.

2.2.1. Formulació de l'algorisme BFS

Estructures necessàries per a la formulació de l'algorisme:

- Un graf $G = (V,A)$ representat mitjançant una llista d'adjacències.
- Un conjunt Q dels vèrtexs que s'han visitat, en l'ordre en què s'ha fet. L'estructura de dades adequada és la d'una cua amb les operacions habituals: *afegir*(Q,v), *eliminar*(Q), *primer*(Q).
- Una taula de vèrtexs (*estat*) que registra els vèrtexs que es van visitant.
- Una llista R que conté els vèrtexs visitats fins ara (i que finalment coincidirà amb el conjunt de tots els vèrtexs).

Estructura de cua

En l'estructura de cua els elements s'eliminen en el mateix ordre en què s'afegeixen.

Quan és possible visitar més d'un vèrtex, sempre triem el d'índex mínim en l'ordenació dels vèrtexs disponibles, segons l'ordenació general dels vèrtexs del graf (si els vèrtexs es representen per lletres, aleshores es fa la tria del primer disponible per ordre alfabètic).

La cua Q s'inicia amb el vèrtex de partida i els vèrtexs que són visitats es van afegint a la cua. Quan s'han visitat tots els vèrtexs adja-

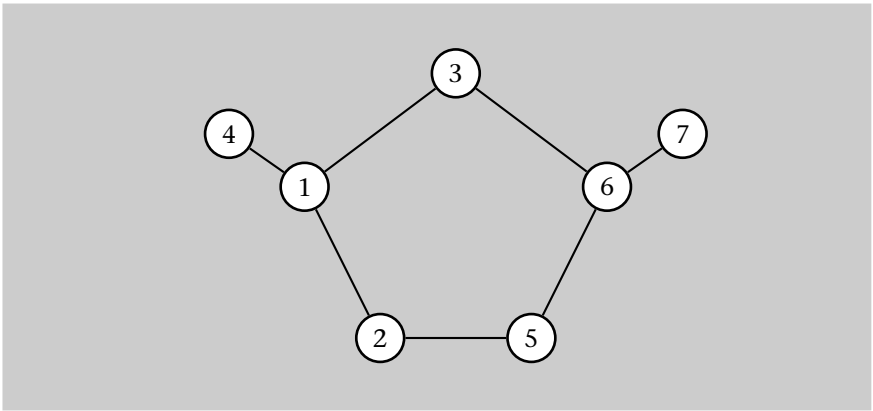
cents al de partida, s'elimina de la cua i es continua amb el vèrtex següent de la cua. Quan aquesta queda buida, s'atura el procés.

```
Entrada : G = (V,A),v ∈ V
Sortida : R, vèrtexs visitats
algorisme BFS(G,v)
  inici
    Q ← ∅
    R ← [v]
    per w ∈ V
        estat[w] ← 0
    fiper
    estat[v] ← 1
    afegir(Q,v)
    mentre Q ≠ ∅
              w ← primer(Q)
              per u adjacent a w
                  si estat[u] = 0
                     aleshores afegir(Q,u)
                               estat[u] ← 1
                               afegir(R,u)
                  fisi
              fiper
              eliminar(Q)
    fimentre
    retorn (R)
  fi
```

2.2.2. Simulació de l'algorisme

Exemple 8

Considerem el graf representat en la figura següent.



Aquesta taula registra el funcionament de l'algorisme d'exploració en amplada (BFS) per a aquest graf, amb vèrtex d'inici $v = 1$.

Q	Vèrtex afegit	Vèrtex eliminat	R
1	1	-	[1]
12	2	-	[1,2]
123	3	-	[1,2,3]
1234	4	-	[1,2,3,4]
234	-	1	[1,2,3,4]
2345	5	-	[1,2,3,4,5]
345	-	2	[1,2,3,4,5]
3456	6	-	[1,2,3,4,5,6]
456	-	3	[1,2,3,4,5,6]
56	-	4	[1,2,3,4,5,6]
6	-	5	[1,2,3,4,5,6]
67	7	-	[1,2,3,4,5,6,7]
7	-	6	[1,2,3,4,5,6,7]
$\emptyset$	-	7	[1,2,3,4,5,6,7]

2.2.3. Anàlisi de l'algorisme BFS

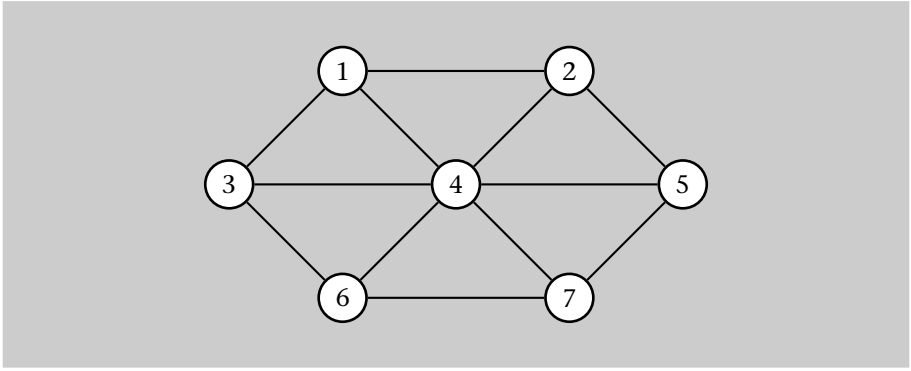
Observeu que si el graf G té ordre n aleshores cada vèrtex s'afegeix a la cua una vegada. A més, quan accedim a un vèrtex de la cua, n'analitzem la llista d'adjacències per a cercar els vèrtexs encara no visitats. En total, haurem analitzat $2m$ arestes, on m és la mida del graf.

Així el nombre total d'operacions serà proporcional a $n + 2m$ i l'algorisme tindrà una complexitat $O(n + m)$.

Exercicis

18. Trobeu la llista dels vèrtexs visitats per l'algorisme BFS en el graf de l'exemple 8 però començant pel vèrtex 6. Escriviu, també, la taula que registra el funcionament de l'algorisme.

19. Trobeu la llista dels vèrtexs visitats per l'algorisme BFS en el graf següent començant pel vèrtex 1. Escriviu, també, la taula que registra el funcionament de l'algorisme.



20. Com en el cas del DFS, l'algorisme BFS, alhora que visita tots els vèrtexs d'un graf, també visita les arestes incidents en aquests vèrtexs. Modifiqueu

l'algorisme perquè retorni la llista d'arestes visitades pel BFS. En lloc de la llista R serà una llista S d'arestes visitades.

21. Useu l'algorisme de l'exercici anterior per a obtenir les arestes visitades en el graf de l'exemple 8. Descriviu també la taula de funcionament de l'algorisme amb la capçalera següent:

Q	Aresta afegida	S

Solucions

18. La llista de vèrtexs visitats és R = [6,3,5,7,1,2,4].

19. La llista de vèrtexs visitats és R = [1,2,3,4,5,6,7].

20. La versió del BFS per a les arestes serà:

```
Entrada : G = (V,A), v ∈ V
Sortida : S, arestes visitades
algorisme BFS(G,v)
  inici
    Q ← ∅
    S ← ∅
    per w ∈ V
        estat[w] ← 0
    fiper
    estat[v] ← 1
    afegir(Q,v)
    mentre Q ≠ ∅
              w ← primer(Q)
              per u adjacent a w
                  si estat[u] = 0
                     aleshores afegir(Q,u)
                               estat[u] ← 1
                               afegir(S,{w,u})
                  fisi
              fiper
              eliminar(Q)
    fimentre
    retorn (S)
  fi
```

21. La taula de funcionament de l'algorisme serà:

Q	Aresta afegida	S
1	-	$\emptyset$
12	{1,2}	[{1,2}]
123	{1,3}	[{1,2},{1,3}]
1234	{1,4}	[{1,2},{1,3},{1,4}]
234	-	[{1,2},{1,3},{1,4}]
2345	{2,5}	[{1,2},{1,3},{1,4},{2,5}]
345	-	[{1,2},{1,3},{1,4},{2,5}]
3456	{3,6}	[{1,2},{1,3},{1,4},{2,5},{3,6}]
456	-	[{1,2},{1,3},{1,4},{2,5},{3,6}]
56	-	[{1,2},{1,3},{1,4},{2,5},{3,6}]
6	-	[{1,2},{1,3},{1,4},{2,5},{3,6}]
67	{6,7}	[{1,2},{1,3},{1,4},{2,5},{3,6},{6,7}]
7	-	[{1,2},{1,3},{1,4},{2,5},{3,6},{6,7}]
$\emptyset$	-	[{1,2},{1,3},{1,4},{2,5},{3,6},{6,7}]

Observació

El graf (V,S) resultant és un arbre.

3. Connectivitat

Un dels conceptes fonamentals de la teoria de grafs és el de *connectivitat*, que es defineix a partir de la possibilitat d'establir un recorregut entre dos vèrtexs qualssevol d'un graf. El concepte de connectivitat és especialment important en aquells problemes en els quals cal mesurar distàncies en una xarxa de distribució o en problemes de vulnerabilitat d'una xarxa d'interconnexió.

Vegeu també

En el mòdul "Grafs eulerians i grafs hamiltonians", també veurem que la connectivitat és útil per a esbrinar si un graf és hamiltonià o eulerià, encara que en aquest darrer cas és més útil conèixer el grau de cada vèrtex.

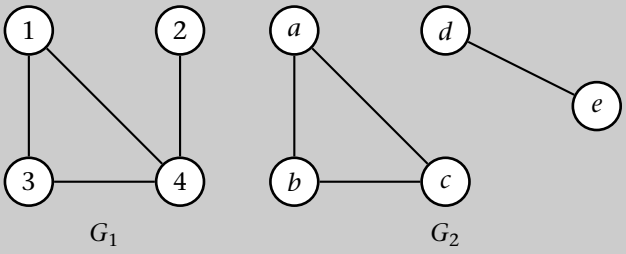
3.1. Connexió entre vèrtexs

Definició 4

Un graf $G = (V,A)$ és **connex** si per a cada parell de vèrtexs u i v de G existeix un $u - v$ camí.

Exemple 9

La figura següent mostra dos grafs G_1 i G_2. El primer és connex, ja que entre cada parella de vèrtexs hi ha un camí que els uneix. El segon no és connex ja que, per exemple, entre els vèrtexs a i e no hi ha cap recorregut que els uneixi.



Quan un graf G no sigui connex, ens preocuparem de determinar els subgrafs de G que siguin connexos i que siguin maximals respecte a aquesta propietat.

Definició 5

En el conjunt de vèrtexs V d'un graf $G = (V,A)$ definim la relació següent:

$$\forall u,v \in V, \quad u \equiv v \Leftrightarrow \text{existeix un } u-v \text{ camí de } G.$$

Relació d'equivalència

Recordeu que una relació $x\ R\ y$ és d'equivalència si, i només si, per a qualsevol x, y, z es compleix:
1) R és reflexiva ($x\ R\ x$);
2) R és simètrica ($x\ R\ y \Rightarrow y\ R\ x$), i
3) R és transitiva ($x\ R\ y$ i $y\ R\ z \Rightarrow x\ R\ z$). La igualtat ($x = y$) és un exemple de relació d'equivalència.

