MODELOS DE REDES

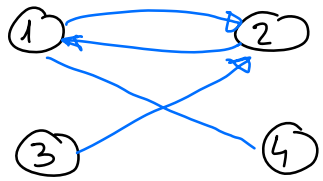
DEFINICIONES

- Una red es un par (N, G) donde N es un conjunto finito cualquiera y G es un subconjunto del producto cartesiano NxN.
- A los elementos de N los llamaremos NODOS y a los de G, ARCOS

- Habitualmente la representaremos así:

Si $N = \{1, 2, 3, 4\}$ y

$G = \{(1,2), (2,1), (3,2), (4,1)\}$



- RED NO DIRIGIDA : Si no distinguimos entre los arcos (i,j) y (j,i), cualesquiera que sean los nodos "i" y "j" en N, diremos que la red es No DIRIGIDA. En otro caso, la llamaremos DIRIGIDA.

CAMINOS

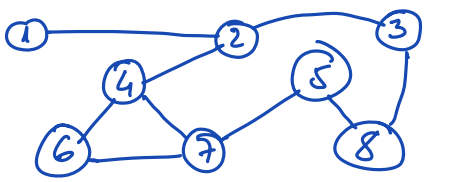
- En una red no dirigida, un camino desde el nodo "i" al nodo "j" es una sucesión finita de arcos,

$$(i, j_1), (j_1, j_2), \dots (j_r, j)$$

donde todos los nodos que aparecen son distintos.

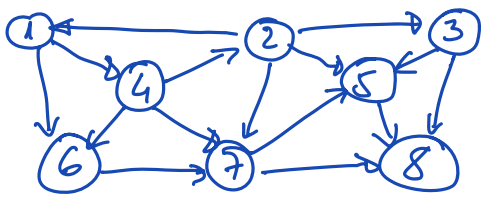
- En una red dirigida, un camino desde el nodo "i" al nodo "j" es una sucesión finita de arcos tal que el primero contiene al nodo "i", el último al nodo "j", todos los nodos que aparecen son distintos y cada arco tiene un nodo en común con el siguiente. Si dichos arcos son recorridos en el sentido que indica su flecha, el camino diremos que es dirigido.

EJEMPLOS En la red dirigida



(1,2)(2,3)(3,8) es un camino de 1 a 8

EJEMPLOS En la red dirigida



(2,1),(2,3),(3,8) es un camino de 1 a 8

(1,4),(4,7),(7,8) es un camino dirigido de 1 a 8

- En general, dos nodos se dirán conectados si existe un camino de uno a otro.
- Llamaremos ciclo a un camino formado por al menos dos arcos distintos, que conecta un nodo consigo mismo.

- Llamaremos subred de (N, G) a una red (M, A), donde M es un subconjunto cualquiera de N y A es el subconjunto de arcos de G que unen nodos de M (la intersección de MxM y G).

Diremos que la red es valorada si cada arco (i,j) tiene asociado un número real $c_{ij} \geq 0$.

PROBLEMA 1: ÁRBOL DE EXTENSIÓN DE EXPANSIÓN MÍNIMA.

- Supondremos que N tiene n nodos y que cada uno de ellos está conectado consigo mismo. Sin embargo, dichos arcos no los tendremos en cuenta para contabilizar el número de arcos de una subred de (N,G).
- Diremos que una subred (M,A) de (N,G) es conexa si todos los nodos de M están conectados por arcos de A. Así, (N,G) es una red conexa si todos sus nodos están conectados.
- Diremos que (M,A) es una componente conexa si es conexa y no está contenida estrictamente en otra subred también conexa.

ÁRBOL DE EXPANSIÓN

- Diremos que la subred (N,A) , siendo A subconjunto de G, es un árbol de expansión para (N,G), si es conexa y el cardinal de A es n-1.

PROPOSICIÓN 1. Si (N,G) es una red conexa, entonces card(G) ≥ card(N)-1.

Supongamos que card (N) = n. En la demostración vamos a renombrar sucesivamente los nodos y contabilizar los arcos.
Tomemos un modo cualquiera. Renombrémoslo como i_1.

Consideremos todos los nodos que están unidos a i_1. Sean $N_1=\{i_2, i_3, \dots i_{k_1}\}$ dichos nodos y $A_1=\{a_2, a_3, \dots a_{k_1}\}$, los arcos correspondientes.

Sean ahora, $N_2=\{i_{k_1+1}, \dots, i_{k_2}\}$, los nodos que están unidos a nodos de N_1 y $A_2=\{a_{k_1+1} \dots, a_{k_2}\}$, los arcos correspondientes .

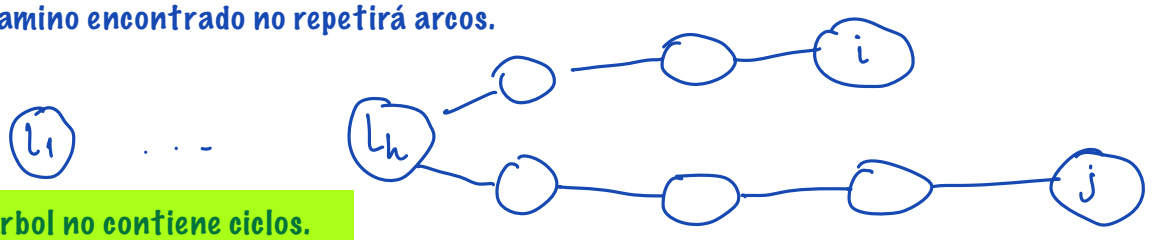
Repitiendo este argumento, llegaremos a que existe r tal que $N = \{i_1\} \cup N_1 \cup \dots \cup N_r$. Además, el cardinal del conjunto de arcos $A= A_1 \cup A_2 \cup \dots A_r$ es n-1.

