Projecte de Programació

Grau en Enginyeria Informàtica, UdG

19 d'abril de 2024

Be Water

Be water, my friend
– Bruce Lee

Al meu país, la pluja no sap ploure
O plou poc, o plou massa
Si plou poc és la sequera
Si plou massa és un desastre
Qui portarà la pluja a escola?
Qui li dirà com s'ha de ploure?
– Raimon

1 Presentació

La present sequera ha fet que es desenterressin una sèrie de projectes relacionats amb l'abastament i distribució d'aigua, que dormien en el fons d'un calaix d'una determinada conselleria d'un determinat govern, des que s'havia acabat l'anterior sequera. Entre aquests projectes enterrats, es trobava el desenvolupament de programes informàtics per a la simulació dels efectes que produirien certs canvis en el subministrament d'aigua a nivell agrícola, ramader, industrial o domèstic. Ens referim, per exemple, a estudiar als efectes de mesures d'estalvi com ara la disminució de la pressió o cabal d'aigua en determinats moments i punts de la xarxa, la interconnexió de diferents xarxes de distribució (i l'efecte que podria tenir això en la disminució de cabals), etc. També, en base a l'estudi dels cabals d'aigua, es pretenia poder detectar possibles fuites d'aigua a la xarxa de distribució.

Una spin-off de la UdG ha guanyat un concurs per al desenvolupament d'una part d'aquests programes i nosaltres, com a estudiants en pràctiques, l'ajudarem.[1]

Figura 1: Xarxa de distribució d'aigua en forma esquemàtica.

El nom en clau del projecte serà **Be Water**.

[1] És broma.

2 Descripció de l'entorn de l'aplicació

Passem a descriure l'entorn en el qual ens mourem, de per tal de delimitar les funcions que haurà de tenir l'aplicació.

2.1 Xarxes de distribució d'aigua

2.1.1 Elements d'una xarxa

Simplificant una mica, definirem una xarxa de distribució d'aigua com la unió d'una sèrie de canonades i aixetes de pas. Considerarem que les aixetes poden estar obertes o tancades (però no mig obertes ni mig tancades), de manera que o bé deixen passar tota l'aigua o bé no en deixen passar gens. La Figura 2 mostra una possible aixeta de pas.

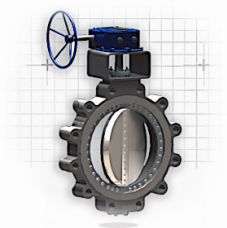


Figura 2: Aixeta de tipus papallona

Les canonades s'empalmen amb determinats elements de connexió, que poden ser en forma de maniguet, en forma de té, en forma de creu, etc., tal com mostra la Figura 3.

Figura 3: Alguns elements de connexió

Les aixetes de pas i les connexions no estan limitades a les que hem descrit, però aquest és un detall que en principi no ens hauria d'importar a l'hora de fer el programa.

Considerarem que cada xarxa de distribució d'aigua pot tenir diversos orígens (punts d'entrada d'aigua) i diversos punts terminals (punts de sortida d'aigua). Cadascun dels punts terminals correspon a un escomesa d'aigua que dóna servei a una finca o edifici, a una indústria, etc., i és on acaba la responsabilitat de la companyia d'aigües. Cada punt terminal tindrà associada una determinada demanda punta d'aigua, expressada en litres per segon, en funció del nombre i tipus d'abonats s'hagi previst connectar-hi. La demanda

real a cada punt terminal serà variable en el temps, entre zero i (suposadament) la demanda punta prevista.

2.1.2 Topologia de les xarxes

Les canonades d'una xarxa d'aigua poden ser de diferents materials, longituds i capacitats (en litres per segon). Les xarxes tindran forma de graf, possiblement amb cicles. Per tal de simplificar, suposarem conegut en quin sentit viatja l'aigua en cada canonada.

Per tal de poder localitzar les aixetes de pas ràpidament amb un GPS, les identificarem amb les seves coordenades geogràfiques. A més, disposarem d'un nom (o àlies) per cada aixeta, per tal de poder-nos-hi referir amb més agilitat. Les coordenades geogràfiques consistiran en una latitud (φ) i una longitud (λ) expressades en graus, minuts i segons. Recordem que la latitud va entre 90 graus nord i 90 graus sud, mentre que la longitud va entre 180 graus est i 180 graus oest. Els graus es divideixen en 60 minuts, i els minuts en 60 segons. Per exemple, la posició "38:53:23N,77:00:32W" (on 'N' indica Nord i 'W' indica oest) correspon a una latitud $\varphi = 38.889722$ i a una longitud $\lambda = -77.008889$. Per tal de tenir més precisió, es poden introduir decimals en els segons.

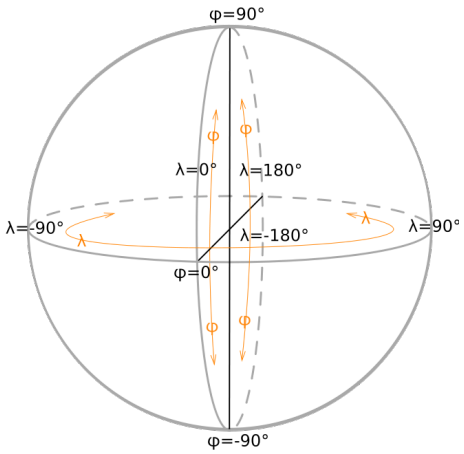


Figura 4: Latitud phi (φ) i Longitud lambda (λ)

En la versió 1.0 de l'aplicació (la que ens toca desenvolupar), suposarem que hi ha una aixeta a cada punt d'origen, una aixeta a cada connexió, i una aixeta a cada punt terminal. A les connexions considerarem que hi ha una sola aixeta independentment de les canonades que s'hi connectin, de manera que quan aquesta aixeta estigui tancada no deixarà passar aigua cap enlloc.

Per tal d'unir dues xarxes, posarem una canonada entre un punt no terminal de la primera i un punt qualsevol de la segona (fins i tot un seu origen). La Figura 5 il·lustra un exemple de connexió de xarxes, on les O indiquen orígens, les C indiquen punts de connexió, i les T indiquen punts terminals.