Proposició 4

Aquesta relació és una relació d'equivalència en el conjunt de vèrtexs del graf.

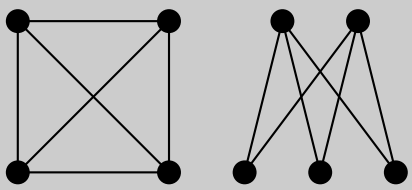
En conseqüència, s'estableix una partició de $V = V_1 \cup \ldots \cup V_k$: els vèrtexs d'una mateixa classe són mútuament accessibles per algun camí; vèrtexs de classes diferents són innaccessibles. Evidentment, també es pot establir, de manera semblant, una partició del conjunt d'arestes $A = A_1 \cup \ldots \cup A_k$.

Definició 6

Els **components connexos** del graf G són els subgrafs $G_i = \langle V_i \rangle$ generats per cadascuna d'aquestes classes d'equivalència V_i; així es pot expressar el graf com a unió de components connexos: $G = G_1 \cup \ldots \cup G_k$.

Exemple 10

El graf G no és connex. Té dos components connexos (el graf complet d'ordre 4, i el graf bipartit complet d'ordre 3 + 2). Així, $G = K_4 \cup K_{3,2}$.



Per la pròpia definició, tot component connex d'un graf G és ell mateix connex com a graf. Així, un graf és connex si, i només si, té un únic component connex.

Comparant l'ordre i la mida d'un graf amb els ordres i les mides dels seus components connexos s'obté la relació següent.

Proposició 5

Si un graf és d'ordre n i mida m, amb k components connexos, d'ordres i mides respectives n_i, m_i ($i = 1, \ldots, k$), aleshores:

$$n = \sum_{i=1}^{k} n_i, \quad m = \sum_{i=1}^{k} m_i .$$

El resultat següent estableix la mida mínima d'un graf connex.

Proposició 6

Si G és un graf connex d'ordre n i mida m, aleshores $m \geq n-1$.

Demostració: Ho demostrarem per inducció respecte a l'ordre n del graf. Si $n = 1$ el resultat és immediat.

Suposem que la propietat és certa per a tot graf connex d'ordre $n - 1 \geq 1$ i demostrem que també ho és per a tot graf connex d'ordre n. Sigui $G = (V,A)$ un graf connex d'ordre n i mida m. Hem de demostrar que $m \geq n - 1$.

Si tots els vèrtexs de G tenen grau major o igual que 2, aleshores apliquem el lema de les encaixades,

$$2m = \sum_{v \in V} g(v) \geq 2n,$$

i, per tant, $m \geq n > n - 1$.

En cas que G tingui un vèrtex v de grau 1, aleshores el graf $G' = G - v$ és un graf connex d'ordre $n - 1$ i mida $m - 1$. Aplicant la hipòtesi d'inducció al graf G' tindrem que $m-1 \geq n-2$, d'on resulta que $m \geq n-1$, tal com volíem demostrar. ∎

Com a conseqüència directa d'aquest resultat s'obté el corol·lari següent.

Proposició 7

Si G és un graf que té ordre n, mida m i k components connexos, aleshores $m \geq n - k$.

Exemple 11

Tots els grafs amb seqüència de graus 4,4,3,3,3,3,3,3 són connexos.

Suposem que existeixi algun graf no connex amb aquesta seqüència de graus i sigui $G = G_1 \cup \ldots \cup G_k$, $k \geq 2$, la descomposició del graf com a reunió de components connexos. Sigui $n = 8$ l'ordre de G i sigui n_i l'ordre del component connex G_i, considerat com a graf. Òbviament es té $n_1 + \ldots + n_k = n = 8$. Ara bé, els vèrtexs són, com a mínim, de grau 3 a cada component connex i, en conseqüència, cada component connex és com a mínim d'ordre 4, és a dir, $n_1 + \ldots + n_k \geq 4k$. A partir de la igualtat anterior es té que $k = 2$, $n_1 = n_2 = 4$. Ara bé, essent els components connexos d'ordre 4, cap d'ells no pot contenir vèrtexs d'ordre 4. Per tant, no existeix cap graf no connex amb aquestes característiques.

Exemple 12

Quan s'elimina un vèrtex d'un graf connex, el nombre de components connexos que es produeixen no pot superar el grau del vèrtex eliminat.

L'afirmació és molt intuïtiva; expressem-la formalment i demostrem-la. Sigui $G = (V,A)$ un graf connex d'ordre $n \geq 3$ i sigui $u \in V$ de grau $g(u)$; aleshores el nombre de components connexos de $G - u$ és menor o igual que $g(u)$.

En efecte, sigui $G' = G - u = G_1 \cup \ldots \cup G_k$, expressió del graf com a reunió de components connexos. Vegem que cada G_i conté com a mínim un vèrtex u_i adjacent a u en G, cosa que demostraria l'afirmació. Si no fos així, per a algun G_{i_0}, aleshores, per la connexió del graf G, algun vèrtex $w \in V(G_{i_0})$ serà adjacent (en G) a algun altre vèrtex $q \in V((G_j),(q \neq u,(j \neq i_0)))$, ja que en cas contrari seria G no connex. Ara bé, essent així, amb l'eliminació de u no es crearia el component connex G_{i_0}.

Exercicis

22. Demostreu que un graf amb la seqüència de graus 3,1,1,1,1,1 no pot ser connex.

23. Trobeu el nombre de components connexos d'un graf que té com a llista d'adjacències,

a	:	f, i, j
b	:	c, g
c	:	b, e, g
d	:	h
e	:	c, g, j, f
f	:	a, i, e
g	:	b, c, e
h	:	d
i	:	a, f
j	:	a, e

24. Si eliminem una aresta d'un cicle d'un graf, es pot incrementar el nombre de components connexos?

25. Quins dels grafs següents són connexos?

$$T_n,\ C_n,\ \ N_n,\ N_n + N_m, T_2 \times C_n.$$

En cas que no siguin connexos indiqueu quants components connexos tenen.

Solucions

22. En primer lloc es pot verificar, si utilitzem l'algorisme de Havel-Hakimi, que la seqüència 3,1,1,1,1,1 és gràfica.

Com que el graf té ordre 6, si el graf fos connex la mida hauria de ser més gran o igual a 5. Però, pel lema de les encaixades, $2m = 3+1+1+1+1+1 = 8$ i $m = 4$.

23. El graf té dos components connexos.

24. No; en particular, en un graf connex mai no es produeix desconnexió amb l'eliminació d'una aresta que pertany a un cicle (ja que sempre resta un recorregut alternatiu per a un parell de vèrtexs que es comuniquin per un recorregut que utilitzi arestes del cicle).

25. Tots són connexos menys N_n ($n > 1$), que conté n components connexos, un per a cada vèrtex.

3.2. Test de connexió

Podem utilitzar l'algorisme DFS introduït en el subapartat 2.1. per a comprovar si un graf $G = (V,A)$ és connex. Recordem que l'algorisme DFS retorna la llista R de tots els vèrtexs accessibles a partir d'un vèrtex v fixat. Si la llista conté tots els vèrtexs del graf, el graf serà connex. En cas contrari, serà disconnex.

```
Entrada : G = (V,A), v ∈ V
Sortida : CERT si G és connex. FALS en cas contrari
algorisme TestConnexió(G,v)
  inici
    connex ← CERT
    R ← DFS(G,v)
    si |R| ≠ |V|
      aleshores connex ← FALS
    fisi
    retorn (connex)
  fi
```

Com que el DFS té una complexitat $O(n+m)$ podem afirmar que comprovar si un graf és connex té una complexitat $O(n+m)$.

Exemple 13

Observem l'aplicació de l'algorisme sobre el graf G_2 de l'exemple 9 començant pel vèrtex a:

P	Vèrtex afegit	Vèrtex eliminat	R
a	a	-	[a]
ab	b	-	[a,b]
abc	c	-	[a,b,c]
ab	-	c	[a,b,c]
a	-	b	[a,b,c]
$\emptyset$	-	a	[a,b,c]

Com que el graf G_2 té ordre 5 i R només conté 3 vèrtexs, podem concloure que G_2 no és connex.

Aquest exemple posa en evidència el resultat següent.

Proposició 8

Sigui $G = (V,A)$ un graf i v un vèrtex de G. El subgraf induït pels vèrtexs visitats de G emprant l'algorisme DFS és el component connex que conté v.

Exercicis

26. Es podria utilitzar el BFS en lloc del DFS en el test de connexió?

27. Utilitzeu el test de connexió i comproveu si el graf G definit per la matriu d'adjacències següent és connex.

$$B = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

28. Modifiqueu el test de connexió perquè ens retorni el nombre de components connexos d'un graf G.

Solucions

26. Sí, ja que els dos algorismes retornen tots els vèrtexs accessibles a partir d'un vèrtex fixat.

27. Suposant que els vèrtexs del graf són el conjunt $V = \{a,b,c,d,e\}$, apliquem el test de connexió:

P	Vèrtex afegit	Vèrtex eliminat	R
a	a	-	[a]
ab	b	-	[a,b]
abc	c	-	[a,b,c]
abcd	d	-	[a,b,c,d]
abc	-	d	[a,b,c,d]
abce	e	-	[a,b,c,d,e]
abc	-	e	[a,b,c,d,e]
ab	-	c	[a,b,c,d,e]
a	-	b	[a,b,c,d,e]
$\emptyset$	-	a	[a,b,c,d,e]

Del test es dedueix que el graf és connex.

28. En aquest cas haurem de repetir el test de connexió per a cada component connex del graf:

Entrada : $G = (V,A)$
Sortida : nombre de components connexos del graf G.
algorisme *ComponentsConnexos*(G)
  inici
    $ncomponents \leftarrow 0$
    $U \leftarrow V$
    mentre $U \neq \emptyset$
      $v \leftarrow$ primer vèrtex d'U
      $R \leftarrow DFS(G,v)$
      $ncomponents \leftarrow ncomponents + 1$
      $U \leftarrow U - R$
    fimentre
    retorn ($ncomponents$)
  fi

Observeu que la complexitat continua essent $O(n + m)$, ja que explorem tots els vèrtexs i les arestes de G.

4. Distàncies en un graf

En una xarxa de carreteres en la qual hi ha diversos itineraris que uneixen dues ciutats és natural que ens preguntem quin itinerari és més curt, o quin és l'itinerari que assoleix la mínima distància entre les dues ciutats.

En aquest apartat definirem el concepte de distància entre dos vèrtexs d'un graf connex. Veurem que podem calcular aquesta distància i també resoldrem el problema de la distància i el camí mínim en un graf ponderat (graf amb un pes associat a cada aresta). Finalment, resoldrem el problema més general de trobar la distància entre un vèrtex i la resta dels vèrtexs del graf, i un camí que assoleixi aquesta distància, mitjançant els algorismes de Dijkstra i Floyd.