Por otra parte, ni que decir tiene que la subred (N,A) es conexa.

PROPOSICIÓN 2. Si (N,G) es una red conexa y card(G) > card(N) - 1, entonces (N,G) contiene al menos un ciclo.

Usando Sean $i_1, N_1, \dots, N_r$, $A_1, \dots, A_r$, como en la proposición anterior ($A=\{a_1, \dots, a_{n-1}\} \subset G$). Sabemos que existe al menos otro arco más en G, digamos $a_n = (i,j)$, i,j en N.
Vamos a construir un camino de i a j con arcos de A que, junto al arco a_n, formará un ciclo en la red (N,G).

Supongamos que el nodo i está en N_{t_i} y que el nodo j está N_{t_j} . Por la construcción utilizada en la posición 1, tanto el nodo i como el nodo j están conectados con el nodo i_1. Si esos caminos no coinciden en ningún nodo antes de llegar a i_1, los usaremos para encontrar un camino desde i hasta j. Así que el camino encontrado pasaría por i_1. Pero es posible que coincidan en otro nodo anterior. Si llamamos i_h al primer nodo en que ocurre eso, el camino que construiremos llegará desde i hasta i_h, y de ahí, continuaremos hasta j.
De este modo, el camino encontrado no repetirá arcos.



COROLARIO 1 Un árbol no contiene ciclos.

Estudiaremos dos algoritmos, el de PRIM y el de KRUSKAL. En el último usaremos el concepto de conexión introducido más arriba. Empezaremos con éste.

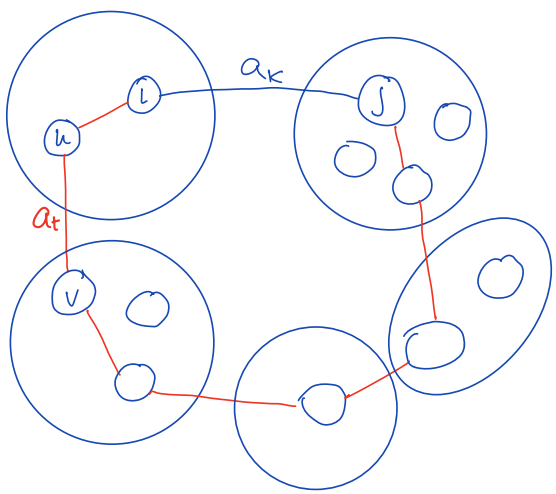
ALGORITMO DE KRUSKAL

Partimos de un conjunto vacío de arcos, A, al que iremos agregando arcos según avance el algoritmo.
Como paso 0, ordenamos los arcos según su longitud, de menor a mayor: Sean $a_1, a_2, \ldots, a_s$, los arcos renombrados y $c_1, \ldots, c_s$, sus longitudes.
Comenzando por el más corto y siguiendo ese orden establecido, en cada paso se analiza un arco distinto y se calculan las componentes conexas de la red (NA). Si el arco analizado une nodos de distintas componentes conexas, se mete en A. En caso contrario, no.

PROPOSICIÓN 3. El algoritmo funciona.

Supongamos que no; concretamente, supongamos que hasta el paso k-1 (en el que se analiza el arco a_{k-1}), el conjunto A está contenido en algún árbol de extensión mínima, digamos A_0, a_k une nodos de dos componentes conexas distintas y, sin embargo, $A \cup \{a_k\}$ no está contenido en ningún árbol de extensión mínima.

Podemos representar la situación del siguiente modo:



Dado que A_0 es un árbol, existe un camino formado por arcos de A_0 que van desde el nodo i hasta el nodo j (en rojo en el dibujo). Dado que los nodos i y j están en componentes conexas distintas, tiene que haber un arco de ese camino que salga de la componente conexa en la que está el nodo i. En el dibujo lo llamamos a_t. Notemos que si a este camino, le añadimos el arco a_k, tendremos un ciclo en la red (N,G).

Definamos un nuevo conjunto de arcos:

$$A_0' = (A_0 \setminus \{a_t\}) \cup \{a_k\}$$

Demostremos que la red (N, A_0') es un árbol. Para ello basta con probar que es conexa. Sean i y j dos nodos cualesquiera de N. Existe un camino formado por arcos de A_0 que conecta dichos nodos. Si ese camino no contiene al arco a_t, ese mismo camino estará en A_0'.

En caso contrario, no podríamos utilizar el arco a_t, pero podríamos unir los nodos u y v usando el resto del ciclo mencionado anteriormente. De esta manera construiríamos un camino que va desde el nodo i al nodo j, probando pues que esta nueva red es conexa.

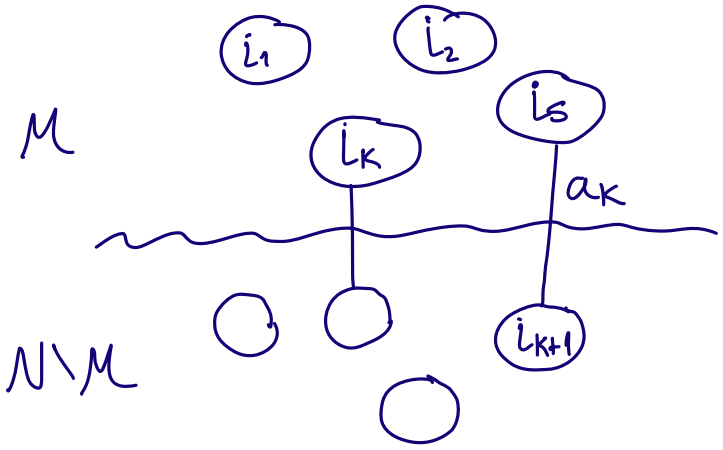
Para concluir la demostración, comparemos la extensión del nuevo árbol encontrado con la de A_0. La extensión de A_0' es la de A_0 más $c_k - c_t$. Pero el arco a_t aún no ha sido analizado por el algoritmo pues une nodos que están en distintas componentes conexas. Por tanto, $c_t \geq c_k$.