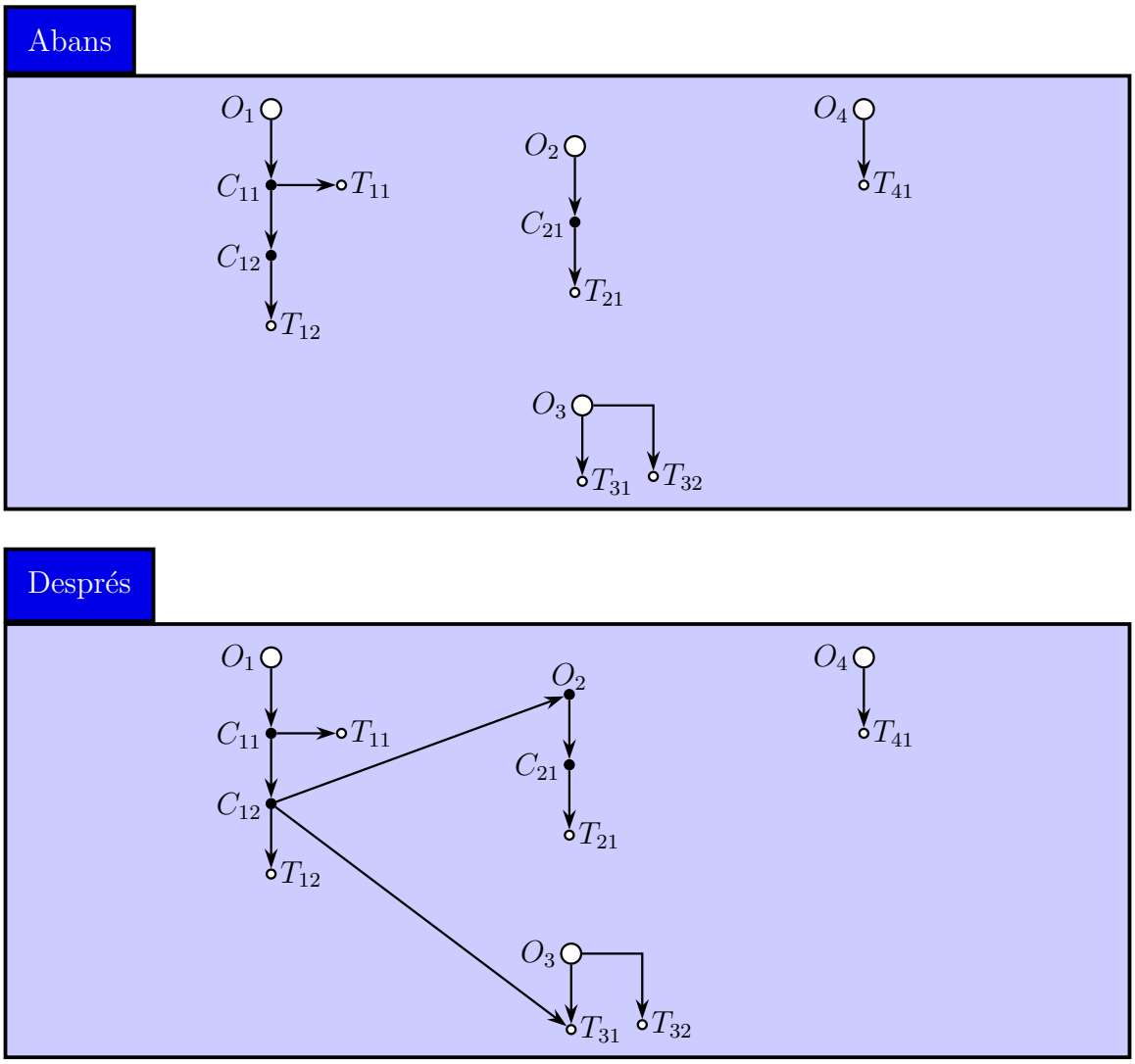


Figura 5: Exemple de connexió de xarxes

Considerarem que si fem arribar aigua a través d'una canonada a un punt d'origen, aquest deixarà de ser origen, i passarà a ser una simple connexió, com és el cas d'O_2 a l'exemple de la Figura 5. Notem també com, en el mateix exemple, després de la connexió obtenim una xarxa que té dos orígens: O_1 i O_3.

2.1.3 Repartiment del cabal

Cada punt d'origen trindrà un determinat cabal potencial, però la quantitat d'aigua que entrarà a la xarxa pels diferents punts d'origen serà variable, en funció de la demanda i de les restriccions que es vulguin imposar per a l'estalvi. Com és lògic, globalment mai no entrarà a la xarxa més aigua de la demanada. Així doncs, a cada moment, la suma dels

cabals en els punt d'origen estarà entre zero i, com a màxim, la demanda del moment en aquell punt.

El cabal es repartirà proporcionalment a la demanda. Per posar un exemple senzill, si el cabal disponible en un punt és de 1000 l/s i l'aigua surt per dues canonades amb demanda de 200 l/s i 3000 l/s respectivament, tenim que la primera representa el 6.25% de la demanda i la segona el 93.75% restant, de manera que a la primera li correspondrà un cabal de 62.5 l/s i a la segona un cabal de 937.5 l/s. D'aquesta manera, en cadascuna d'aquestes dues canonades, un 31.25% de la seva demanda serà satisfeta.

D'altra banda, si en un punt de connexió hi entren diferents canonades, la demanda de sortida (és a dir, la suma de les demandes de les canonades sortints) es repartirà entre les canonades entrants proporcionalment a la capacitat de cadascuna. Veure la secció 4.1 per més detalls respecte al càlcul del cabal a les canonades d'una xarxa.

2.2 Operacions i consultes

A grans trets, les operacions i consultes que està previst efectuar en la versió 1.0 de l'aplicació són les següents (a la secció 3 les acabem de detallar i en donem exemples).

1. Unir xarxes.
2. Associar abonats a punts terminals.
3. Obrir i tancar aixetes.
4. Desfer una sèrie d'operacions d'obrir i tancar aixetes, de manera que tornem a una configuració anterior.
5. Establir un determinat cabal potencial en un punt d'origen.
6. Establir una determinada demanda en un punt terminal.
7. Consultes.
   (a) Trobar cicles en una xarxa.
   (b) Determinar si una xarxa té forma d'arbre.
   (c) Donada una xarxa sense cicles, calcular el cabal mínim que hi hauria d'haver als punts d'origen per tal que cap punt terminal, d'entre aquells on arribi aigua, no rebi menys d'un determinat percentatge de la seva demanda actual.

   Aquesta consulta és important perquè per sota de determinat cabal, la pressió pot ser massa baixa com per què alguns aparells (maquinària industrial, rentaplats, etc.) funcionin bé.
   (d) Donat un subconjunt de les canonades d'una xarxa sense cicles, determinar en quines d'aquestes canonades si es satisfés la demanda actual es sobrepassaria la seva capacitat.

Aquesta consulta és en el sentit contrari que l'anterior: un cabal massa elevat podria representar una pressió massa elevada en algun punt de la xarxa.

(e) Donada una xarxa en forma d'arbre, un conjunt de punts terminals d'aquesta que no reben aigua, un conjunt de punts que sí i un conjunt de punts que no se sap, determinar les aixetes de la xarxa més properes[2] als punts terminals per "sota" de les quals **és segur** que s'ha trencat (o embussat) alguna canonada.

Aquí la idea és que alguns abonats ens han avisat que no reben aigua i això no ens quadra amb la configuració actual de la xarxa. De manera que només pot passar que s'hagi embussat o trencat alguna canonada. Per tant es tracta de determinar quines aixetes cal tancar per tal de no perdre aigua i, en segon lloc, afectar el mínim nombre d'abonats. Després de fer algunes trucades hem confirmat que alguns altres abonats sí que reben aigua, d'altres no ho sabem, i hem de decidir quines aixetes tancar.

(f) Donat un abonat, determinar el cabal que suposadament hauria d'arribar a la seva finca/indústria a partir de la configuració actual de la xarxa a la qual està abonat. Per respondre a aquesta consulta suposarem que la xarxa no té cicles.

(g) Donat un conjunt d'aixetes i una posició geogràfica, llistar les aixetes ordenades segons la seva proximitat a la posició geogràfica donada i, en cas d'empat, alfabèticament.

La motivació d'aquesta consulta és que quan calgui anar a obrir o tancar aixetes, serà convenient minimitzar la distància a recórrer.

8. Dibuixos.

(a) Dibuix d'una xarxa, on es vegi:

i. Els diferents punts de la xarxa (orígens, connexions i punts terminals) identificat cadascun d'ells amb el nom de l'aixeta que hi ha en aquell punt.
ii. L'estat de les aixetes: oberta/tancada.
iii. Les coordenades geogràfiques de cada aixeta.
iv. Les canonades, amb indicació del sentit de l'aigua a cada canonada.
v. La capacitat i el cabal actual a cada canonada.
vi. La demanda punta i la demanda actual a cada punt terminal.

La posició dels diferents punts de la xarxa en el dibuix ha de ser coherent amb les seves coordenades geogràfiques. Per tal de facilitar el càlcul del cabal d'aigua a cada canonada, suposarem que la xarxa no té cicles.

Pel que respecta a dibuixar xarxes, suggerim l'ús de GraphStream (`https://graphstream-project.org/`), que permet el dibuix de grafs dinàmics. A classe de pràctiques en donarem algunes indicacions.

[2]Ens referim a la proximitat dins de la xarxa (és a dir, seguint les canonades), no a la proximitat geogràfica.

(b) Dibuix resultant de calcular el flux màxim d'una xarxa. En aquest cas la xarxa pot tenir cicles. El problema del flux màxim (*maximum flow problem*) consisteix en determinar la quantitat màxima de flux que pot passar per una xarxa, amb un sol origen (font) i un sol punt terminal (pou). En el cas de tenir més d'una font i més d'un pou (*multi-source multi-sink maximum flow problem*), el problema es pot transformar fàcilment al cas d'una sola font i un sol pou. A la Wikipedia `http://en.wikipedia.org/wiki/Maximum_flow_problem` hi trobareu referències a diferents algorismes que permeten resoldre aquest problema. A classe de pràctiques n'explicarem algun.

Pel que respecta al dibuix, a diferència de l'anterior, volem que sigui senzill com el de la Figura 6. És a dir, en aquest dibuix no hi han d'aparèixer els noms de les aixetes, les coordenades, ...

Aquesta consulta/dibuix ve motivada per la possibilitat d'utilitzar la nostra aplicació no només per a l'estudi de les xarxes de distribució d'aigua potable, sinó també per estudiar la capacitat de les xarxes d'aigües pluvials. De fet, el problema del flux màxim té múltiples aplicacions. A part de l'esmentat problema del flux màxim d'aigua a través d'una xarxa de clavegueram, podríem pensar en calcular el flux màxim de peces que poden circular per una cadena de muntatge, el flux màxim de vehicles en una xarxa de carreteres, etc.

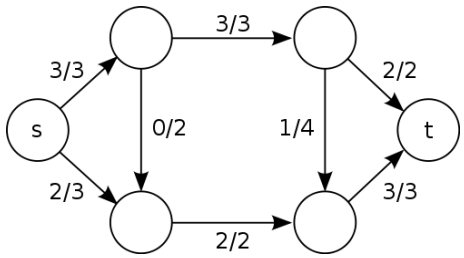


Figura 6: Exemple d'una xarxa de flux amb flux màxim. La font és s i el pou és t. Els números denoten el flux i la capacitat.