4.1. Distància entre vèrtexs

Definició 7

Donat un graf connex $G = (V,A)$, la **distància** entre dos dels seus vèrtexs és la mínima de les longituds dels camins que connecten aquests vèrtexs:

$$d_G(u,v) = \min\{\ell(C) \mid C \text{ és un } u-v \text{ camí}\}.$$

En el cas no connex, la distància entre dos vèrtexs d'un mateix component connex es defineix com en el cas anterior. En el cas de vèrtexs mútuament inaccessibles s'assigna el valor convencional infinit (∞).

Quan quedi clar en quin graf calculem la distància entre dos vèrtexs u i v posarem, simplement, $d(u,v)$.

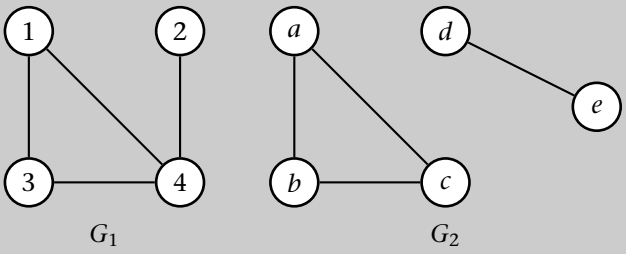
Definició 8

Donat un graf connex $G = (V,A)$, el **diàmetre** $D(G)$ és

$$D(G) = \max\{d_G(u,v) | u,v \in V\}.$$

Exemple 14

Considerem els grafs G_1 i G_2 de l'exemple 9.



A G_1, la $d(1,2) = 2$, $d(1,3) = 1$, $d(1,4) = 1$. El $D(G_1) = 2$.

A G_2, la $d(a,c) = d(a,b) = d(b,c) = 1$, $d(a,d) = \infty$, per tant, $D(G_2) = \infty$.

Exemple 15

Vegem que, donat un graf qualsevol no connex G, el graf complementari G^c és connex i de diàmetre ≤ 2.

Siguin x, y dos vèrtexs de components connexos diferents de G que, per tant, no són adjacents en G; en conseqüència ho són en el complementari G^c i $d_{G^c}(x,y) = 1$.

Siguin ara x, y del mateix component connex G_i de G. Atès que el graf no és connex per hipòtesi, existeix un altre component connex diferent G_j que conté un vèrtex $z \in V(G_j)$; aleshores x, y no són adjacents a z en el graf G i, en conseqüència, ho són en el complementari. Per tant, en el complementari, els vèrtexs x, y estan connectats per un camí a través de z, concretament el camí x,z,y, i resulta, doncs, que $d_{G^c}(x,y) \leq 2$ (observeu que no podem afirmar que la distància sigui 2, ja que podria passar que x, y fossin no adjacents en G, cas en el qual serien adjacents en el complementari).

A partir de la distància entre vèrtexs és possible una caracterització dels grafs bipartits.

Teorema 9

Un graf és bipartit si, i només si, tots els cicles són de longitud parella.

Demostració: Hem de demostrar dues implicacions:

1) Si el graf és bipartit, aleshores tots els cicles són de longitud parella.

2) Si tots els cicles d'un graf són de longitud parella, aleshores el graf és bipartit.

Suposem que el graf és $G = (V,A)$.

1) Suposem que el graf és (V_1,V_2)-bipartit; la conclusió es deriva fàcilment del fet que tot camí ha d'utilitzar alternativament vèrtexs de V_1 i de V_2.

2) Hem de definir adequadament una bipartició (V_1,V_2) del conjunt de vèrtexs de manera que el graf resulti ser (V_1,V_2)-bipartit.

Sigui $u_0 \in V$; definim els conjunts

$$V_1 = \{v \in V \mid d(u_0,v) \text{ és parell}\} \quad V_2 = \{v \in V \mid d(u_0,v) \text{ és senar}\}.$$

Aquests conjunts determinen una partició del conjunt de vèrtexs de graf i ara es tracta de veure que el graf és (V_1,V_2)-bipartit.

Per a comprovar-ho, hem de veure que no hi ha arestes que connectin vèrtexs d'un mateix conjunt de la partició; ho demostrarem per a parells de vèrtexs de V_1 i es procediria anàlogament per a V_2. Siguin, doncs, $u,v \in V_1$ i suposem que existeix l'aresta $a = \{u,v\} \in A$. Vegem que s'arriba a alguna contradicció, utilitzant el fet que tots els cicles són de longitud parella. Siguin R_1 un u_0-u camí de longitud mínima i R_2 un u_0-v camí de longitud mínima, tots dos de longituds parelles, per definició de V_1. Si es recorren els camins anteriors des de l'inici u_0, s'assolirà un vèrtex w_0 que és l'últim que comparteixen els camins R_1 i R_2. Per la minimalitat, els subcamins $u_0 - w_0$ sobre R_1 i R_2 han de ser de la mateixa longitud k. Considerem ara els subcamins $S_1 : w_0 - u$ sobre R_1 i $S_2 : w_0 - u$ sobre R_2; aleshores tenim $\ell(R_1) = k + \ell(S_1)$ i $\ell(R_2) = k + \ell(S_2)$, d'on veiem que la diferència $\ell(S_2) - \ell(S_1)$ és un número parell i, en conseqüència, $\ell(S_1)$ i $\ell(S_2)$ han de ser de la mateixa paritat. Considerant el camí format per aquestes seccions (passant per w_0), en resulta un $u - v$ camí de longitud parella. Conjuntament amb l'aresta $a = \{u,v\}$ es crearia un cicle de longitud senar, que contradiria la hipòtesi.

∎

Exercicis

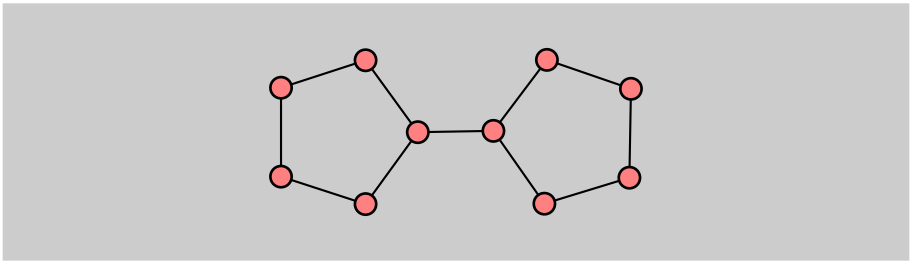
29. Comproveu que la distància donada entre vèrtexs d'un graf connex satisfà les propietats generals d'una mètrica:

a) $d(u,v) \geq 0$ i $d(u,v) = 0$ si, i només si, $u = v$;

b) $d(u,v) = d(v,u)$;

c) $d(u,v) \leq d(u,w) + d(w,v)$ (desigualtat triangular).

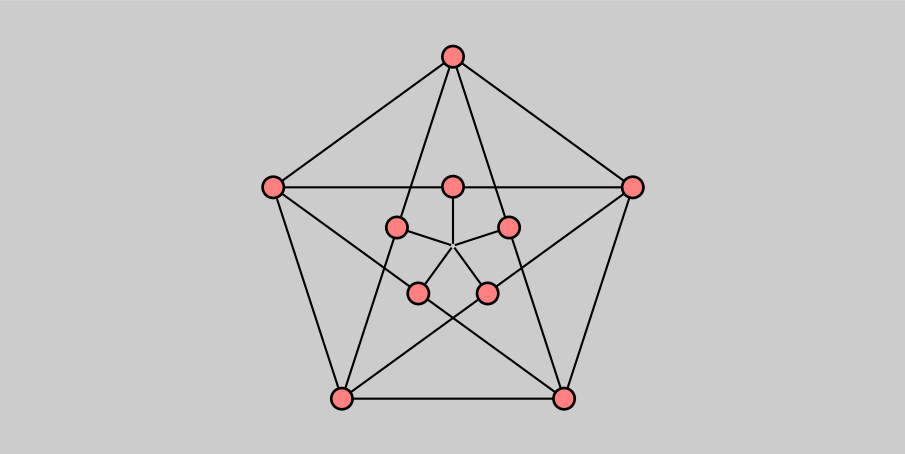
30. Indiqueu quin és el diàmetre del graf següent.



31. És certa la propietat: $u \approx v \Leftrightarrow d(u,v) = 1$?

32. Calculeu el diàmetre dels grafs K_n, $K_{n,m}$, C_n, T_n.

33. Utilitzeu la caracterització dels grafs bipartits per a decidir si el graf següent és bipartit o no.



34. Els grafs acíclics són bipartits. Cert o fals?

Solucions

29. Les propietats *a* i *b* són òbvies. Per a demostrar la propietat *c*, sigui C el camí que assoleix la $d(u,v)$. Sigui C_1 el camí que assoleix la $d(u,w)$ i C_2 el camí que assoleix la $d(w,v)$. Aleshores, si unim C_1 i C_2 tenim un camí que uneix u i v. Per la definició de distància, la longitud d'aquest camí ha de ser més gran o igual que la $d(u,v)$.

30. El diàmetre és 5.

31. Sí (acabeu de justificar la resposta).

32. $D(K_n) = 1$, perquè la distància d'un vèrtex a qualsevol altre sempre és 1.

$D(K_{n,m}) = 2$, perquè es pot anar d'un vèrtex a qualsevol altre recorrent dues arestes.

$D(C_n) = \frac{n}{2}$ si n és parell. $D(C_n) = \frac{n-1}{2}$ si n és senar.

$D(T_n) = n - 1$, que és la distància entre els dos extrems.

33. No és bipartit, ja que hi ha cicles de longitud 5.

34. Cert, en aplicació de la caracterització de bipartits en termes dels cicles, ja que no hi ha cap cicle i en particular, cap cicle de longitud senar.

4.2. El problema del camí mínim en un graf

Un exemple típic d'un graf és una xarxa de carreteres que connecten un conjunt de ciutats. El problema algorísmic més natural en aquest graf és trobar el camí més curt (**distància**) entre un parell de ciutats (**vèrtexs**).

La distància entre dos vèrtexs en un graf no ponderat es pot trobar, de manera eficient, si apliquem l'algorisme de cerca primàriament en amplada (**BFS** o *Breadth First Search*). Si el graf és ponderat, aleshores la distància entre dos vèrtexs no es pot calcular directament i cal aplicar algorismes específics.

L'**algorisme de Dijkstra** troba la distància entre dos vèrtexs mitjançant la construcció d'un arbre des del vèrtex inicial u_0 a cadascun dels altres vèrtexs del graf.

L'**algorisme de Floyd** troba la distància entre tots els parells de vèrtexs d'un graf.

Observació

Un graf simple es pot interpretar com un graf ponderat si associem a cada aresta un pes igual a 1.