Si esos dos números fueran iguales, entonces tendríamos un nuevo árbol de extensión mínima, lo cual es una contradicción, pues, $A_0' \supset A \cup \{a_k\}$, y este último no está contenido en ningún árbol de extensión mínima. Y si fueran distintos entonces llegamos a otra contradicción pues A_0 no sería árbol de extensión mínima.

ALGORITMO DE PRIM

(Robert Clay Prim, 1957) (Jarnicks, checo, 1930)

Durante el algoritmo iremos renombrando nodos y arcos de la red original (N,G). Partimos de un conjunto M de nodos y un conjunto de arcos A, ambos vacíos.



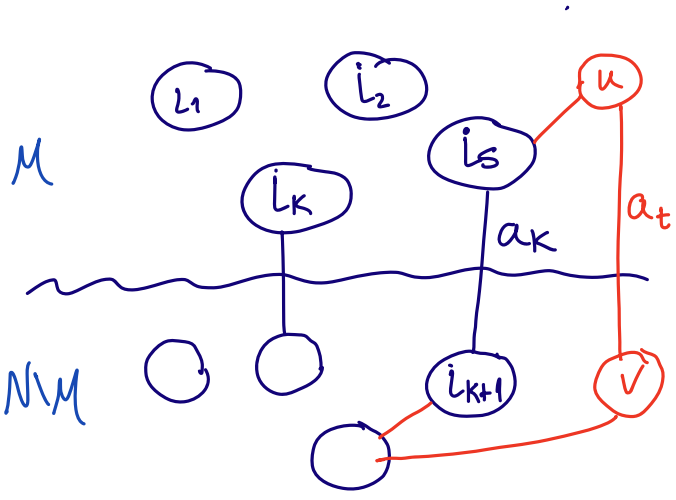
Paso 0. Consideremos un nodo cualquiera, i_s. Lo metemos en M.

Paso k. De todos los arcos que unen nodos de N\M con nodos de M, seleccionamos el más corto. Si hay varios, cogemos uno cualquiera. Llamamos a_k a dicho arco y lo metemos en A. Al nodo de N\M en el que incide ese arco, lo llamamos i_{k+1} y lo metemos en M.

El algoritmo terminará en n-1 pasos, siendo n el cardinal de N.

PROPOSICIÓN 4. El algoritmo funciona.

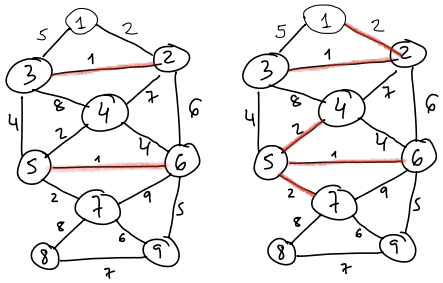
Supongamos que el algoritmo no funciona. Entonces existe un número natural k tal que hasta el paso k-1 toda va bien, esto es, el conjunto A está contenido en algún árbol de extensión mínima, digamos A_0, pero $A \cup \{a_k\}$ no está contenido en ningún árbol de extensión mínima. Según el algoritmo, el arco a_k une los nodos i_k e i_{k+1}. Dado que A_0 es un árbol, estos dos nodos deben estar unidos por un camino de arcos en A_0. En dicho camino, habrá algún arco que va desde M a N\M. Sea a_t uno de esos arcos. Puesto que el algoritmo elige el arco más corto que va desde M a N\M, $c_t \geq c_k$. Definamos un nuevo conjunto de arcos:

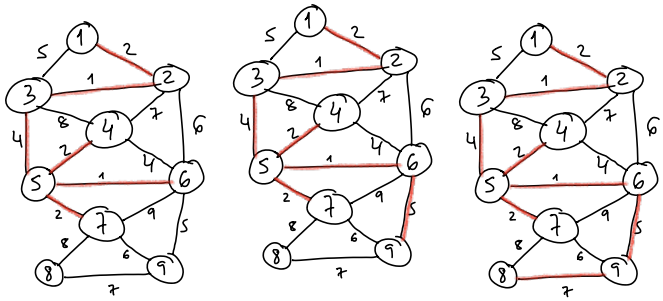


$$A_0' = (A_0 \setminus \{a_t\}) \cup \{a_k\}$$

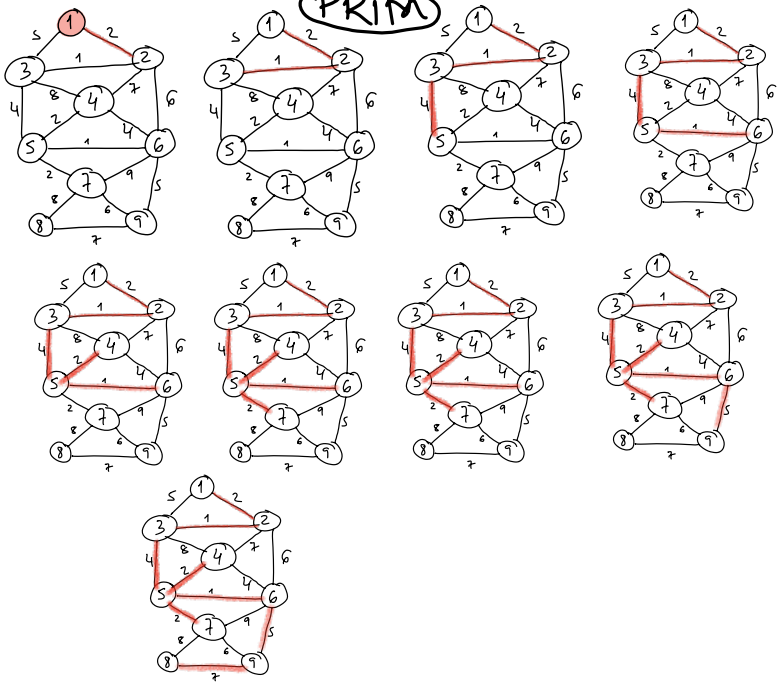
A partir de aquí, la demostración es idéntica a la del algoritmo anterior.