3 Descripció de l'entrada i la sortida

A continuació, mitjançant exemples, detallem el format de l'entrada i la sortida i acabem de precisar alguns detalls de les operacions. Les dades d'entrada i les consultes apareixeran, intercalades, en un fitxer de text. Les respostes a les consultes s'escriuran en un fitxer de text. Els dibuixos es visualitzaran a la pantalla.

3.1 Construcció de les xarxes de distribució

L'entrada consistirà en una seqüència d'operacions com la que es presenta a continuació. Començarà amb la construcció d'una sèrie de xarxes de distribució d'aigua, i continuarà amb una sèrie de consultes i operacions de modificació de les xarxes. No hi haurà cap marca de fi.

Les coordenades geogràfiques seran de la forma "41:57:47.29N,2:49:53.64E". La part de l'esquerra de la coma denota la latitud, i la part de la dreta la longitud. El primer valor són els graus, el segon valor són els minuts, i el tercer valor són els segons. La lletra que hi ha a continuació indica la direcció (N per al nord, S per al Sud, E per a l'est i W per a l'oest). La latitud està entre 90 graus nord i 90 graus sud. La longitud està entre 180 graus est i 180 graus oest. Cada grau es divideix en 60 minuts, i cada minut en 60 segons. Per tal de tenir més precisió, els segons s'acostumen a representar com un número real.

```
terminal                ← Donem d'alta un punt terminal
T11                     ← nom
41:57:47.29N,2:49:53.64E ← coordenades
23977.5                 ← demanda punta en l/s
terminal                ← Donem d'alta un punt terminal
T12                     ← nom
41:58:24.45N,2:48:52.3E ← coordenades
3456.23                 ← demanda punta en l/s
connexio                ← Donem d'alta una connexió
C11                     ← nom
41:57:47.34N,2:49:53.85E ← coordenades
origen                  ← Donem d'alta un origen
O1                      ← nom
41:53:7.56N,2:33:14.32E ← coordenades
connectar               ← Connectem
C11                     ← el punt C11
T11                     ← amb el punt T11
270.45                  ← mitjançant una canonada de capacitat 270.45 l/s
connectar               ← Connectem
C11                     ← el punt C11
T12                     ← amb el punt T12
20000                   ← mitjançant una canonada de capacitat 20000 l/s
connectar               ← Connectem
O1                      ← el punt O1
C11                     ← amb el punt C11
5400.5                  ← mitjançant una canonada de capacitat 5400.5 l/s
```

La seqüència d'operacions del fitxer d'entrada pot estar en qualsevol ordre. Hi poden aparèixer consultes i modificacions intercalades. L'única restricció és que si una operació fa referència a un identificador (nom), aquest ha d'haver estat definit abans. Si això no és

així, o hi ha algun error de format, cal llançar una excepció. No hi ha cap restricció sobre com poden ser els identificadors. L'ordre en què enumerem els punts connectats determina el sentit de l'aigua entre aquests, i només hi pot haver una canonada entre dos punts. De tota manera, cal assegurar que el sentit de l'aigua sigui coherent: l'aigua ha d'arribar als punts terminals i, si en un punt hi arriba aigua, cal que hi hagi camí des de com a mínim un punt d'origen. Noteu que això no evita l'existència de cicles.

Una seqüència d'instruccions com l'anterior hauria donat lloc a una xarxa com la que es mostra esquemàticament a la Figura 7 (aquest esquema no reflecteix la distribució geogràfica real). Recordem que suposarem que tenim aixetes tant a l'origen com als punts de connexió i als punts terminals (veure secció 2.1.2). Aquestes aixetes, per defecte, estaran totes obertes.

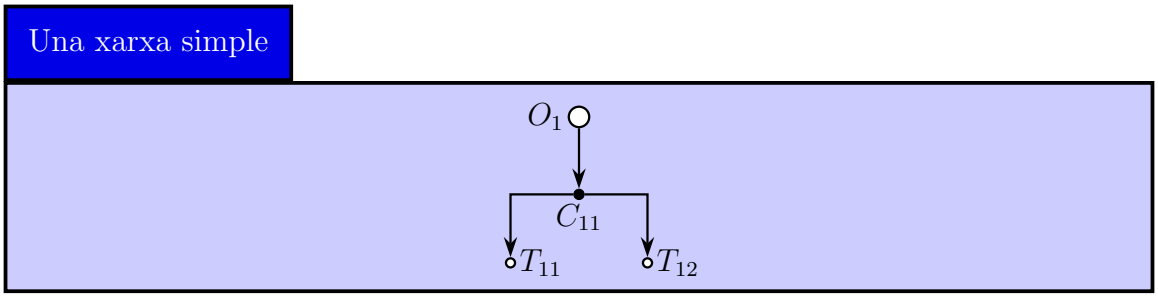


Figura 7: Xarxa amb dos únics punts terminals

Podem definir tantes xarxes com vulguem. Per exemple, per tenir una situació com la que mostra la Figura 8, només caldria afegir:

`terminal`	← **Donem d'alta un punt terminal**
`T21`	← nom
`42:23:21N,2:55:13.4E`	← coordenades
`1300`	← demanda punta en l/s
`origen`	← **Donem d'alta un origen**
`O2`	← nom
`42:25:57.61N,2:54:14.24E`	← coordenades
`connectar`	← **Connectem**
`O2`	← el punt `O2`
`T21`	← amb el punt `T21`
`1500`	← mitjançant una canonada de capacitat 1500 l/s

Tal com hem explicat a la secció 2.1.2, la unió de dues xarxes es faria simplement connectant un punt no terminal de la primera amb un punt qualsevol de la segona (fins i tot un origen):

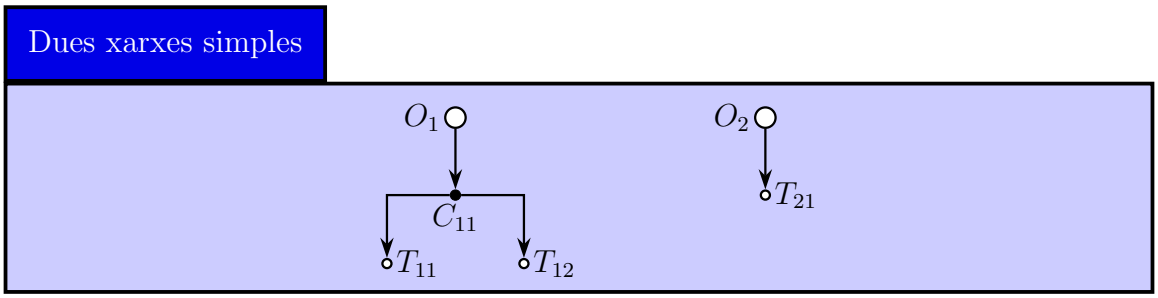


Figura 8: Dues xarxes independents

```
connectar   ← Connectem
O1          ← el punt O1
O2          ← amb el punt O2
1000        ← mitjançant una canonada de capacitat 1000 l/s
```

Així obrindríem una situació com la que mostra la Figura 9. Notem que l'origen de la segona xarxa ha passat a ser una connexió, per bé que manté el seu nom O_2.

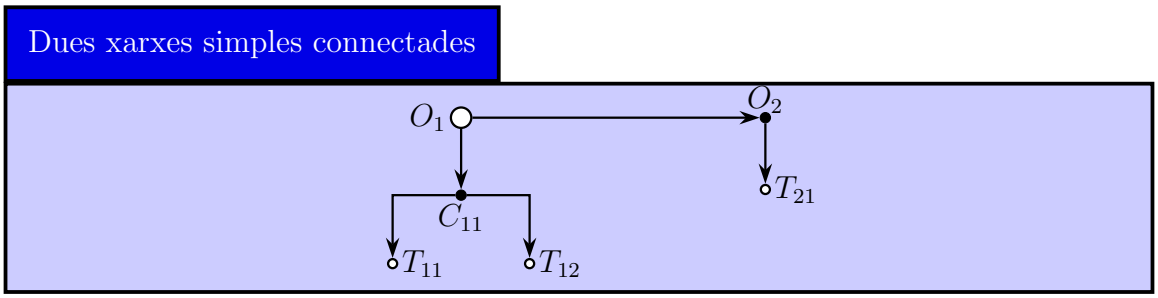


Figura 9: Dues xarxes connectades

3.2 Abonar clients a punts terminals

Cada punt terminal pot tenir associats diversos abonats, que identificarem amb un `String` (possiblement, el DNI). Suposarem que la resta de dades dels abonats rau en un altre mòdul que no ens pertoca desenvolupar a nosaltres, o bé estan guardades en una base de dades. L'assignació d'abonats és com segueix:

```
abonar      ← Abonar
77324554Z   ← el client 77324554Z
T21         ← al punt terminal T21
abonar      ← Abonar
45678443R   ← el client 45678443R
T21         ← al punt terminal T21
```

3.3 Obrir i tancar aixetes

Recordeu que, si no es diu el contrari, les aixetes es consideren obertes, i que tenim aixetes tant als punts d'origen de les xarxes com a totes les seves connexions i a tots els seus punts terminals. Per obrir i tancar aixetes, el format és:

```
tancar  ← Tancar
C11     ← l'aixeta C11
obrir   ← Obrir
O2      ← l'aixeta O2
```

Seguint l'exemple (Figura 9), acabem de deixar sense aigua els punts terminals T_{11} i T_{12}. D'altra banda, l'operació d'obrir l'aixeta del punt O_2 no té cap efecte, perquè aquesta aixeta ja estava oberta.