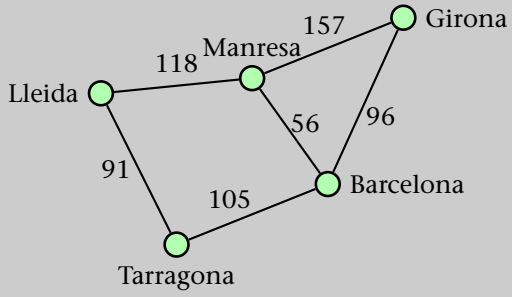
Definició 9

Un **graf ponderat** és una parella (G,w) on $G = (V,A)$ és un graf i w és una funció $w : A \to \mathbb{R}$ que assigna pesos a les arestes del graf.

Exemple 16

En una xarxa de comunicacions ens pot interessar assignar a cada aresta un pes indicatiu del temps que costa recórrer l'aresta en qüestió, o els quilòmetres, o el cost econòmic corresponent o altres aspectes que dependran del problema i del model que n'hàgim construït.

El graf següent és un graf ponderat. El pes de cada aresta és la distància quilomètrica entre dues ciutats.



Normalment els pesos són positius o nuls, però també es poden considerar situacions en què siguin negatius.

Definició 10

Donat un graf ponderat (G,w) i un camí $C : v_0,v_1,\ldots,v_k$ definim el **pes del camí** C com $w(C) = \sum_{i=1}^{k} w(v_{i-1},v_i)$, i la **distància** entre dos vèrtexs $u,v \in G$ com

$$d_G(u,v) = \min\{w(C) \mid C \text{ és un } u-v \text{ camí }\}.$$

Si el graf G no és ponderat (o tots els pesos són igual a 1) aleshores aquesta definició coincideix amb la que hem donat anteriorment (definició 7).

Com abans, en el cas de vèrtexs mútuament inaccessibles s'assigna el valor convencional ∞ a la seva distància.

Exemple 17

En el graf ponderat de l'exemple 16, el camí C_1 : Barcelona, Manresa, Lleida té un pes 174. En canvi, el camí C_2 : Barcelona, Tarragona, Lleida té un pes 196. La distància entre Barcelona i Lleida és 174 passant per Manresa.

Variants del problema del camí mínim

El problema del camí mínim admet diverses variants que es poden resoldre utilitzant adaptacions dels mateixos algorismes:

1) Camí mínim des d'un vèrtex inicial (*Single Source Shortest Path*): donat (G,w) i $s \in V$, cercar la $d(s,v)$ per a tot $v \in V$.

2) Camí mínim fins a un vèrtex destinació (*Single Destination Shortest Path*): donat (G,w) i $t \in V$, cercar la $d(v,t)$ per a tot $v \in V$.

3) Camí mínim entre un parell de vèrtexs (*Single Pair Shortest Path*): donat (G,w) i $s,t \in V$, cercar la $d(s,t)$.

4) Camí mínim entre tots els parells de vèrtexs (*All Pairs Shortest Path*): donat (G,w), cercar la $d(u,v)$ per a tot $u,v \in V$.

Exemple 18

En aquest exemple farem servir novament el graf ponderat de l'exemple 16.

Si ens imaginem les ciutats ordenades alfabèticament, aleshores la solució a les quatre variants del problema del camí mínim en aquest graf serà:

1) Camí mínim des de Barcelona:

(0,Barcelona), (96,Girona), (174,Lleida), (56,Manresa), (105,Tarragona).

2) Camí mínim fins a Tarragona:

(105,Barcelona), (201,Girona), (91,Lleida), (161,Manresa), (0,Tarragona).

3) Camí mínim entre Girona i Lleida: 270 passant per Barcelona i Manresa.

4) Camí mínim entre tots els parells de ciutats:

	Barcelona	Girona	Lleida	Manresa	Tarragona
Barcelona	0	96	174	56	105
Girona	96	0	270	152	201
Lleida	174	270	0	118	91
Manresa	56	152	118	0	161
Tarragona	105	201	91	161	0

A continuació estudiarem els algorismes específics per a resoldre les variants 1 i 4. Posposarem per als exercicis la resolució de les variants 2 i 3.

4.2.1. Algorisme de Dijkstra

L'algorisme de Dijkstra s'aplica sobre un graf (o digraf) ponderat i calcula la distància des d'un vèrtex inicial s a la resta de vèrtexs del graf. A cada pas, etiquetarem els vèrtexs en la forma $(dist(u),v)$ on $dist(u)$ és la distància mínima actual del vèrtexs s al vèrtexs u, i v és el predecessor d'u en el camí mínim que uneix s i u.

Formulació de l'algorisme de Dijkstra

Estructures necessàries per a la formulació de l'algorisme:

- Un graf ponderat (G,w) representat mitjançant una llista d'adjacències.
- Un conjunt U dels vèrtexs que s'han visitat, en l'ordre en què s'ha fet.
- Una taula de distàncies, $dist(\cdot)$, indexada pels vèrtexs de G, que registra la distància del vèrtex inicial als vèrtexs que es van visitant.
- Al final, la taula $dist(\cdot)$ registra la distància des del vèrtex inicial a la resta de vèrtexs.

Quan és possible visitar més d'un vèrtex, sempre triem el d'índex mínim en l'ordenació dels vèrtexs disponibles.

A cada pas es fixa la distància d'un dels vèrtexs del graf. Així, després de n passos haurem calculat la distància a tots els vèrtexs del graf.

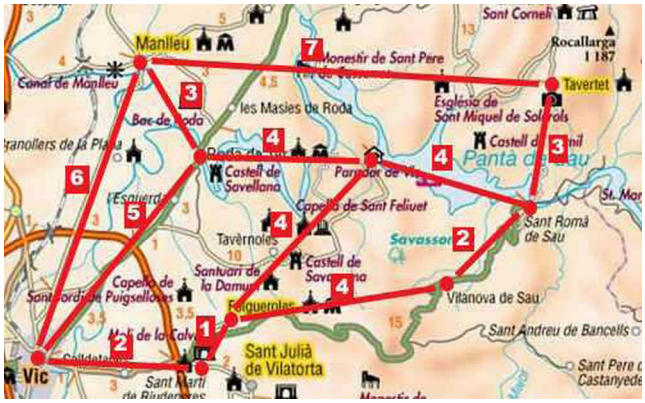
Entrada : (G,w) d'ordre n i un vèrtex inicial s.
Sortida : La distància, $dist(\cdot)$, de s a la resta de vèrtexs.
algorisme *Dijkstra*(G,s)
  inici
    $U \leftarrow \emptyset$
    per $v \in V \backslash \{s\}$
      $dist(v) \leftarrow \infty$
      Etiquetem v amb $(dist(v),s)$
    fiper
    $dist(s) \leftarrow 0$
    Etiquetem s amb $(dist(s),s)$
    per $i \leftarrow 0$ **fins** $n-1$
      u_i vèrtex que assoleix el $\min\{dist(v) \mid v \in V - U\}$
      $U \leftarrow U \cup \{u_i\}$
      per $v \in V - U$ adjacent a u_i
        si $dist(u_i) + w(u_i,v) < dist(v)$
          aleshores $dist(v) \leftarrow dist(u_i) + w(u_i,v)$
            Etiquetem v amb $(dist(v),u_i)$
        fisi
      fiper
    fiper
    retorn $(dist)$
  fi

L'algorisme es pot utilitzar per a obtenir un camí de longitud mínima entre el vèrtex inicial i qualsevol vèrtex, ja que després d'aplicar l'algorisme, tot vèrtex v té associada una etiqueta $(dist(v),u_i)$ indicativa de la distància del vèrtex v al vèrtex de partida, $dist(v) = d(s,v)$, i del camí seguit per a calcular la distància. Si $v \neq s$ s'obté un $s-v$ camí de longitud mínima amb $s = q_0,q_1,\ldots,q_k = v$, on els q_i estan etiquetats amb $(dist(q_i),q_{i-1})$ per a $i = 1,\ldots,k$.

Simulació de l'algorisme de Dijkstra

Exemple 19

Considerem el graf definit sobre el mapa següent, que representa les distàncies aproximades entre diverses localitzacions de la comarca d'Osona:



Podem utilitzar l'algorisme de Dijkstra per a trobar la distància mínima entre Vic i la resta de localitats. Utilitzarem una taula com la següent per a representar els diferents passos de l'algorisme de Dijkstra:

Vèrtexs	V	M	CS	SM	F	VS	SR	PV	T
U	0	0	0	0	0	0	0	0	0
$(dist(\cdot),\cdot)$	$(0,V)$	(∞,V)	(∞,V)	(∞,V)	(∞,V)	(∞,V)	(∞,V)	(∞,V)	(∞,V)

Aquesta taula representa l'estat inicial de l'algorisme de Dijkstra. Els vèrtexs són: **V**ic, **M**anlleu, **C**astell de **S**avellana, **S**ant **M**artí de Riudeperes, **F**olgueroles, **V**ilanova de **S**au, **S**ant **R**omà de Sau, **P**arador de **V**ic, **T**avertet. El conjunt U està representat per un **mapa de bits**, és a dir, conté un **1** si el vèrtex corresponent pertany a U o **0** en cas contrari.

$i = 0$. El vèrtex de pes mínim és el vèrtex V. Podem etiquetar els vèrtexs M, CS, SM.

Vèrtexs	V	M	CS	SM	F	VS	SR	PV	T
U	**1**	0	0	0	0	0	0	0	0
$(dist(\cdot),\cdot)$	$(0,V)$	**(6,V)**	**(5,V)**	**(2,V)**	(∞,V)	(∞,V)	(∞,V)	(∞,V)	(∞,V)

$i = 1$. El vèrtex de pes mínim és el vèrtex SM. Podem etiquetar el vèrtex F.

Vèrtexs	V	M	CS	SM	F	VS	SR	PV	T
U	1	0	0	**1**	0	0	0	0	0
$(dist(\cdot),\cdot)$	$(0,V)$	$(6,V)$	$(5,V)$	$(2,V)$	**(3,SM)**	(∞,V)	(∞,V)	(∞,V)	(∞,V)

$i = 2$. El vèrtex de pes mínim és el vèrtex F. Podem etiquetar els vèrtexs VS i PV.

Vèrtexs	V	M	CS	SM	F	VS	SR	PV	T
U	1	0	0	1	**1**	0	0	0	0
$(dist(\cdot),\cdot)$	$(0,V)$	$(6,V)$	$(5,V)$	$(2,V)$	$(3,SM)$	**(7,F)**	(∞,V)	**(7,F)**	(∞,V)

i = 3. El vèrtex de pes mínim és el vèrtex CS però no etiquetem cap nou vèrtex.

Vèrtexs	V	M	CS	SM	F	VS	SR	PV	T
U	1	0	**1**	1	1	0	0	0	0
$(dist(\cdot),\cdot)$	$(0,V)$	$(6,V)$	$(5,V)$	$(2,V)$	$(3,SM)$	$(7,F)$	(∞,V)	$(7,F)$	(∞,V)

i = 4. El vèrtex de pes mínim és el vèrtex M. Podem etiquetar el vèrtex T.