KRUSKAL

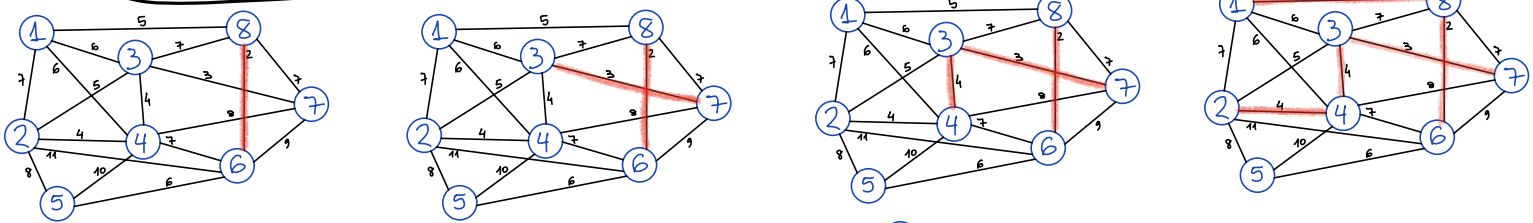




PRIM



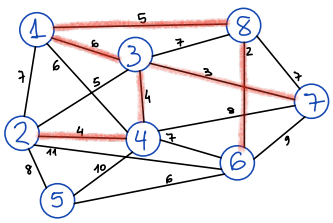
KRUSKAL



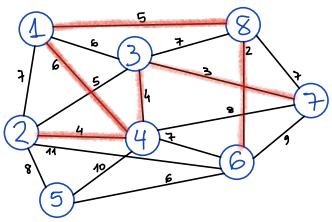
Son elegibles (1,3) o (1,4).

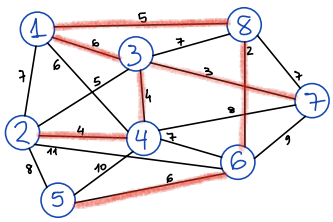
Con cada elección sale un árbol distinto.

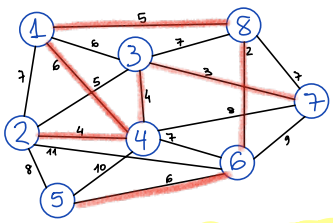
(1,3)



(1,4)

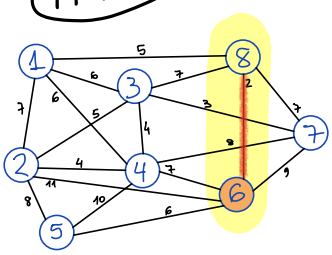


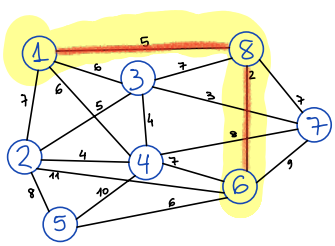




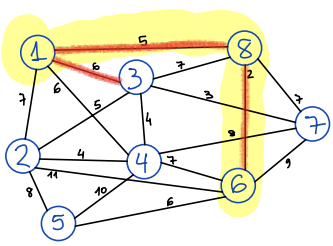
Tenemos dos árboles de extensión mínima

PRIM



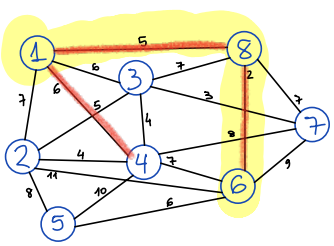


(1,3)



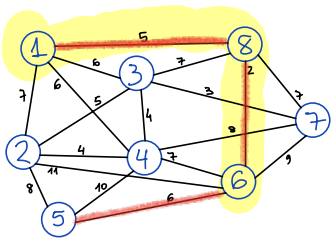
(A)

(1,4)



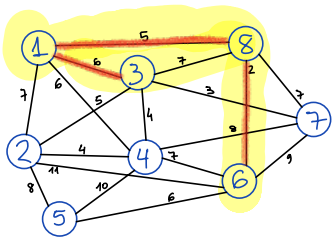
(B)

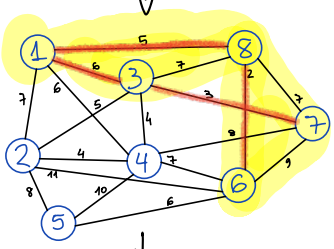
(5,6)

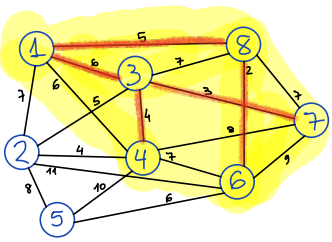


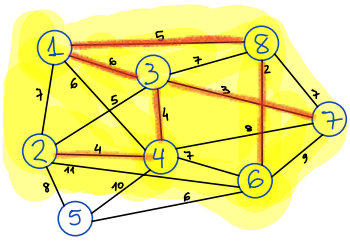
(C)