3.4 Tornar enrere

Aquesta operació es farà normalment després d'haver fet una sèrie de modificacions experimentals d'obertura i tancament d'aixetes. La seva sintaxi és:

```
backtrack  ← Reculem
n          ← n operacions d'obrir/tancar aixetes
```

Desfarà les `n` darreres operacions d'obrir i tancar aixetes. En l'exemple que ens ocupa, podríem fer `backtrack 1` o `backtrack 2`, perquè només hem tocat dues aixetes fins ara (veure apartat anterior). L'operació `backtrack 1` implicaria tornar a l'estat anterior a l'última operació d'obrir/tancar aixetes. Com que l'última operació havia estat obrir l'aixeta O_2, però aquesta ja estava oberta, l'estat no canviarà. L'operació `backtrack 2` deixaria les aixetes tal com estaven al principi (totes obertes).

3.5 Establir cabal a l'origen

Per defecte, el cabal d'aigua en els punts d'origen de les xarxes és zero. Per establir un determinat cabal potencial en un punt d'origen farem:

```
cabal  ← Cabal potencial
O1     ← al punt O1
25000  ← de 25000 l/s
```

Noteu que en la situació de la Figura 9 seria un error fixar el cabal del punt O_2 perquè, tot i que conserva el nom, ha deixat de ser un origen en el moment que hem unit les dues xarxes.

3.6 Establir demanda actual

Per establir la demanda d'aigua actual en un punt terminal farem:

```
demanda  ← Demanda actual
T11      ← al punt T11
5000     ← de 5000 l/s
```

Es pot establir demanda d'aigua a qualsevol punt terminal, encara que no n'hi arribi per culpa d'alguna aixeta tancada. Per defecte, la demanda a tots els punts terminals on no s'ha establert aquesta és 0. La demanda màxima que es pot establir és la especificada com a demanda punta en aquell punt (veure apartat 3.1). No pot establir-se demanda d'aigua a punts de la xarxa que no siguin terminals.

3.7 Consultes

A continuació expliquem les diferents consultes possibles a partir de l'exemple més complet que hem vist a la secció 2.1.2, i que tornem a reproduir aquí a la Figura 10. Disposem de dues xarxes independents, la primera amb dos orígens (O_1 i O_3) i la segona amb un (O_4).

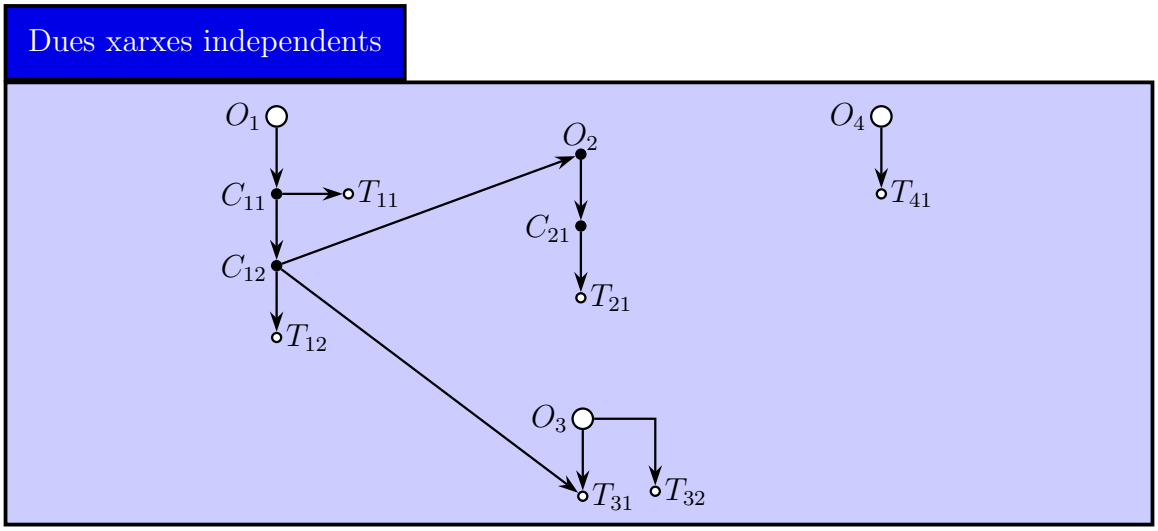


Figura 10: Estat de les xarxes després d'algunes connexions

3.7.1 Trobar cicles en una xarxa

Identificarem les xarxes mitjançant qualsevol dels seus punts d'origen. Per preguntar si una xarxa té cicles, escriurem:

```
cicles  ← Preguntem si hi ha cicles
O1      ← a la xarxa on pertany O1
```

La resposta (que s'escriurà en el fitxer de sortida) serà amb el format `O1 no te cicles` o bé `O1 te cicles` (en l'exemple que ens ocupa seria `O1 no te cicles`), sense accents. Noteu que podríem haver identificat la mateixa xarxa indistintament amb l'origen `O3`.

3.7.2 Determinar si una xarxa té forma d'arbre

Ens interessa poder saber si una xarxa té forma d'arbre. El format de la consulta seria com segueix:

`arbre`	← **Preguntem si té forma d'arbre**
`O1`	← la xarxa on pertany `O1`

La resposta serà amb el format `O1 es un arbre` o bé `O1 no es un arbre` (en l'exemple que ens ocupa seria `O1 no es un arbre`, perquè al node T_{31} hi arriben dues canonades).

3.7.3 Cabal mínim als punts d'origen

Donada una xarxa sense cicles, calcular el cabal mínim que hi hauria d'haver als punts d'origen per tal que cap punt terminal, d'entre aquells on arribi aigua, no rebi menys d'un determinat percentatge de la seva demanda actual. Com abans, permetrem identificar una xarxa mitjançant qualsevol dels seus punts d'origen. La sintaxi seria:

`cabal minim`	← **Cabal mínim als punts d'origen**
`O3`	← de la xarxa on pertany `O3`
`50%`	← per tal de satisfer la demanda en un 50% com a mínim

La resposta seria exactament com segueix (sense accents, i amb salt de línia al final de cada línia):

```
cabal minim
55
```

Recordem que només cal satisfer la demanda dels llocs on arriba l'aigua. En l'exemple que ens ocupa (Figura 10), suposarem que l'aixeta O_3 està tancada i les altres estan obertes, i tenim una demanda de 30 l/s en el punt T_{11}, 50 l/s en el punt T_{12}, 20 l/s en el punt T_{21}, 10 l/s en el punt T_{31} i 10 l/s en el punt T_{32}. Tal com hem explicat a la secció 2.1.3, l'aigua es reparteix per les diferents canonades, proporcionalment a la demanda. Per satisfer un 50% de la demanda, cal que arribin la meitat dels litres demanats a cadascun dels punts terminals que hem enumerat (exceptuant el punt T_{32}, on no arriba aigua perquè l'aixeta O_3 està tancada), això és, 15 l/s al punt T_{11}, 25 l/s al punt T_{12}, 10 l/s al punt T_{21} i 5 l/s al punt T_{31}. Per tant, faria falta un cabal de 55 l/s en conjunt als orígens (en aquest cas, només O_1).

Noteu que en aquesta consulta no hem de tenir en compte per a res la capacitat de les canonades.

3.7.4 Excés de cabal

Donat un subconjunt de les canonades d'una xarxa sense cicles, determinar en quines d'aquestes canonades si es satisfés la demanda actual es sobrepassaria la seva capacitat.

```
exces cabal   ← Preguntem si hi ha excés de cabal
C12-O2        ← a la canonada C12-O2 i
O3-T32        ← a la canonada O3-T32
```

La llista de canonades de la consulta s'acaba amb una nova operació o bé amb el final del fitxer.