Vèrtexs	V	M	CS	SM	F	VS	SR	PV	T
U	1	**1**	1	1	1	0	0	0	0
$(dist(\cdot),\cdot)$	$(0,V)$	$(6,V)$	$(5,V)$	$(2,V)$	$(3,SM)$	$(7,F)$	(∞,V)	$(7,F)$	**(13,M)**

i = 5. El vèrtex de pes mínim és el vèrtex VS. Podem etiquetar el vèrtex SR.

Vèrtexs	V	M	CS	SM	F	VS	SR	PV	T
U	1	1	1	1	1	**1**	0	0	0
$(dist(\cdot),\cdot)$	$(0,V)$	$(6,V)$	$(5,V)$	$(2,V)$	$(3,SM)$	$(7,F)$	**(9,VS)**	$(7,F)$	$(13,M)$

i = 6. El vèrtex de pes mínim és el vèrtex PV però no etiquetem cap nou vèrtex.

Vèrtexs	V	M	CS	SM	F	VS	SR	PV	T
U	1	1	1	1	1	1	0	**1**	0
$(dist(\cdot),\cdot)$	$(0,V)$	$(6,V)$	$(5,V)$	$(2,V)$	$(3,SM)$	$(7,F)$	$(9,VS)$	$(7,F)$	$(13,M)$

i = 7. El vèrtex de pes mínim és el vèrtex SR. Podem etiquetar el vèrtex T.

Vèrtexs	V	M	CS	SM	F	VS	SR	PV	T
U	1	1	1	1	1	1	**1**	1	0
$(dist(\cdot),\cdot)$	$(0,V)$	$(6,V)$	$(5,V)$	$(2,V)$	$(3,SM)$	$(7,F)$	$(9,VS)$	$(7,F)$	**(12,SR)**

i = 8. Finalment, el vèrtex de pes mínim és el vèrtex T però no etiquetem cap nou vèrtex.

Vèrtexs	V	M	CS	SM	F	VS	SR	PV	T
U	1	1	1	1	1	1	1	1	**1**
$(dist(\cdot),\cdot)$	$(0,V)$	$(6,V)$	$(5,V)$	$(2,V)$	$(3,SM)$	$(7,F)$	$(9,VS)$	$(7,F)$	$(12,SR)$

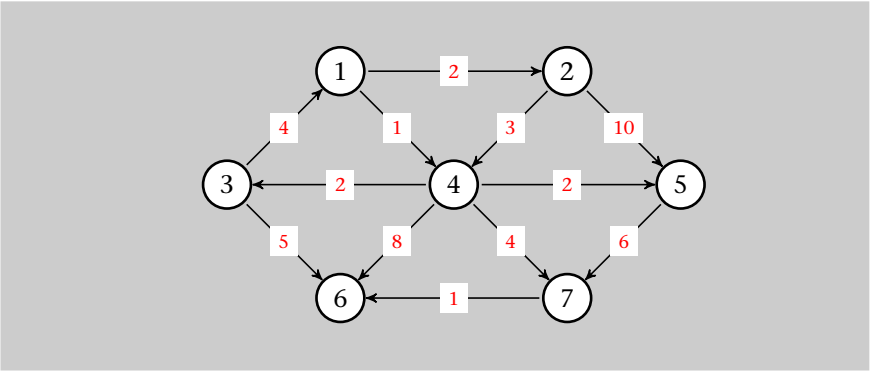
La darrera taula mostra la distància mínima entre Vic i la resta de ciutats. Observeu que també és possible reconstruir l'itinerari que caldria seguir per a desplaçar-se de Vic a qualsevol de les ciutats. Per exemple, la distància de Vic a Tavertet és 12 i l'itinerari es reconstrueix a partir de l'etiquetatge de la darrera taula: T(12,SR), SR(9,VS), VS(7,F), F(3,SM), SM(2,V). L'itinerari serà, Vic, Sant Martí de Riudeperes, Folgueroles, Vilanova de Sau, Sant Romà de Sau, Tavertet.

A partir d'aquestes taules que presenten cadascun dels passos, es pot construir la **taula de l'algorisme de Dijkstra**, que inclou cadascuna de les files $(dist(\cdot),\cdot)$ (es marca amb un asterisc el vèrtex que es visita):

V	M	CS	SM	F	VS	SR	PV	T
(0,V)	(∞,V)	(∞,V)	(∞,V)	(∞,V)	(∞,V)	(∞,V)	(∞,V)	(∞,V)
(0,V)*	(6,V)	(5,V)	(2,V)	(∞,V)	(∞,V)	(∞,V)	(∞,V)	(∞,V)
(0,V)	(6,V)	(5,V)	(2,V)*	(3,SM)	(∞,V)	(∞,V)	(∞,V)	(∞,V)
(0,V)	(6,V)	(5,V)	(2,V)	(3,SM)*	(7,F)	(∞,V)	(7,F)	(∞,V)
(0,V)	(6,V)	(5,V)*	(2,V)	(3,SM)	(7,F)	(∞,V)	(7,F)	(∞,V)
(0,V)	(6,V)*	(5,V)	(2,V)	(3,SM)	(7,F)	(∞,V)	(7,F)	(13,M)
(0,V)	(6,V)	(5,V)	(2,V)	(3,SM)	(7,F)*	(9,VS)	(7,F)	(13,M)
(0,V)	(6,V)	(5,V)	(2,V)	(3,SM)	(7,F)	(9,VS)	(7,F)*	(13,M)
(0,V)	(6,V)	(5,V)	(2,V)	(3,SM)	(7,F)	(9,VS)*	(7,F)	(12,SR)
(0,V)	(6,V)	(5,V)	(2,V)	(3,SM)	(7,F)	(9,VS)	(7,F)	(12,SR)*

Exemple 20

Considerem el digraf definit pel gràfic següent i calculem les distàncies des del vèrtex 1 a la resta de vèrtexs.



En el cas dels digrafs, la distància d'un arc només té sentit en la direcció indicada per l'arc. Si no hi ha cap arc en sentit contrari, s'utilitza com a distància el valor ∞, igual que en els casos on no hi ha cap aresta.

En aquest cas, la taula de l'algorisme de Dijkstra del digraf és aquesta (s'assenyala amb un asterisc el vèrtex que es visita):

1	2	3	4	5	6	7
(0,1)	(∞,1)	(∞,1)	(∞,1)	(∞,1)	(∞,1)	(∞,1)
(0,1)*	(2,1)	(∞,1)	(1,1)	(∞,1)	(∞,1)	(∞,1)
(0,1)	(2,1)	(3,4)	(1,1)*	(3,4)	(9,4)	(5,4)
(0,1)	(2,1)*	(3,4)	(1,1)	(3,4)	(9,4)	(5,4)
(0,1)	(2,1)	(3,4)*	(1,1)	(3,4)	(8,3)	(5,4)
(0,1)	(2,1)	(3,4)	(1,1)	(3,4)*	(8,3)	(5,4)
(0,1)	(2,1)	(3,4)	(1,1)	(3,4)	(6,7)	(5,4)*
(0,1)	(2,1)	(3,4)	(1,1)	(3,4)	(6,7)*	(5,4)

De la darrera fila de la taula es dedueix que la distància del vèrtex 1 a la resta de vèrtexs val $d(1,1) = 0$,$d(1,2) = 2$,$d(1,3) = 3$,$d(1,4) = 1$,$d(1,5) = 3$,$d(1,6) = 6$, $d(1,7) = 5$.

Anàlisi de l'algorisme de Dijkstra

Per a analitzar aquest algorisme el dividirem en dues parts:

1) S'inicia una taula de mida n amb una complexitat $O(n)$.

2) El bucle principal s'executa n vegades. En el pas i-èsim calculem el mínim d'una llista que conté $n - i$ elements. Això es pot fer amb $n - i$ comparacions.

En el bucle més intern actualitzem les etiquetes dels vèrtexs adjacents al vèrtex analitzat. El nombre màxim de vèrtexs adjacents que actualitzem en el pas i-èsim és $n - i - 1$.

Resumint, en el bucle principal fem

$$\sum_{i=0}^{n-1} n - i + \sum_{i=0}^{n-1} n - i - 1 = n^2$$

operacions elementals.

Això ens dóna una complexitat $O(n^2)$.

Tot l'algorisme, doncs, tindrà una complexitat $\max\{O(n),O(n^2)\} = O(n^2)$, independentment del nombre d'arestes del graf.

4.2.2. Camí mínim en un graf no ponderat

Si el graf és no ponderat (o totes les arestes tenen pes 1) aleshores podem utilitzar l'algorisme d'exploració en amplada (BFS) per a calcular la distància entre un vèrtex inicial i la resta de vèrtexs del graf.

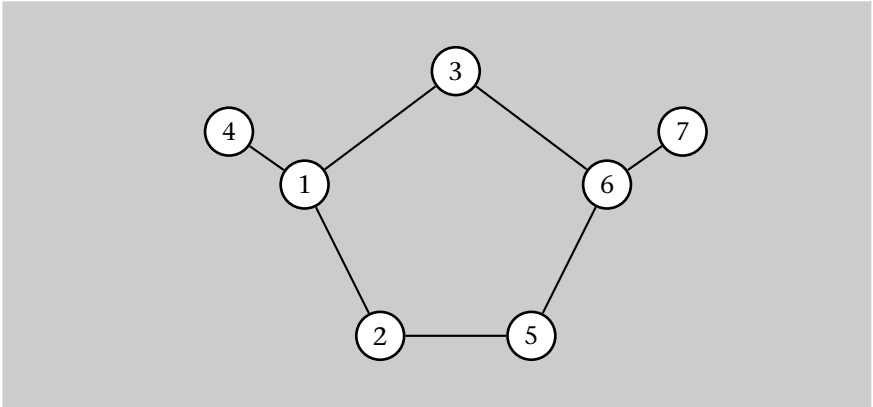
Les estructures de dades necessàries per a formular l'algorisme són les mateixes que en el BFS si afegim una taula $dist(\cdot)$ que emmagatzemi les distàncies del vèrtex inicial a la resta de vèrtexs.

Entrada : $G = (V,A)$ no ponderat, $s \in V$
Sortida : La distància, $dist(\cdot)$, de s a la resta de vèrtexs.
algorisme *Distàncies_no_ponderat*(G,s)

```
inici
  Q ← ∅
  per w ∈ V
      estat[w] ← 0
      dist[w] ← ∞
  fiper
  estat[s] ← 1
  dist[s] ← 0
  afegir(Q,s)
  mentre Q ≠ ∅
          w ← primer(Q)
          per u adjacent a w
              si estat[u] = 0
                 aleshores afegir(Q,u)
                           estat[u] ← 1
                           dist[u] ← dist[w] + 1
              fisi
          fiper
          eliminar(Q)
  fimentre
  retorn (dist)
fi
```

Exemple 21

Considerem el graf no ponderat que es representa en la figura següent.