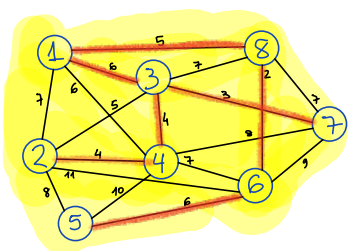
(A)





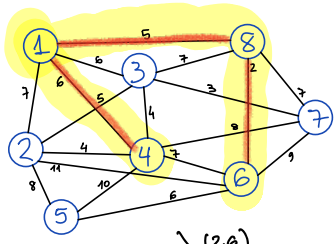






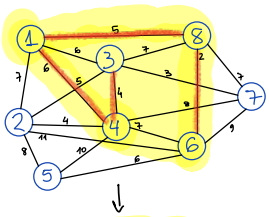
ÁRBOL Nº 1

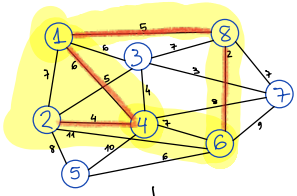
B)

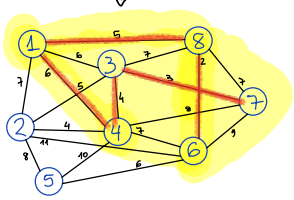


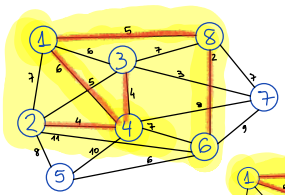
(3,4)

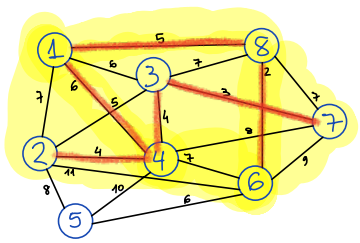
(2,6)

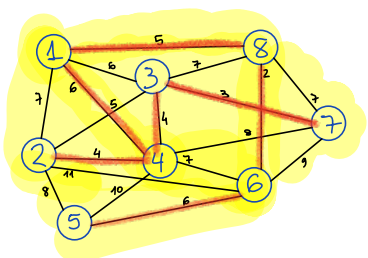






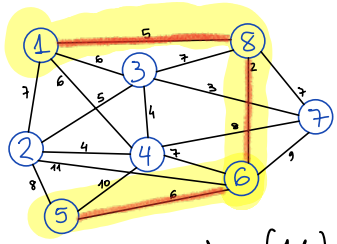






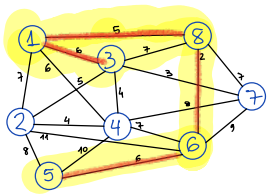
ÁRBOL Nº 2

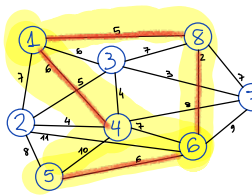
(C)



(1,3)

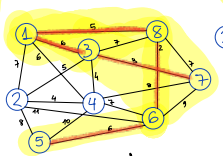
(1,4)

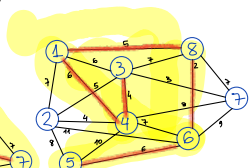


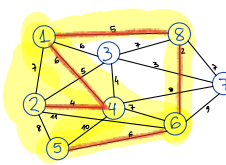


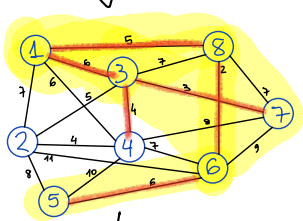
(3,4)

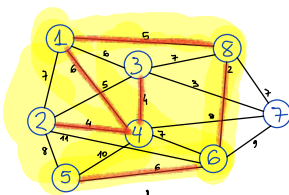
(2,4)











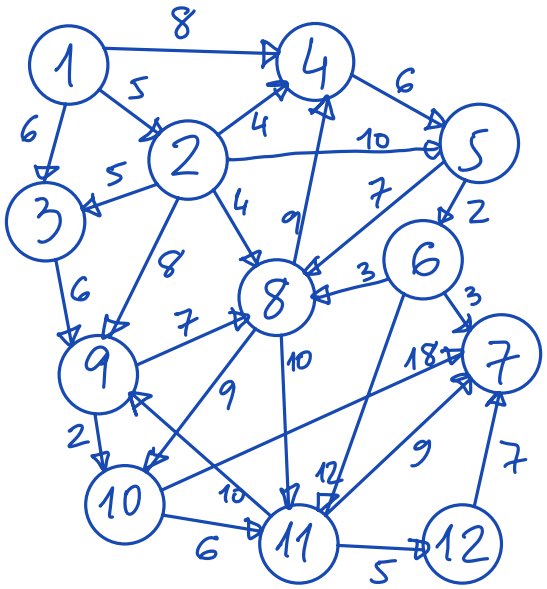
PROBLEMA 2: CAMINO MÁS CORTO.

Para este problema consideraremos una red (N,G) dirigida, con n nodos. Cada arco a_j de G, tendrá asociado un número real no negativo c_j que será la medida de dicho arco. Ese número puede representar la distancia entre los dos nodos que conecta. El problema que nos planteamos es el de encontrar el camino más corto entre dos nodos dados de la red.

El número c_j también podría representar el tiempo mínimo que se tarda en recorrer la distancia que hay entre los nodos que conecta el arco a_j. En ese caso, hablaríamos del problema de encontrar el camino más rápido.

Estudiaremos dos algoritmos, el de Dijkstra y el de Floyd.