Propagarem la demanda des dels punts terminals cap als punts d'origen a través de les canonades. En el cas que puguem propagar la demanda per diferents canonades, ho farem proporcionalment a la seva capacitat. A diferència del cas normal (apartat 4.1), aquí propagarem la demanda encara que es sobrepassi la capacitat de les canonades. A més, si a causa d'aixetes tancades no podem rebre aigua per alguna canonada, repartirem la demanda entre les canonades que sí que ens poden enviar aigua (noteu que això també ho fem diferent que en el cas normal).

Un cop coneguem la demanda en els punts d'origen, intentarem satisfer-la amb el cabal disponible, detectant en quines canonades s'excediria la capacitat si es satisfés la demanda.

Suposem la xarxa de la Figura 10, amb una demanda de 500 l/s a cadascun dels punts terminals, l'aixeta O_3 tancada, i un cabal de 1000 l/s a O_1. Degut al fet que O_3 està tancada, els 500 l/s de demanda del punt T_{31} es propagaran tots cap al punt C_{12}. En definitiva, al punt O_1 li arribarà una demanda de 2000 l/s, que és la suma dels 500 l/s de cadascun dels punts terminals T_{11}, T_{12}, T_{21} i T_{31}.

Ara intentarem repartir els 1000 l/s disponibles a O_1 proporcionalment a la demanda de cada canonada (encara que s'excedeixi la seva capacitat). En particular, intentaríem fer arribar 250 l/s a T_{21} (la quarta part). Llavors, assumint que la capacitat de la canonada $C_{12} - O_2$ fos de 200 l/s, escriurem en el fitxer de sortida:

```
exces cabal
C12-O2
```

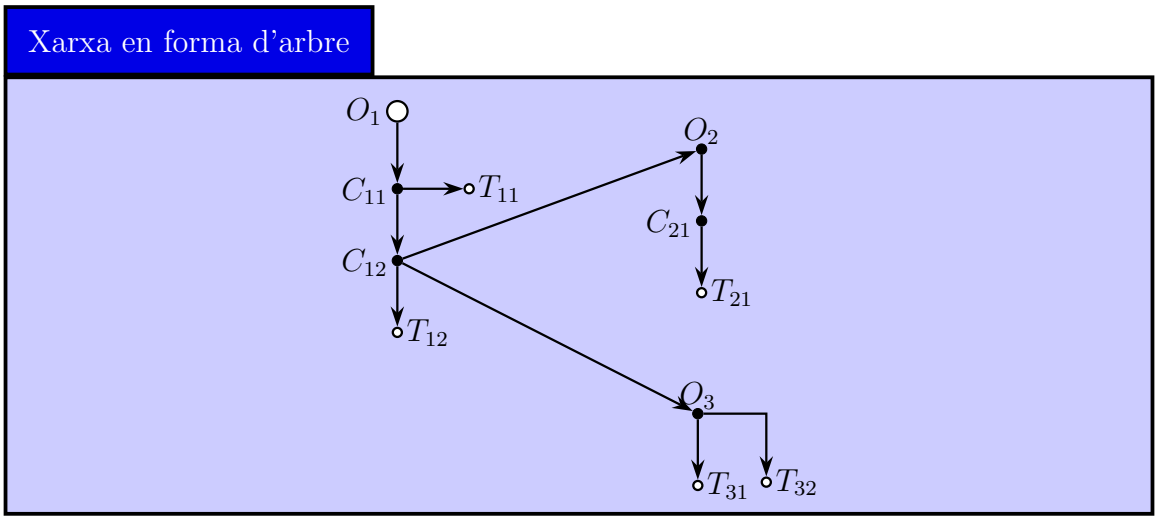
Remarquem que cal llistar les canonades on es superi la seva capacitat en el mateix ordre que apareixen a la consulta. En el cas que no es superi la capacitat de cap canonada, escriurem igualment la línia `exces cabal`, i deixarem buit el llistat de canonades a la resposta.

3.7.5 Aixetes a tancar

Donada una xarxa en forma d'arbre, un conjunt de punts terminals d'aquesta que no reben aigua, un conjunt de punts que sí i un conjunt de punts que no se sap, determinar les aixetes

de la xarxa més properes[3] als punts terminals per "sota" de les quals **és segur** que s'ha trencat (o embussat) alguna canonada.

Tingueu en compte que aquesta consulta només pot aplicar-se a xarxes en forma d'arbre, és a dir, assumirem com a precondició que els punts terminals considerats pertanyen a una xarxa en forma d'arbre. Per il·lustrar el funcionament d'aquesta consulta modifiquem l'exemple considerat per tal que la xarxa principal tingui forma d'arbre.



El format de l'entrada serà:

```
situacio   ← Situació basada en dades reals
T12 NO     ← T12 no rep aigua
T21 NO     ← T21 no rep aigua
T31 SI     ← T31 rep aigua
T11 NO     ← T11 no rep aigua
```

Notem que els punts terminals dels quals ens informen pertanyen tots a una mateixa xarxa, i que aquesta té forma d'arbre. Dels punts que no ens informen, no podrem assumir res, excepte pels que sapiguem segur que no hi arriba aigua degut a què alguna aixeta està tancada.

En l'exemple donat, anem a suposar que l'aixeta T_{11} estigui tancada, i totes les altres obertes en el moment de rebre la informació. Aleshores, el fet que al punt T_{11} no hi hagi aigua no ens aporta cap informació. Tenint en compte la resta d'informacions, podem estar segurs que hi ha un problema a la canonada entre C_{12} i T_{12}, perquè T_{12} no rep aigua i en canvi T_{31} sí. A més, ha d'haver-hi algun problema en alguna canonada entre C_{12} i T_{21}, perquè T_{21} no rep aigua. L'aixeta més propera a T_{21} per sota de la qual és segur que està el problema, és la C_{12} novament. Per tant, la resposta hauria de ser:

[3]Ens referim a la proximitat dins de la xarxa (és a dir, seguint les canonades), no a la proximitat geogràfica.

```
tancar
C12
```

Podria donar-se el cas que calgués tancar més d'una aixeta. En tal cas, les llistaríem en ordre alfabètic. Tanmateix, les aixetes que calgui tancar i que quedin per sota (és a dir, a continuació) d'una altra aixeta que calgui tancar, no les llistarem.

3.7.6 Cabal en un punt terminal

Donat un abonat, determinar el cabal que suposadament hauria d'arribar a la seva finca/indústria a partir de la configuració actual de la xarxa a la qual està abonat. L'entrada serà:

```
cabal abonat   ← Cabal que arriba al punt terminal on està abonat
43768332Q      ← el client 43768332Q
```

La resposta tindrà el format:

```
cabal abonat
345.65
```

on el valor llistat és la quantitat d'aigua en l/s que teòricament hauria d'arribar en aquest moment al punt terminal al qual està associat el client. Suposarem que la xarxa no té cicles.

3.7.7 Llistar segons proximitat geogràfica

Donada una sèrie d'aixetes i una posició geogràfica, llistar les aixetes ordenades segons la seva proximitat a la posició geogràfica donada i, en cas d'empat, alfabèticament. L'entrada seria com segueix:

```
proximitat                ← Llistar ordenadament per proximitat
41:57:47.29N,2:49:53.64E  ← al punt 41:57:47.29N,2:49:53.64E
C12                       ← l'aixeta C11
O4                        ← l'aixeta O4
T11                       ← l'aixeta T11
```

La sèrie d'aixetes s'acaba amb una nova operació o bé amb el final de fitxer. La resposta seria, p.ex.,

```
proximitat
O4
C12
T11
```

si el punt O_4 fos més proper a la posició 41:57:47.29N,2:49:53.64E que el punt C_{12}, i aquest hi fos més proper que el punt T_{11}. En cas d'empat, els llistaríem en ordre alfabètic.

Recordem que tots els punts d'una xarxa (orígens, connexions i punts terminals) tenen associada una posició geogràfica. El càlcul de la distància es farà segons la fórmula de Haversine (que assumeix una terra esfèrica, i no té en compte les muntanyes, però és molt acurada):

$$
\begin{aligned}
R &= \text{radi de la terra } (6.371km) \\
\Delta lat &= lat_2 - lat_1 \\
\Delta long &= long_2 - long_1 \\
a &= \sin^2(\Delta lat/2) + \cos(lat_1) \times \cos(lat_2) \times \sin^2(\Delta long/2) \\
c &= 2 \times atan2(\sqrt{a}, \sqrt{1-a}) \\
d &= R \times c
\end{aligned}
$$

Aquesta fórmula ens dóna la distància d entre un punt de latitud lat_1 i longitud $long_1$, i un punt de latitud lat_2 i longitud $long_2$, totes expressades en radians. Per això, primer haurem de passar les coordenades geogràfiques a graus decimals. El nombre de graus decimals és el nombre de graus, més els minuts dividits per 60, més els segons dividits per 3600. Després, passarem els graus decimals a radians multiplicant els graus decimals per $\pi/180$.