La taula registra el funcionament de l'algorisme per a aquest graf, amb vèrtex d'inici $s = 1$.

Q	Vèrtex afegit	Vèrtex eliminat	*dist*
1	1	-	[0,∞,∞,∞,∞,∞,∞]
12	2	-	[0,1,∞,∞,∞,∞,∞]
123	3	-	[0,1,1,∞,∞,∞,∞]
1234	4	-	[0,1,1,1,∞,∞,∞]
234	-	1	[0,1,1,1,∞,∞,∞]
2345	5	-	[0,1,1,1,2,∞,∞]
345	-	2	[0,1,1,1,2,∞,∞]
3456	6	-	[0,1,1,1,2,2,∞]
456	-	3	[0,1,1,1,2,2,∞]
56	-	4	[0,1,1,1,2,2,∞]
6	-	5	[0,1,1,1,2,2,∞]
67	7	-	[0,1,1,1,2,2,3]
7	-	6	[0,1,1,1,2,2,3]
∅	-	7	[0,1,1,1,2,2,3]

Si comparem aquest algorisme amb l'algorisme de Dijkstra aplicat a un graf no ponderat, podem observar que mentre el de Dijkstra té una complexitat $O(n^2)$, el BFS té una complexitat $O(n+m)$. Per a grafs poc densos (poques arestes), l'algorisme BFS té un comportament més eficient que l'algorisme de Dijkstra.

4.2.3. Algorisme de Floyd

El problema de cercar els camins mínims entre tots els parells de vèrtexs d'un graf es pot resoldre si apliquem n vegades l'algorisme de Dijkstra:

Entrada : (G,w) d'ordre n
Sortida : La distància, $d(\cdot,\cdot)$, entre tots els parells de vèrtexs.
algorisme *tots_els_parells*(G)
  inici
    per $s \in V$
      $d(s,\cdot) \leftarrow Dijkstra(G,s)$
    fiper
    retorn (d)
  fi

que tindria una complexitat $O(n^3)$.

Objectiu de l'algorisme de Floyd

L'algorisme de Floyd cerca els camins mínims entre tots els parells de vèrtexs d'un graf.

Una altra alternativa és utilitzar un algorisme específic, comparable en eficiència a l'algorisme de Dijkstra, però amb un comportament més bo per a grafs densos. És l'**algorisme de Floyd**.

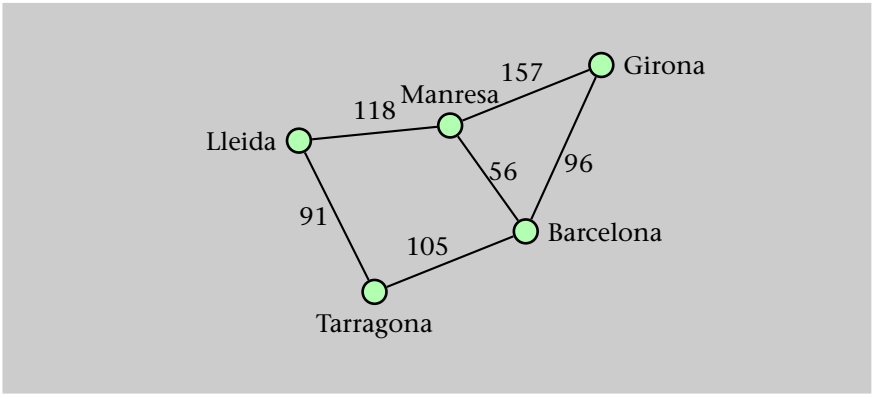
L'algorisme de Floyd té en consideració els vèrtexs ordenats i, en el pas k-èsim, compara el pes del camí obtingut fins al moment, utilitzant els $k-1$ vèrtexs anteriors, amb el camí obtingut afegint el vèrtex k-èsim.

Suposarem els vèrtexs etiquetats $V = \{1,2,3,\ldots,n\}$ i utilitzarem una matriu bidimensional d_{ij} $(1 \leq i,j \leq n)$ per a emmagatzemar les distàncies.

Entrada : (G,w) d'ordre n
Sortida : La distància, $d(\cdot,\cdot)$, entre tots els parells de vèrtexs.
algorisme *Floyd*(G)
  inici
    per $i \leftarrow 1$ **fins** n
      per $j \leftarrow 1$ **fins** n
        si $i = j$ **aleshores** $d_{ij}^0 \leftarrow 0$ **fisi**
        si $(i,j) \in A$ **aleshores** $d_{ij}^0 \leftarrow w(i,j)$ **fisi**
        si $(i,j) \notin A$ **aleshores** $d_{ij}^0 \leftarrow \infty$ **fisi**
      fiper
    fiper
    per $k \leftarrow 1$ **fins** n
      per $i \leftarrow 1$ **fins** n
        per $j \leftarrow 1$ **fins** n
          $d_{ij}^k \leftarrow \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
        fiper
      fiper
    fiper
    retorn (d_{ij}^n)
  fi

Exemple 22

Si apliquem l'algorisme de Floyd al graf de l'exemple 16,



obtenim la sèrie de matrius bidimensionals (suposem les ciutats ordenades alfabèticament):

$$d^0 = \begin{pmatrix} 0 & 96 & \infty & 56 & 105 \\ 96 & 0 & \infty & 157 & \infty \\ \infty & \infty & 0 & 118 & 91 \\ 56 & 157 & 118 & 0 & \infty \\ 105 & \infty & 91 & \infty & 0 \end{pmatrix} \quad d^1 = \begin{pmatrix} 0 & 96 & \infty & 56 & 105 \\ 96 & 0 & \infty & 152 & 201 \\ \infty & \infty & 0 & 118 & 91 \\ 56 & 152 & 118 & 0 & 161 \\ 105 & 201 & 91 & 161 & 0 \end{pmatrix}$$

$$d^2 = \begin{pmatrix} 0 & 96 & \infty & 56 & 105 \\ 96 & 0 & \infty & 152 & 201 \\ \infty & \infty & 0 & 118 & 91 \\ 56 & 152 & 118 & 0 & 161 \\ 105 & 201 & 91 & 161 & 0 \end{pmatrix} \quad d^3 = \begin{pmatrix} 0 & 96 & \infty & 56 & 105 \\ 96 & 0 & \infty & 152 & 201 \\ \infty & \infty & 0 & 118 & 91 \\ 56 & 152 & 118 & 0 & 161 \\ 105 & 201 & 91 & 161 & 0 \end{pmatrix}$$

$$d^4 = \begin{pmatrix} 0 & 96 & 174 & 56 & 105 \\ 96 & 0 & 270 & 152 & 201 \\ 174 & 270 & 0 & 118 & 91 \\ 56 & 152 & 118 & 0 & 161 \\ 105 & 201 & 91 & 161 & 0 \end{pmatrix} \quad d^5 = \begin{pmatrix} 0 & 96 & 174 & 56 & 105 \\ 96 & 0 & 270 & 152 & 201 \\ 174 & 270 & 0 & 118 & 91 \\ 56 & 152 & 118 & 0 & 161 \\ 105 & 201 & 91 & 161 & 0 \end{pmatrix}$$

Observeu que d^5 coincideix amb la taula obtinguda en l'exemple 18.

Anàlisi de l'algorisme de Floyd

L'algorisme de Floyd és molt fàcil d'analitzar. Bàsicament té dues parts: la iniciació de la matriu de distàncies i el càlcul de les distàncies.

La iniciació té una complexitat $O(n^2)$ i el càlcul de les distàncies té una complexitat $O(n^3)$; així, tot l'algorisme tindrà una complexitat $O(n^3)$, independentment del nombre d'arestes i comparable amb l'eficiència de l'algorisme de Dijkstra aplicat n vegades.

Exercicis

35. La taula següent representa la distància entre diversos aeroports units per una línia aèria.

	A	B	C	D	E	F	G
A	0	5	3	2	-	-	-
B	5	0	2	-	3	-	1
C	3	2	0	7	7	-	-
D	2	-	7	0	2	6	-
E	-	3	7	2	0	1	1
F	-	-	-	6	1	0	-
G	-	1	-	-	1	-	0

a) Quina és la distància mínima entre l'aeroport A i la resta d'aeroports?

b) Quin és el nombre mínim de transbordaments d'avió que cal fer per a anar de l'aeroport A a la resta d'aeroports?

36. Si en l'algorisme de Dijkstra modifiquem la condició

$$dist(u_i) + w(u_i,v) \leq dist(v)$$

canviant $\leq$ per $<$, com repercutirà en el funcionament de l'algorisme? S'obtindrà la mateixa distància entre dos vèrtexs? S'obtindrà el mateix camí?

37. Suposeu que numerem els vèrtexs del graf K_n, $V = \{1,2,3,\ldots,n\}$, i a cada aresta assignem un pes, $w(i,j) = |j - i|$. Quina serà la distància entre dos vèrtexs qualssevol?

38. Modifiqueu l'algorisme de Dijkstra per a obtenir la solució a la variant del problema del camí mínim entre una parella de vèrtexs (*single pair shortest path*), és a dir, donats (G,w) i dos vèrtexs $s,t \in V$, trobar la distància mínima de s a t.

39. A partir de l'algorisme de Dijkstra proposeu una solució a la variant del problema del camí mínim fins a un vèrtex destinació (*single destination shortest path*), és a dir, donats (G,w) i un vèrtex $t \in V$, trobar la distància mínima de tots els vèrtexs de G a t.

40. La taula següent representa el temps necessari per a connectar directament diversos nodes d'una xarxa. El símbol "-" significa que els dos nodes no són accessibles directament. Fent servir l'algorisme de Floyd, trobeu el temps mínim necessari per a connectar dos a dos tots els parells de nodes de la xarxa:

	1	2	3	4	5
1	0	3	8	-	4
2	-	0	-	1	7
3	-	4	0	-	-
4	2	-	5	0	-
5	-	-	-	6	0

Solucions

35. En el primer cas hem d'aplicar l'algorisme de Dijkstra sobre el graf que s'obté a partir de la taula de distàncies següent:

A	B	C	D	E	F	G
(0,A)	(∞,A)	(∞,A)	(∞,A)	(∞,A)	(∞,A)	(∞,A)
(0,A)*	(5,A)	(3,A)	(2,A)	(∞,A)	(∞,A)	(∞,A)
(0,A)	(5,A)	(3,A)	(2,A)*	(4,D)	(8,D)	(∞,A)
(0,A)	(5,A)	(3,A)*	(2,A)	(4,D)	(8,D)	(∞,A)
(0,A)	(5,A)	(3,A)	(2,A)	(4,D)*	(5,E)	(5,E)
(0,A)	(5,A)*	(3,A)	(2,A)	(4,D)	(5,E)	(5,E)
(0,A)	(5,A)	(3,A)	(2,A)	(4,D)	(5,E)*	(5,E)
(0,A)	(5,A)	(3,A)	(2,A)	(4,D)	(5,E)	(5,E)*

La darrera fila de la taula ens dóna les distàncies mínimes entre l'aeroport A i la resta d'aeroports.