Ejemplo:



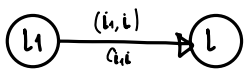
ALGORITMO DE DIJKSTRA

El algoritmo calcula el camino más corto que va desde un nodo cualquiera que denotaremos i_1 hacia cualquier otro nodo de la red.
El algoritmo irá renombrando el resto de los nodos, i_2, i_3, ..., hasta llegar a i_n.

En el paso k-ésimo se calcula, para cada nodo i de N no renombrado, un número que denotaremos por E_i^k y llamaremos etiqueta temporal del nodo i en el paso k. En ese cálculo interviene la etiqueta del paso anterior. A medida que avanza el algoritmo, algunas de esas etiquetas se harán permanentes. Concretamente, la etiqueta temporal del nodo i, E_i^k, calculada al comienzo de este paso (k+1)-ésimo, se hace permanente. Al terminar el algoritmo, este número será la medida del camino más corto que, empezando en el nodo i_1, acaba en dicho nodo.

Paso 1. Seleccionamos el nodo i_1. Le asignamos la etiqueta, ya permanente, $E_{i_1}^* = 0$.

Paso 2. Si i es un nodo tal que existe el arco (i_1, i) en G, le asignamos su etiqueta temporal E_i^2 del siguiente modo: $E_i^2 = c_{i_1 i}$, la distancia desde i_1 a i.



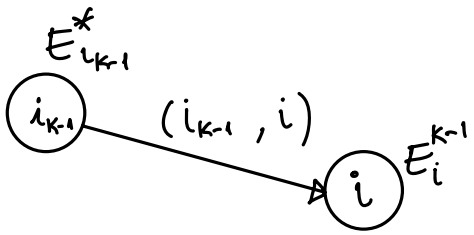
Si no existe el arco (i_1, i) en G, entonces E_i^2 será infinito.
Finalmente, seleccionamos el nodo que tiene la menor de las etiquetas temporales E_i^2. Renombramos ese nodo como i_2 y su etiqueta la hacemos permanente. La denotaremos así: $E_{i_2}^*$.

Paso k. Al comienzo de este paso, ya son permanentes las siguientes etiquetas: $E_{i_1}^*$, ..., $E_{i_{k-1}}^*$. Calculemos la etiqueta temporal E_i^k del nodo i en este paso:

-Si existe el arco (i_{k-1}, i) en G, entonces: $E_i^k = \min\left\{E_i^{k-1},\ E_{i_{k-1}}^* + c_{i_{k-1} i}\right\}$

-Si no existe el arco (i_{k-1}, i) en G, entonces: $E_i^k = E_i^{k-1}$

Finalmente, seleccionamos el nodo que tiene la menor de las etiquetas temporales E_i^k. Renombrados ese nodo como i_k y su etiqueta la hacemos permanente. La denotaremos así: $E_{i_k}^*$.



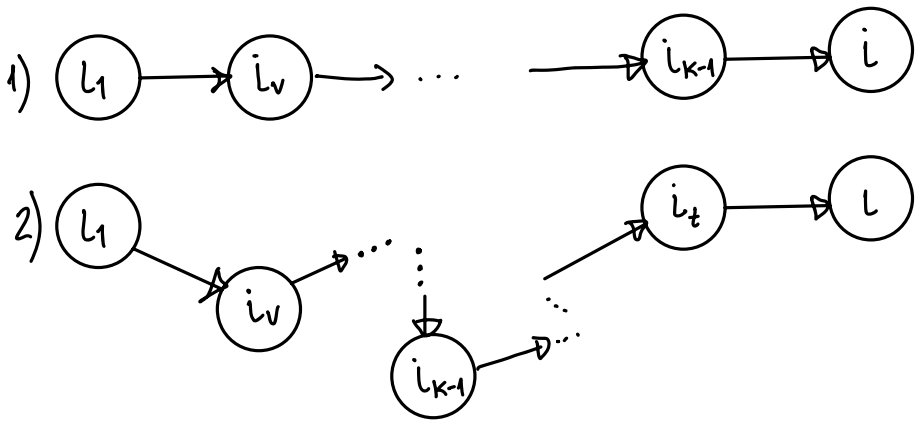
A continuación veamos que el algoritmo funciona. Necesitamos probar un par de proposiciones antes.

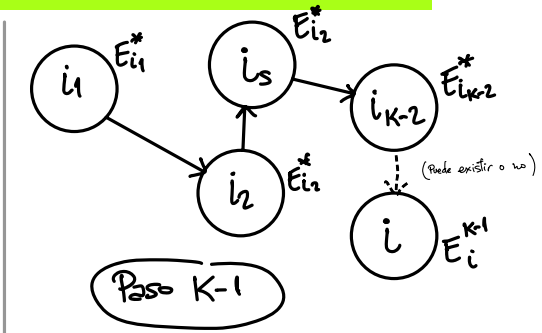
PROPOSICIÓN 5. La etiqueta permanente del nodo i_K es menor o igual que la del nodo i_{K+1}, k=1, ... , n-1

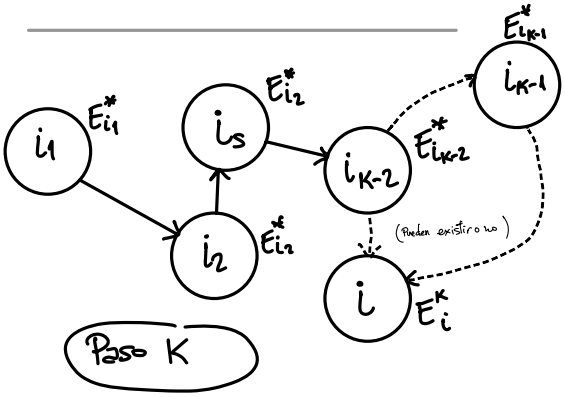
$E^*_{i_K}$ se define como el mínimo de las etiquetas del paso k, E^k_i . Entre ellas está $E^k_{i_{K+1}}$, que acabará siendo la etiqueta permanente $E^*_{i_{K+1}}$. Por otra parte, esta etiqueta será la menor de las etiquetas temporales del paso k+1. Esa etiqueta $E^{K+1}_{i_{K+1}}$ coincidirá con $E^k_{i_{K+1}}$ o con $E^*_{i_K} + c_{i_K i_{K+1}}$. En cualquier caso, por tanto, será un número mayor o igual que $E^*_{i_K}$.