Per exemple, la posició "38:53:23N,77:00:32W" correspon a una latitud de 38.889722 graus decimals i a una longitud de -77.008889 graus decimals. Recordem que les posicions amb latitud sud o longitud oest corresponen a graus negatius. Passats a radians, tindríem una latitud de 0.678753698 i una longitud de -1.344058665. Aquests serien els valors de, p.ex., lat_1 i $long_1$.

Nota: la funció $atan2(y, x)$, disponible a la classe `Math` de Java, retorna l'angle θ de la conversió de les coordenades cartesianes (x, y) a coordenades polars (r, θ).

3.7.8 Dibuix d'una xarxa

Fer un dibuix d'una xarxa tal com s'ha especificat a l'apartat 8a. Com sempre, permetrem identificar una xarxa mitjançant qualsevol dels seus punts d'origen. La sintaxi seria:

```
dibuix  ← Visualitzar un dibuix
O3      ← de la xarxa on pertany O3
```

El dibuix es visualitzarà a la pantalla.

3.7.9 Dibuix resultant de calcular el flux màxim d'una xarxa

Calcular el flux màxim d'una xarxa i fer un dibuix del resultat tal com s'ha especificat a l'apartat 8b. La sintaxi seria:

```
max-flow  ← Calcular i visualitzar flux màxim
O3        ← de la xarxa on pertany O3
```

El dibuix es visualitzarà a la pantalla.

4 Aclariments

Donem aquí una sèrie d'aclariments que s'han extret del text principal per no entorpir.

4.1 Càlcul del cabal a cada canonada

Per tal de calcular el cabal actual a cada canonada d'una xarxa sense cicles, seguirem el següent procediment (que és una simplificació de la realitat).

1. Raonarem en primer lloc de baix cap a dalt (és a dir, des dels punts terminals cap als punt d'origen). Recursivament, a cada node[4] de la xarxa tindrem una demanda, que és la suma de les demandes de les canonades que hi ha per sota d'aquest node (en el cas dels nodes terminals, seria la demanda actual). Si en un node l'aixeta està tancada, entendrem que la demanda en aquest punt és zero. Per tal de propagar la demanda cap amunt, si la suma de les capacitats de les canonades que entren en un punt és insuficient per satisfer aquesta demanda, aleshores propagarem cap amunt per cadascuna d'elles una demanda igual a la seva capacitat. Si, pel contrari, la suma de les capacitats de les canonades és més que suficient, repartirem la demanda proporcionalment a les capacitats de les canonades. Compte: la propagació cap amunt de la demanda la farem independentment del fet que pugui satisfer-se o no (per culpa de cabal insuficient o aixetes tancades més amunt).

2. Amb això haurem arribat a saber quina demanda tenim a cada punt d'origen. Ara caldrà fer un recorregut cap avall per repartir el cabal disponible. Si el cabal disponible en un punt és més que suficient per satisfer la demanda de totes les canonades que hi ha per sota, aleshores es subministrarà el cabal que es demana a cadascuna d'elles (notar que, d'acord amb el recorregut ascendent que hem descrit abans, el cabal sol·licitat per cada canonada serà com a màxim el corresponent a la seva capacitat). Si el cabal disponible és insuficient, aleshores es distribuirà proporcionalment a la demanda.

   Nota: No és cap error tenir cabal sobrant (superior a la demanda). La idea és que estem parlant d'un cabal potencial, i mai no injectem a la xarxa més aigua que la que surt. De la mateixa manera, no es cap error que hi hagi més demanda que oferta.

A continuació donem un exemple. La Figura 11 mostra les demandes calculades en el recorregut ascendent, i la Figura 12 mostra els cabals calculats en el recorregut descendent (arrodonits a dos decimals).

Una creu indica una aixeta tancada. Les expressions de la forma n/m s'han d'interpretar de la següent manera. A la Figura 11:

[4]Entenem per node qualsevol punt de la xarxa on hi ha una aixeta: punt terminal, connexió o origen.

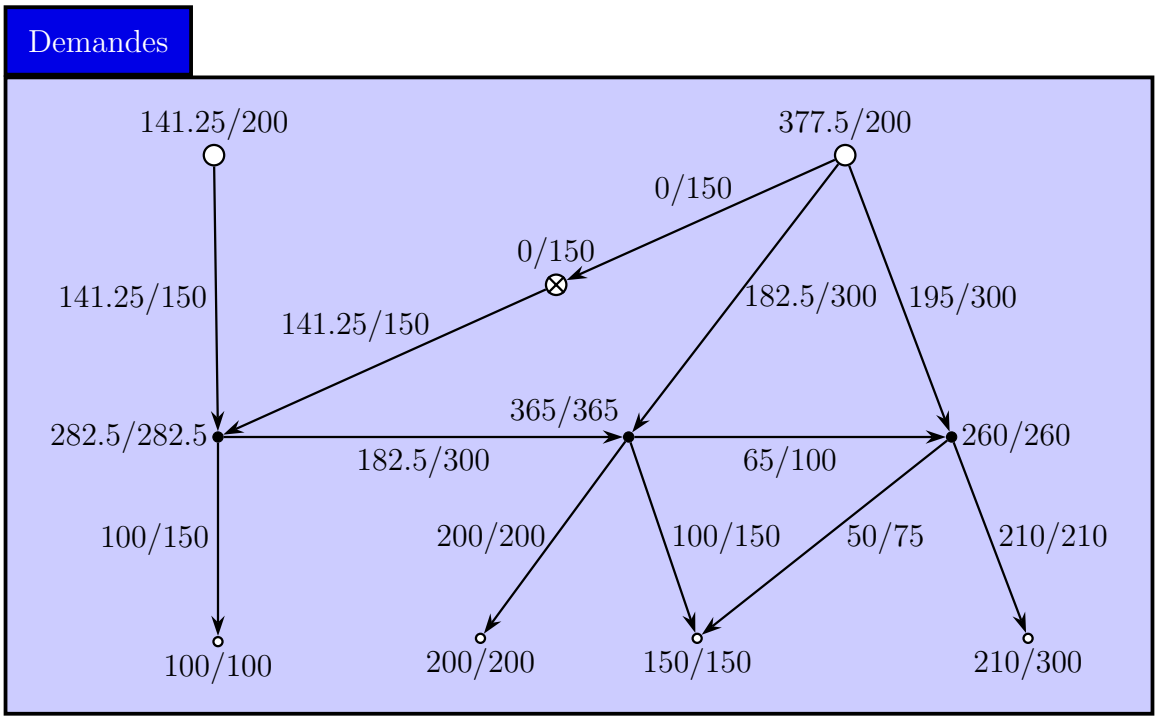


Figura 11: Determinació de demandes mitjançant recorregut ascendent.

- En els punts d'origen: n = demanda, m = cabal potencial.
- A les canonades: n = demanda propagada, m = capacitat.
- A les connexions i punts terminals: n = demanda propagada, m = demanda.

A la Figura 12:

- En els punts d'origen: n = cabal real, m = cabal potencial.
- A les canonades: n = cabal real, m = capacitat.
- A les connexions i punts terminals: n = cabal real, m = demanda.

5 Fitxers JAR i variant per a equips de tres

Aquest projecte està pensat inicialment per a equips de dos membres. Els equips de tres hi hauran d'afegir una interfície gràfica Swing o JavaFX completa, que permeti fer totes les operacions de manera interactiva. La implementació d'una interfície gràfica no està prohibida per als equips de dos, però en cap cas és necessària. El programa s'ha de poder utilitzar en mode text via terminal, de la següent manera:

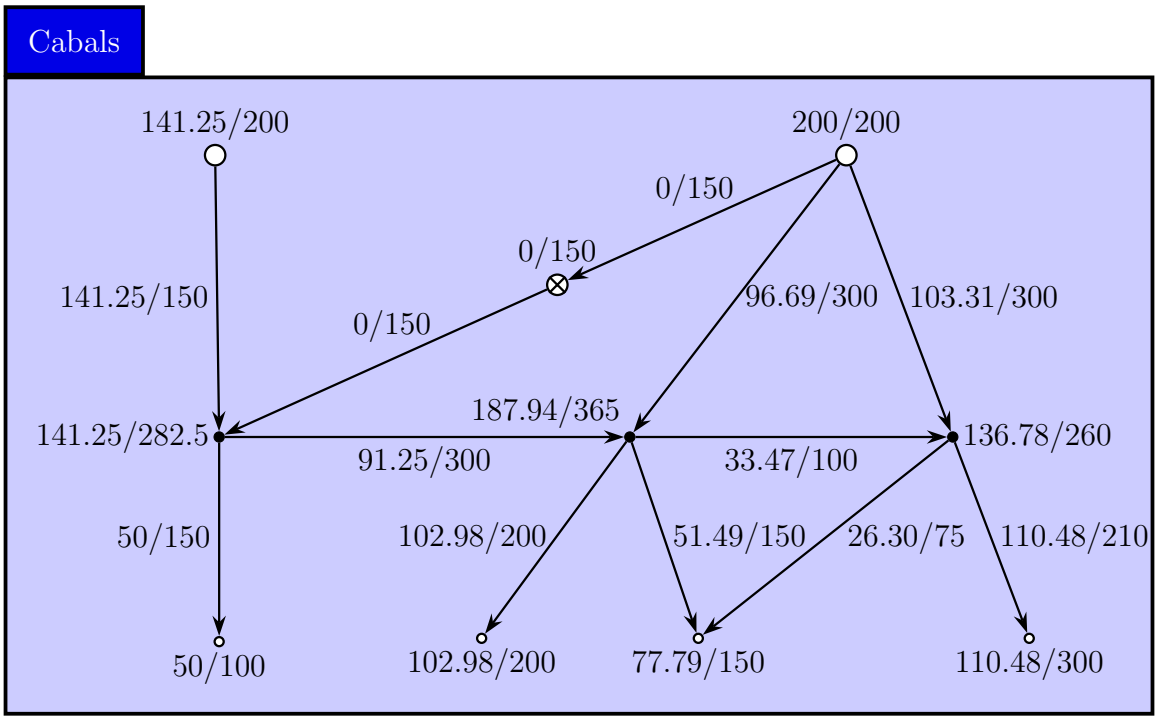


Figura 12: Determinació de cabals mitjançant recorregut descendent.

```
java -jar BeWater.jar fitxer_entrada fitxer_sortida
```

Per a la versió gràfica farem:

```
java -jar BeWaterGUI.jar
```

En la versió gràfica, els noms dels fitxers es preguntaran a l'usuari, permetent també fer operacions interactives. Els equips de tres hauran d'entregar les dues versions: `BeWater.jar` i `BeWaterGUI.jar`.

Si hem utilitzat GraphStream, o qualsevol altra biblioteca, la inclourem a la carpeta `lib` del projecte, i en el fitxer de manifest del jar indicarem el seu camí (camí relatiu, sisplau, de manera que es pugui executar en qualsevol màquina; veure apunts sobre fitxers jar al Moodle). En canvi, no inclourem les biblioteques dependents del Sistema Operatiu com JavaFX. En cas de necessitar JavaFX, executaríem el programa de la següent manera:

```
java --module-path cami_a_javafx/lib --add-modules javafx.controls
  -jar BeWater.jar fitxer_entrada fitxer_sortida
```

o bé

```
java --module-path cami_a_javafx/lib --add-modules javafx.controls
  -jar BeWaterGUI.jar
```

(afegint el mòduls de JavaFX necessaris a la crida)

6 Normes bàsiques

- L'aplicació s'ha de desenvolupar en Java (versió ≥ 8).
- S'han de seguir tots els principis de disseny explicats a classe de teoria.
- S'han de respectar les normes bàsiques per al codi font (document al Moodle).
- S'ha de respectar el codi d'honor (document al Moodle).
- El resultat final ha de ser:
  - Un programa fiable i eficient, i al mateix temps raonablement fàcil d'entendre, modificar i mantenir.
  - Una documentació clara i útil.
- Un i només un membre de cada equip serà responsable de programar i provar cada classe (s'indicarà el nom de l'autor en forma de comentari al principi).
- Per raons intrínseques de l'assignatura, competències a assolir i demés, no està permesa la realització del projecte de manera individual.
- Si durant el desenvolupament del projecte un alumne abandona, caldrà comunicar-ho al professor de pràctiques tan aviat com sigui possible, per tal de reorganitzar els equips de la manera que es consideri oportuna.
- L'assistència a classes de pràctiques serà obligatòria durant el desenvolupament del projecte, per tal de fer-ne el seguiment.
- No s'acceptaran projectes del quals no se n'hagi pogut fer un seguiment satisfactori.
- Caldrà pujar setmanalment les novetats a un repositori de GitHub, encara que siguin poques o es tracti de canvis provisionals. No us oblideu que una part important de la nota tindrà a veure amb el seguiment.
- A l'inici del projecte, cada membre de l'equip obrirà un *issue* (assumpte) dins del repositori de GitHub, on anirà explicant els seus avenços setmanals i les seves propostes de treball de cara a la setmana següent. Aquest *issue* restarà obert fins al final del projecte, i el professor hi farà els comentaris que cregui convenients. Remarquem que ha d'haver-hi un *issue* de seguiment per cada alumne (poseu-hi el vostre nom en el títol). Aquests *issues* de seguiment s'hauran d'actualitzar abans de cada sessió de pràctiques.
- Si teniu preguntes particulars, obriu *issues* a part del destinat al seguiment setmanal. Demanem prioritzar els *issues* de GitHub davant del correu electrònic.
- Si voleu rebre còpia per correu electrònic dels comentaris en els *issues* de GitHub, cal que us poseu com a *watchers* del vostre propi repositori.

- Les preguntes i respostes d'interès general s'han fer al fòrum del Moodle, especialment si creieu que se'n poden beneficiar altres estudiants.

7 Lliuraments

7.1 Lliurament inicial

El lliurament inicial consistirà en un disseny preliminar dels mòduls i operacions necessàries.

7.1.1 Contingut

1. Proposta inicial de divisió en mòduls, en forma d'interfícies o classes Java. Cal donar una descipció general de cada mòdul i la capçalera de les seves operacions públiques (només les capçaleres, deixant la seva implementació per més endavant) junt amb la seva especificació pre-post.

   Heu de proposar tots els mòduls i operacions públiques que cregueu necessaris. S'entén que es tracta d'una proposta inicial que més endavant podreu canviar si fa falta, però heu de procurar afinar-la al màxim des de bon principi. Cal que hi apareguin com a mínim totes les operacions referenciades a l'enunciat. També es pot incloure la capçalera d'alguna operació privada, si la creieu necessària, però amb les operacions públiques és suficient.

   El que no heu de fer de cap manera en aquest primer lliurament és afegir res de codi ni d'estructures de dades (això correspondria al nivell d'implementació).

   Transcriurem tots els mòduls en un document en format PDF, per tal de facilitar-ne la correcció.

   A l'annex 1 trobareu un exemple orientatiu.

2. Diagrama de classes (document en format PDF), indicant les relacions d'ús (- - - - →), implementació (- - - - ▷) i extensió/refinament (——▷).

   Recordeu que la relació d'ús indica una dependència feble: considerem que un mòdul A usa un mòdul B quan en alguna operació d'A apareix algun paràmetre o resultat de tipus B.

7.1.2 Data límit

Hi ha una tasca al Moodle per fer el lliurament. Compte: la data límit és diferent per cada grup de pràctiques. Es recomana no esperar a la data límit. Lliureu-ho tan aviat com pugueu i aviseu el professor per tal que us ho corregeixi al més aviat possible.

7.2 Lliurament final

El lliurament final no serà al Moodle, sinó dins del vostre repositori de GitHub Classroom.[5]

7.2.1 Contingut

1. Codi font definitiu i complet (fitxers `.java`, dins de la carpeta `src`).

2. Fitxer JAR de l'aplicació (`out/artifacts/BeWater_jar/BeWater.jar`).

3. Biblioteques necessàries per executar el programa (fitxers `.jar`, carpeta `lib`). P.ex. els fitxers JAR de GraphStream. El programa ha de ser executable des de línia de comanda tal com s'indica a la secció 5.

4. Joc de proves utilitzat (fitxers `.txt`, carpeta `test`).

5. Document descrivint les diferents proves realitzades i els resultats obtinguts. Si voleu, podeu incloure-hi captures de pantalla. Cal que s'hi observi el funcionament de l'aplicació amb claredat (fitxer `proves.pdf`, carpeta `doc`).

6. Diagrama de classes (fitxer `diagrama.pdf`, dins de la carpeta `doc`).

7. Documentació del codi font en format HTML, generada amb doxygen (fitxers `.html` i auxiliars, dins de la subcarpeta `doc/html`).

   Cal documentar tant la part pública com la part privada, incloent una descripció general de cada mòdul, una descripció breu de cada operació, i la seva especificació pre-post. També cal descriure els atributs (dades) de les classes i els seus invariants. A l'apartat "Normes i exemples" del Moodle hi teniu un exemple de com documentar. Seguiu també les indicacions de l'apartat "Normes bàsiques per al codi font".