En el segon cas, podem considerar el mateix graf però ara sense pesos, és a dir, només ens interessa saber el nombre d'arestes que uneixen l'aeroport A

amb la resta d'aeroports. En aquest cas, podem aplicar algorisme per a calcular distàncies en el cas no ponderat:

Q	Vèrtex afegit	Vèrtex eliminat	*dist*
A	A	-	[0,∞,∞,∞,∞,∞,∞]
AB	B	-	[0,1,∞,∞,∞,∞,∞]
ABC	C	-	[0,1,1,∞,∞,∞,∞]
ABCD	D	-	[0,1,1,1,∞,∞,∞]
BCD	-	A	[0,1,1,1,∞,∞,∞]
BCDE	E	-	[0,1,1,1,2,∞,∞]
BCDEG	G	-	[0,1,1,1,2,2,∞]
CDEG	-	B	[0,1,1,1,2,2,∞]
DEG	-	C	[0,1,1,1,2,2,∞]
DEGF	F	-	[0,1,1,1,2,2,2]
EGF	-	D	[0,1,1,1,2,2,2]
GF	-	E	[0,1,1,1,2,2,2]
F	-	G	[0,1,1,1,2,2,2]
∅	-	-	[0,1,1,1,2,2,2]

La llista [0,1,1,1,2,2,2] ens dóna el nombre de transbordaments que caldrà fer per a connectar l'aeroport A amb la resta d'aeroports.

36. L'algorisme continua funcionant correctament i, per tant, obtenim la mateixa distància mínima, però el camí pot ser diferent.

37. Observem que si apliquem l'algorisme de Dijkstra sobre un graf ponderat, a cada pas fixem la distància a un vèrtex que ja no es mou. Si partim del vèrtex i, en el primer pas fixarem la distància del vèrtex $i-1$ o $i+1$ que són els que estan a distància 1 de i. En el pas següent, tant si triem $i-1$ com $i+1$ fixarem la distància del vèrtex $i+2$, i així successivament. Per tant, la distància entre i i j serà el pes de l'aresta que uneix i i j: $d(i,j) = |j-i|$.

38. Com que, a cada pas, l'algorisme de Dijkstra fixa la distància a un vèrtex que ja no es modificarà, n'hi haurà prou de modificar el bucle de l'algorisme perquè s'aturi quan fixem la distància al vèrtex t:

```
Entrada : (G,w) d'ordre n i dos vèrtexs s,t.
Sortida : La distància, dist, de s a t.
algorisme SinglePairShortestPath(G,s,t)
  inici
    U ← ∅
    per v ∈ V
        dist(v) ← ∞
    fiper
    dist(s) ← 0
    i ← 0
    mentre (i < n) ∧ (t ∉ U)
              u_i vèrtex que assoleix el min{dist(v) | v ∈ V − U}
              U ← U ∪ {u_i}
              per v ∈ V − U adjacent a u_i
                  si dist(u_i) + w(u_i,v) < dist(v)
                     aleshores dist(v) ← dist(u_i) + w(u_i,v)
                               Etiquetem v amb (dist(v),u_i)
                  fisi
              fiper
              i ← i + 1
    fimentre
    retorn (dist(t))
  fi
```

Aquest algorisme és equivalent (té la mateixa complexitat) que l'algorisme de Dijkstra, però si només necessitem cercar la distància entre dos vèrtexs concrets, té un comportament més bo que la versió general.

39. En el cas que G sigui un graf simple (no dirigit) aleshores es pot utilitzar directament l'algorisme de Dijkstra per a resoldre aquesta variant:

Entrada : (G,w) no dirigit, d'ordre n i un vèrtex t.
Sortida : La distància, $d(\cdot)$, de tots els vèrtexs a t.
algorisme *SingleDestinationShortestPath*(G,t)
  inici
    retorn (*Dijkstra(G,t)*)
  fi

Si G és un digraf (graf dirigit), només cal crear un nou graf G' amb els mateixos vèrtexs que G i de manera que els seus arcs siguin els arcs de G amb les orientacions canviades. Tot seguit, només cal aplicar l'algorisme de Dijkstra a aquest nou graf amb origen en el vèrtex t:

Entrada : (G,w) digraf, d'ordre n i un vèrtex t.
Sortida : La distància, $d(\cdot)$, de tots els vèrtexs a t.
algorisme *SingleDestinationShortestPath*(G,t)
  inici
    Generar $G' = (V,A')$ a partir de G
    retorn (*Dijkstra(G',t)*)
  fi

40. La taula inicial per a aplicar l'algorisme de Floyd és:

$$d^0 = \begin{pmatrix} 0 & 3 & 8 & \infty & 4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & 5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}.$$

Després d'aplicar l'algorisme obtenim la taula

$$d^5 = \begin{pmatrix} 0 & 3 & 8 & 4 & 4 \\ 3 & 0 & 6 & 1 & 7 \\ 7 & 4 & 0 & 5 & 11 \\ 2 & 5 & 5 & 0 & 6 \\ 8 & 11 & 11 & 6 & 0 \end{pmatrix},$$

on el valor de la posició (i,j) representa el temps mínim per a connectar el node i amb el j.

Exercicis d'autoavaluació

1. Quants camins de longitud k = 2, 3, 4 i 5 entre dos vèrtexs diferents té el graf K_4? Trobeu una fórmula per a calcular el nombre de camins de longitud k entre dos vèrtexs diferents de K_n.

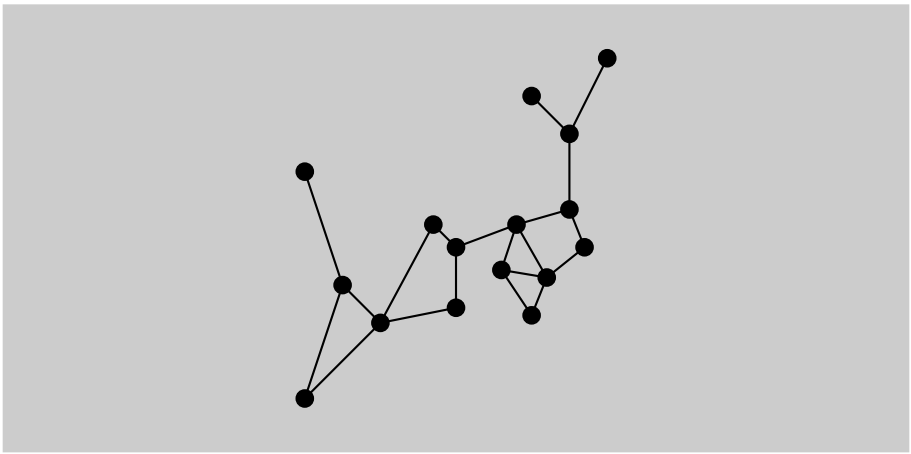
2. L'algorisme DFS utilitza una pila per a recordar l'ordre en què s'han visitat els vèrtexs i poder recuperar-los. La pila es pot substituir per una crida recursiva. Escriviu la versió recursiva de l'algorisme DFS i comproveu-ne el funcionament amb el graf de l'exemple 7.

3. Sigui $G = (V,A)$ un graf de disset vèrtexs amb quatre components connexos. Demostreu que almenys un dels components té un mínim de cinc vèrtexs.

4. Demostreu que un graf amb seqüència de graus 2,2,2,2,2 ha de ser necessàriament connex.

5. Sigui $G = (V,A)$ un graf d'ordre $|V| = n \geq 2$ en què el grau de cada vèrtex $v \in V$ satisfà $g(v) \geq \frac{1}{2}(n-1)$. Demostreu que G és un graf connex. Estudieu si podem afirmar si són connexos els grafs amb les seqüències de graus: 2,2,2,2; 2,2,2,2,2; 3,3,3,3,3,3.

6. Una aresta a d'un graf connex $G = (V,A)$ es diu que és una **aresta-pont** si el graf $G - a = (V,A - \{a\})$ és no connex. Considereu el graf de la figura següent i indiqueu quines són les arestes pont.



7. Sigui G un graf connex que només conté vèrtexs de grau parell. Demostreu que G no pot contenir cap aresta-pont.

8. Un procés de manufactura comença amb un tros de fusta en brut. La fusta ha de ser tallada, polida, foradada i pintada. Tallar s'ha de fer abans de foradar i, polir, abans de pintar. Considereu que els requeriments de temps per a cada operació són els següents: tallar demana una unitat; polir, també una; pintar, quatre si la fusta no és tallada i dues si ja ho és; foradar demana tres unitats si la fusta no és polida, cinc si és polida però no pintada i set si ja és pintada.

a) Quin ordre de realització del procés cal seguir per a minimitzar el temps total? És únic?

b) Quin cost hauria de tenir foradar si ho volem fer després de polir però abans de pintar, perquè el cost continuï essent mínim?

9. Una pagesa té una garrafa de vuit litres plena de vi, i disposa només de dues garrafes buides de cinc i tres litres, respectivament. Quin és el nombre mínim de passos que ha de fer per a repartir el vi en dues parts iguals?

10. Proposeu un algorisme eficient per a calcular el diàmetre d'un graf.

11. Es defineix el **centre** d'un graf G com el vèrtex tal que la suma de distàncies a la resta de vèrtexs sigui mínima. Proposeu un algorisme eficient per a calcular el centre d'un graf G.

Solucionari

1. Per a $k = 2$ hi ha dos camins. Si $k = 3$ també n'hi ha 2. Per a $k = 4$ i $k = 5$ no n'hi ha cap.

En general, un camí de longitud k passarà per $k+1$ vèrtexs sense repetir-ne cap. Donats dos vèrtexs diferents, el nombre de camins entre ells serà donat per les maneres diferents de triar la resta de $k-1$ vèrtexs d'entre els $n-2$ vèrtexs restants (tots excepte els dos vèrtexs extrems fixats). És a dir, el nombre de camins de longitud k, si $k \leq n-1$, serà $V(n-2,k-1)$, el nombre de $(k-1)$-mostres ordenades sense repetició del conjunt dels $n-2$ vèrtexs.

Notació

Recordeu que el nombre de r-mostres ordenades sense repetició d'un conjunt X de n elements es denota $V(n,r)$. Es pot calcular per la fórmula, $V(n,r) = n \cdot (n-1) \cdot (n-2) \cdots (n-(r-1))$.