PROPOSICIÓN 6. La etiqueta temporal del nodo i en el paso k es la medida del camino más corto desde i_1 hasta dicho nodo, en la subred de (N,G) compuesta por los nodos i_1, ... , i_{K-1}, i, cualquiera que sea i distinto de los nodos i_1, ... , i_{K-1}.

Realicemos esta demostración por inducción sobre k. La tesis es evidentemente cierta si k=1 y también para k=2 (comprobadlo). Supongamos que el resultado es cierto hasta k-1. Probémoslo para k:
Así, por hipótesis de inducción, la etiqueta temporal del nodo i en el paso k-1, E^{K-1}_i, es la medida del camino más corto para ir desde i_1 hasta i en la subred formada por los nodos i_1, ... , i_{K-2},i.
La tesis que hemos de probar es la del enunciado de la proposición. Fijémonos que en la subred de este enunciado aparece un nodo nuevo respecto a los que hay en la red bajo la hipótesis de inducción: el nodo i_{K-1}. Por tanto, para encontrar el camino más corto ahora, tendremos que analizar los caminos que van de i_1 a i pasando por dicho nodo, si es que los hay.
Esos caminos pueden ser de dos tipos:







El camino más corto del tipo 1 que se podría construir mediría $E^*_{i_{K-1}} + c_{i_{K-1} i}$ y el más corto de tipo 2 mediría al menos $E^*_{i_{K-1}}$. Ambas situaciones se tienen en cuenta en el cálculo de E^K_i .

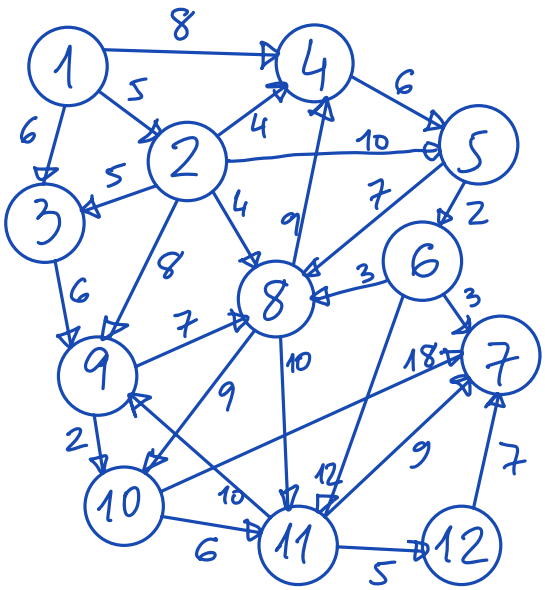
PROPOSICIÓN 7. La etiqueta permanente del nodo i es la medida del camino más corto desde i_1 hasta dicho nodo.

Esta demostración la haremos también por inducción sobre k, empezando con i_n y terminando en i_1 . La proposición anterior aplicada en el paso n-1 nos demuestra ésta para el caso i_n. Supongamos que la tesis de esta proposición es cierta hasta i_{K+1} y probémosla para i_K . Usando nuevamente la anterior proposición, la etiqueta permanente del nodo i_K es la medida del camino más corto que une el nodo i_1 con i_K en la subred formada por los nodos $i_1, \dots, i_K$. Al añadir a esta subred el resto de nodos, los caminos que unen esos dos nodos en toda la red (N,G) pasarán necesariamente por algunos de esos nodos añadidos y, por tanto, dichos caminos medirán al menos $E^*_{i_K}$ (aplicando la proposición 5), c.q.d..

ALGORITMO DE FLOYD

	1	2	3	4	5	6	7	8	9	10	11	12
1												
2												
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												

Apliquémoslo al ejemplo anterior : **Ejemplo:**



DATOS (Paso 0)

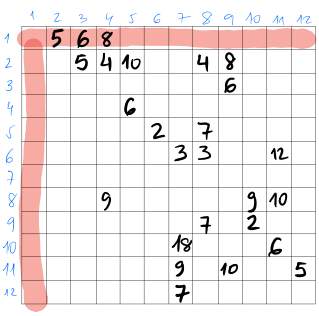
C

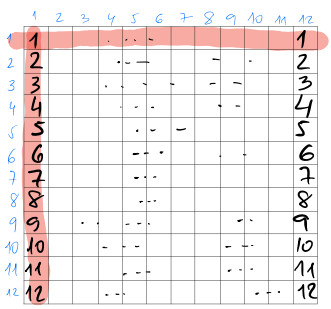
	1	2	3	4	5	6	7	8	9	10	11	12
1		5	6	8								
2			5	4	10			4	8			
3									6			
4					6							
5						2		7				
6							3	3			12	
7												
8				9						9	10	
9								7		2		
10							18				6	
11							9		10			5
12							7					