8. L'objectiu de la pràctica no és només tenir un programa que funcioni, sinó donar l'ocasió d'aplicar els principis de disseny que hem explicat a teoria en un cas real. Per això, serà convenient que repasseu els apunts i que refeu al vostre codi el que sigui necessari per complir-los. Us demanem, a més, que reflexioneu sobre quins avantatges i inconvenients tenen les diferents decisions de disseny que hagueu pres.

   Concretament, dels apunts del bloc Principis SOLID:

   - Detecteu 5 elements del vostre codi que sospitosos de "mal disseny". Heu de posar-hi el codi, o una part, i explicar per què.
   - Expliqueu 3 elements dels Principis SOLID que heu aplicat o que, amb el que ara heu après, aplicaríeu i com.

   Dels apunts del bloc Getters & Setters:

[5] A classe de pràctiques us explicarem com unir-vos al repositori que hem creat per a cada equip.

- Preneu 3 exemples de getters/setters del vostre codi (si n'hi ha) i expliqueu per què seria convenient modificar-los (o no), i com ho faríeu.

En resum, que argumenteu el disseny que heu escollit, i mostreu que heu entès la teoria que correspon. No pretenem que el disseny sigui perfecte, sinó que sigueu capaços d'entendre el que heu fet i defensar les vostres eleccions.

Per cada un dels elements i exemples, 5 línies hauria de ser suficient. Poseu-ho en un fitxer de nom `teoria.pdf` a la carpeta `doc`.

9. Opcional: document explicant les vicissituds viscudes durant el desenvolupament del projecte, i la vostra opinió sobre el projecte i l'assignatura (fitxer `comentaris.pdf`, carpeta `doc`).

7.2.2 Data límit

La data límit per al lliurament final és el **18 de maig de 2024**. Aquest lliurament consta com a PAC oficial al calendari de l'EPS, i per tant és improrrogable.

Ho heu d'entendre com si es tractés una data d'examen. Abans de les 23:59h del dia 18 de maig caldrà que tingueu tot el material demanat a les carpetes corresponents del vostre repositori de GitHub.

El lliurament és automàtic: al venciment del termini, el propi sistema del GitHub Classroom empaquetarà el vostre darrer *commit* i el posarà a disposició dels professors. Vosaltres no heu de fer res més enllà de posar el material a les carpetes corresponents abans que venci el termini (no és necessari crear cap *release*).

Després del termini de lliurament veureu que encara és possible pujar coses al repositori (no quedarà bloquejat). No obstant, la versió que es corregirà serà la que hi havia en el moment de vèncer el termini. No s'acceptaran excuses de retard. Per tant, per evitar problemes de qualsevol tipus convé que no espereu a darrera hora.

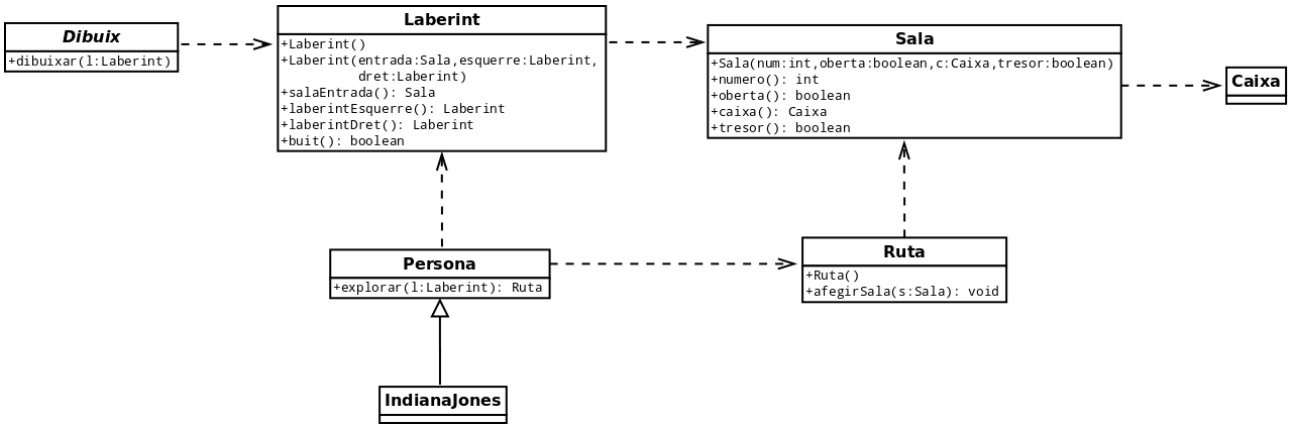
Annex 1

Exemple de disseny modular inicial (d'un context diferent). Posem les operacions públiques, sense implementar. No posem cap operació privada ni cap atribut de dades.

```
public class Laberint {

//Descripció general: Laberint format per sales,
//                    cadascuna d'elles amb una porta d'entrada
//                    i dues portes de sortida.

public Laberint()
//Pre:  ---
//Post: Crea un laberint buit.

public Laberint(Sala entrada, Laberint esquerre, Laberint dret)
//Pre:  entrada != null
//Post: Crea un laberint a partir d'una sala i dos laberints.
        La sala  entrada  s'estableix com a sala d'entrada al laberint.
        Les portes esquerra i dreta de sortida d'aquesta sala es
        connecten amb el laberint  esquerre  i  dret  respectivament.

public Sala salaEntrada()
//Pre:  ---
//Post: Retorna la sala d'entrada d'aquest laberint.

public Laberint laberintEsquerre()
//Pre:  ---
//Post: Retorna el laberint que hi ha a partir de la porta de sortida
        esquerra de la sala d'entrada d'aquest laberint.

public Laberint laberintDret()
//Pre:  ---
//Post: Retorna el laberint que hi ha a partir de la porta de sortida
        dreta de la sala d'entrada d'aquest laberint.

public boolean buit()
//Pre:  ---
//Post: Diu si aquest laberint és buit (sense cap sala)

}

-----------------------------------------------------------------------
```

```
public class Sala {

//Descripció general: Una habitació amb objectes

public Sala(int num, boolean oberta, Caixa c, boolean tresor)
//Pre:  ---
//Post: Crea una sala amb número identificador  num, i caixa
        amb objectes  c. Els booleans  oberta  i  tresor  indiquen
        si la porta d'entrada està oberta, i si la sala conté un
        tresor, respectivament.

public int numero()
//Pre:  ---
//Post: Retorna el número de sala.

public boolean oberta()
//Pre:  ---
//Post: Diu si la porta d'entrada està oberta.

public Caixa caixa()
//Pre:  ---
//Post: Retorna la caixa que hi ha dins la sala.

public boolean tresor()
//Pre:  ---
//Post: Diu si la sala conté un tresor.

}
-------------------------------------------------------------------------

public class Persona {

//Descripció general: Persona que visita un laberint, gastant una
                      unitat de vida a cada moviment (canvi de sala).

public Ruta explorar(Laberint l)
Pre:  ---
Post: Explora el laberint  l. Si troba un tresor, aleshores retorna
      la ruta seguida fins arribar a la sala del tresor.
      En cas contrari retorna una ruta buida.
}
```

```
-------------------------------------------------------------------------

public class IndianaJones extends Persona {

//Descripció general: Personatge de ficció, que explora laberints
                      seguint una estratègia òptima consistent en agafar
                      els objectes que troba pel camí que li poden
                      aportar major benefici, i no repetir mai camins
                      ja intentats (sobreescriu el mètode Explorar).

}

-------------------------------------------------------------------------

public class Ruta {

//Descripció general: Sequència de sales

public Ruta()
//Pre:  ---
//Post: Crea una ruta buida.

public void afegirSala(Sala s)
//Pre:  ---
//Post: Afegeix la sala  s  al final de la ruta.

}

-------------------------------------------------------------------------

public abstract class Dibuix {

//Descripció general: Mòdul funcional de dibuix de laberints

public void dibuixar(Laberint l)
//Pre:  ---
//Post: Dibuixa el laberint  l  en una finestra.

}

-------------------------------------------------------------------------
```


Dibuix
+dibuixar(l:Laberint)
Laberint
+Laberint()
+Laberint(entrada:Sala,esquerre:Laberint,
dret:Laberint)
+salaEntrada(): Sala
+laberintEsquerre(): Laberint
+laberintDret(): Laberint
+buit(): boolean
Sala
+Sala(num:int,oberta:boolean,c:Caixa,tresor:boolean)
+numero(): int
+oberta(): boolean
+caixa(): Caixa
+tresor(): boolean
Caixa
Persona
+explorar(l:Laberint): Ruta
Ruta
+Ruta()
+afegirSala(s:Sala): void
IndianaJones