2. Versió recursiva de l'algorisme DFS:

```
Entrada : G = (V,A), v ∈ V
Sortida : R, vèrtexs visitats
algorisme DFS(G,v)
  inici
    R ← ∅
    per w ∈ V
        estat[w] ← 0
    fiper
    dfsrec(G,v,R,estat)
    retorn (R)
  fi

funció dfsrec(G,v,R,estat)
  inici
    estat[v] ← 1
    afegir(R,v)
    per w adjacent a v
        si estat[w] = 0
            aleshores dfsrec(G,w,R,estat)
        fisi
    fiper
  fi
```

3. Sigui V el conjunt de vèrtexs i sigui $|V| = 17$. Si G_1, G_2, G_3, G_4 són els components connexos de G i V_1, V_2, V_3, V_4, els conjunts de vèrtexs respectius, aleshores tenim $\sum_{i=1}^{4} |V_i| = |V| = 17$. Si no es complís l'afirmació que cal demostrar, seria $|V_1| \leq 4$, $|V_2| \leq 4$, $|V_3| \leq 4$, $|V_4| \leq 4$, ja que l'ordre d'un graf sempre és un enter. D'aquí en resulta una contradicció: $17 = |V_1|+|V_2|+|V_3|+|V_4| \leq 4+4+4+4 = 16$.

4. Sigui $G = (V,A)$ un graf d'ordre $n = 5$ amb la seqüència de graus anterior. Si fos no connex tindria un mínim de dos components connexos, que podem suposar que són G_1 i G_2 (i podria haver-n'hi més). Suposem que, com a grafs que són, els seus ordres són respectivament n_1 i n_2; es compleix òbviament $n_1 + n_2 \leq n = 5$. Ara bé, aquests components connexos són grafs 2-regulars i, per tant, els ordres respectius han de ser com a mínim 3. En conseqüència $6 = 3+3 \leq n_1+n_2 \leq n = 5$, que és absurd. Això demostra que el graf és connex.

5. Suposem que G és no connex i siguin, per tant, $C_1, \ldots, C_k$, $k \geq 2$ els components connexos de G, amb conjunts de vèrtexs respectius $V_1, \ldots, V_k$, i sigui $q_i = |V_i|$, per a $i = 1, \ldots, k$. Evidentment és $q_i + \ldots + q_k = n$.

Considerem dos vèrtexs u,w que pertanyen a dos components connexos diferents. Suposem, per exemple, que $u \in C_i$, $w \in C_j$, $i \neq j$. Aleshores u pot ser

adjacent com a molt als $q_i - 1$ vèrtexs restants en el component connex al qual pertany, és a dir que $g(u) \leq q_i - 1$, i anàlogament per a w. Tenint en compte això i aplicant la hipòtesi, podem afirmar:

$$\frac{n-1}{2} \leq g(u) \leq q_i - 1, \quad \frac{n-1}{2} \leq g(w) \leq q_j - 1.$$

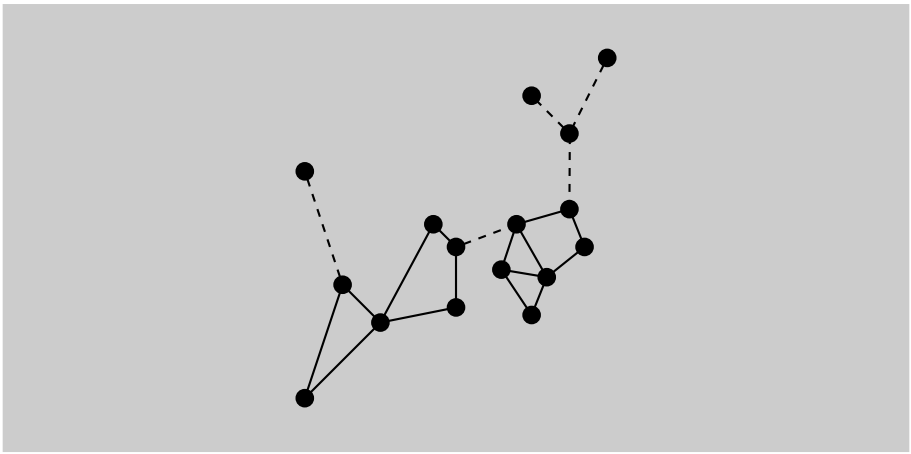
D'aquí podem obtenir immediatament

$$n - 1 = \frac{n-1}{2} + \frac{n-1}{2} \leq g(u) + g(w) \leq q_i + q_j - 2 \leq n - 2,$$

que és una conclusió absurda. Per tant, el graf ha de ser connex.

Pel que fa als grafs amb les seqüències indicades, han de ser connexos en aplicació del resultat. Produïu exemples de grafs amb aquestes seqüències de graus.

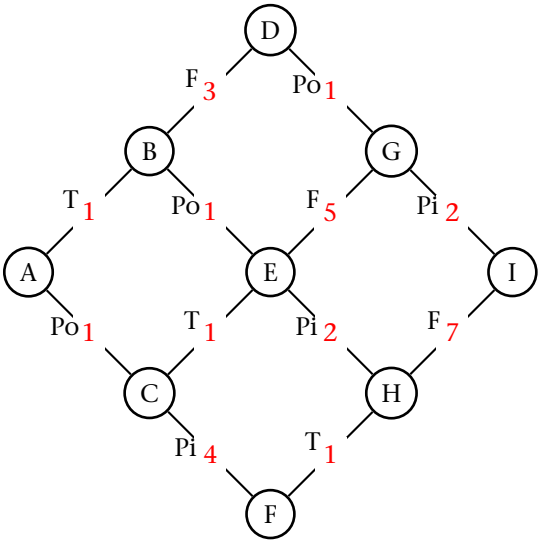
6. En la figura següent indiquem les arestes-pont amb línia discontínua.



7. Suposem que $G = (V,A)$ és connex i sigui $a = \{v,w\}$ una aresta-pont. Considerem el graf auxiliar que resulta de l'eliminació de l'aresta anterior, és a dir, $G' = G - a$.

Si al graf G teníem $g_G(v) = 2k \geq 2$ i $g_G(w) = 2k' \geq 2$, al nou graf G' tindrem $g_{G'}(v) = 2k - 1$ i $g_{G'}(w) = 2k' - 1$, graus senars, i la resta de graus no han variat. Essent a aresta-pont i G graf connex, el graf $G' = G_1 \cup G_2$ és reunió de dos components connexos i els vèrtexs extrems de l'aresta a s'han distribuït un a cadascun d'aquests components connexos. Vegem que això porta cap a una contradicció. En efecte, si per exemple $v \in V(G_1)$, aleshores, considerat com a graf, seria un graf amb un vèrtex de grau senar i la resta de grau parell, amb la qual cosa hi hauria un nombre senar de vèrtexs de grau senar, cosa que és impossible, com a conseqüència del lema de les encaixades.

8. El graf següent mostra les diferents situacions que es donen en el problema:

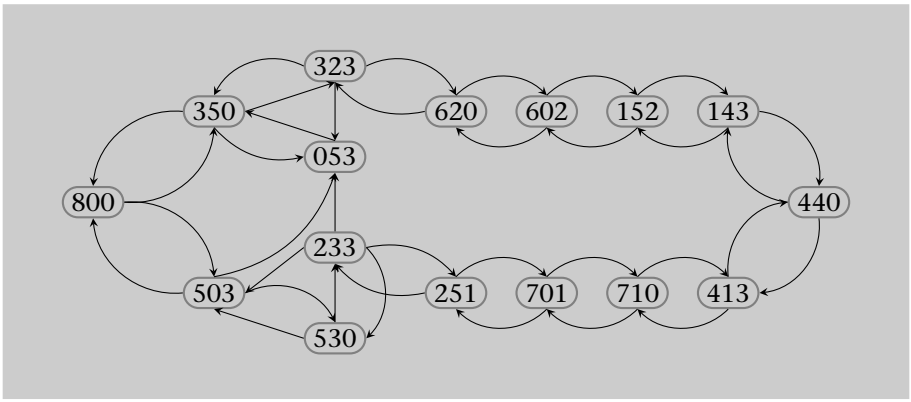


Les arestes representen cadascun dels processos que s'han de fer i el seu cost; els vèrtexs són els estats finals d'aquests processos.

a) Seguint l'algorisme de Dijkstra podeu comprovar de manera senzilla que el recorregut més barat des de A fins a I és ABDGI. El cost serà de 7.

b) El cost actual de foradar després de polir i abans de pintar és 5. El cost total de fer tot el procés d'aquesta manera és 9. Ja que el cost mínim és 7, s'hauria de reduir en 2 unitats el cost de foradar després de polir i abans de pintar per a igualar el recorregut mínim. Per tant, el cost ha de ser 3.

9. El digraf següent representa les diferents transicions entre les garrafes. L'etiqueta de cada vèrtex representa el contingut de cadascuna de la tres garrafes:



Notem que no hem representat tots els estats. Només aquells més rellevants.

Atès que només volem conèixer el nombre mínim de transicions necessàries per a repartir el vi, estem en un problema de cerca del camí mínim en un cas no ponderat. Si apliquem l'algorisme, obtenim que el nombre mínim de passos és 7 i la seqüència de transicions és: 800 → 350 → 323 → 620 → 602 → 152 → 143 → 440.

10. Tant per a calcular el diàmetre com el centre d'un graf G necessitem disposar de les distàncies entre tots els parells de vèrtexs del graf. Per tant, utilitzarem l'algorisme de Floyd per a calcular-los.

```
Entrada : (G,w) graf ponderat d'ordre n.
Sortida : El diàmetre, D, del graf.
algorisme Diàmetre(G)
  inici
    d ← Floyd(G);
    D ← 0
    per i ← 1 fins n
        per j ← 1 fins n
            si d(i,j) > D
              aleshores D ← d(i,j)
            fisi
        fiper
    fiper
    retorn (D)
  fi
```

Per a estudiar l'eficiència d'aquest algorisme, observem que bàsicament s'aplica l'algorisme de Floyd (que ja sabem que té una complexitat $O(n^3)$) i cerquem el valor màxim d'una taula de n^2 valors, que té una complexitat $O(n^2)$. En total tindrà una complexitat $O(n^3)$ comparable a la de l'algorisme de Floyd.

11. Per a calcular el centre del graf necessitem, a més de les distàncies entre tots els parells de vèrtexs, la suma de distàncies de cada vèrtex a la resta:

```
Entrada : (G,w) graf ponderat d'ordre n.
Sortida : El centre, C, del graf.
algorisme Centre(G)
  inici
    d ← Floyd(G);
    C ← 1
    Dmin ← ∑_{j=1}^{n} d(1,j)
    per i ← 2 fins n
        D ← ∑_{j=1}^{n} d(i,j)
        si D < Dmin
          aleshores Dmin ← D
                    C ← i
        fisi
    fiper
  fiper
  retorn (C)
fi
```

Per a calcular l'eficiència observem que calculem les distàncies entre tots els parells de vèrtexs utilitzant l'algorisme de Floyd (amb una complexitat $O(n^3)$) i calculem la suma de totes les files de la matriu d. Això és fa amb una complexitat $O(n^2)$. En total tindrà una complexitat $O(n^3)$ comparable a la de l'algorisme de Floyd.

Bibliografia

Biggs, N. L. (1994). *Matemática Discreta*. (1a. edició, traducció de M. Noy). Barcelona: Ediciones Vicens Vives.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. (2001). *Introduction to Algorithms*. Cambridge: MIT Press.