D

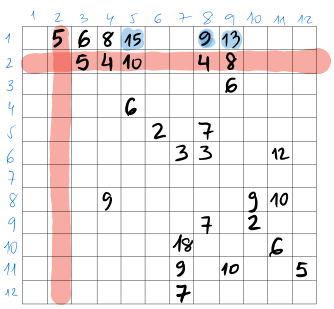
	1	2	3	4	5	6	7	8	9	10	11	12
1	1											1
2	2											2
3	3											3
4	4											4
5	5											5
6	6											6
7	7											7
8	8											8
9	9											9
10	10											10
11	11											11
12	12											12

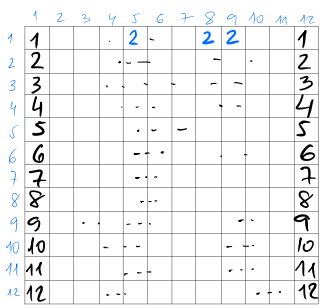
PASO 1 (No cambia nada)



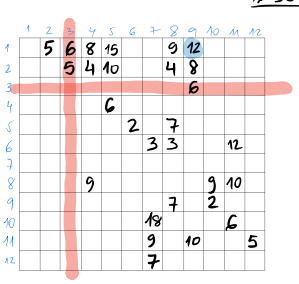


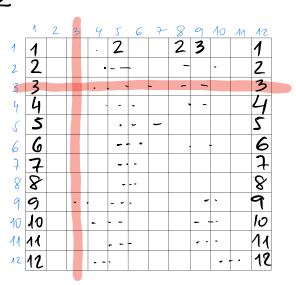
PASO 2



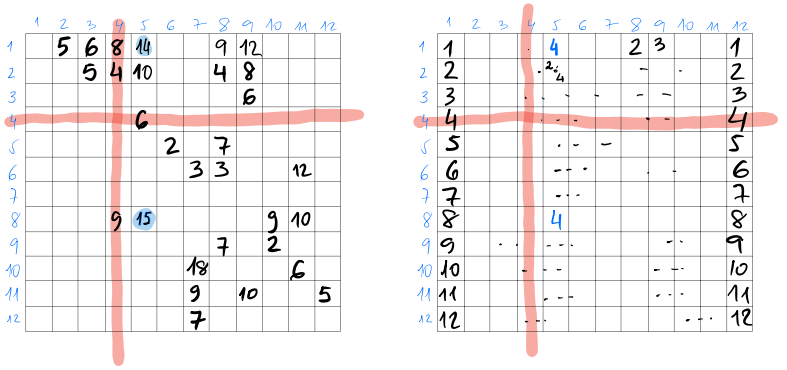


PASO 3

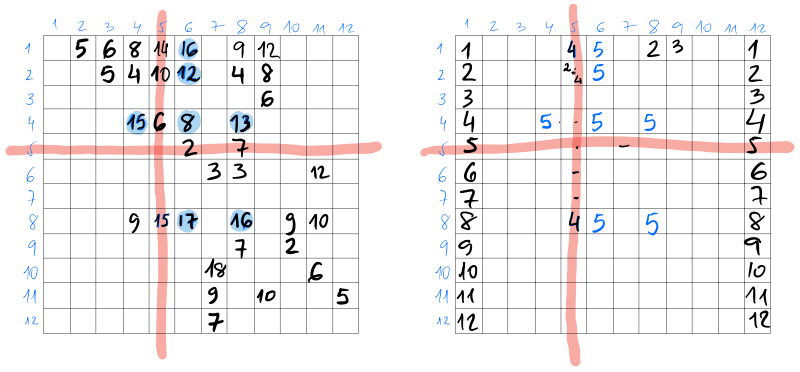




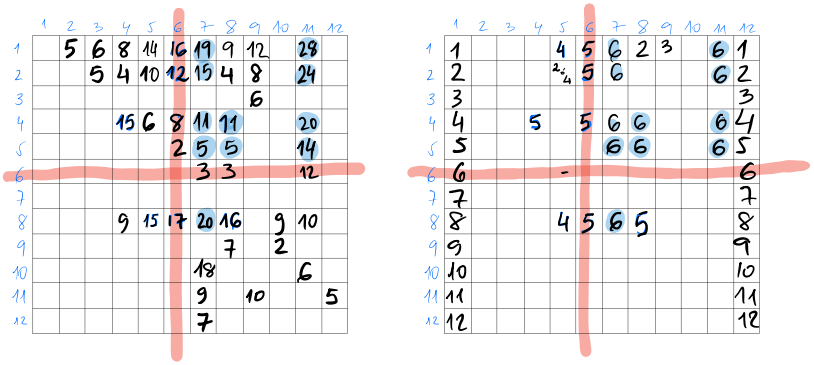
PASO 4



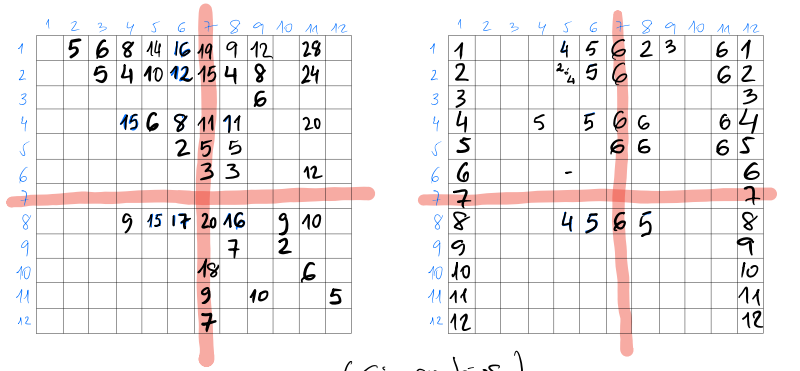
PASO 5



PASO 6

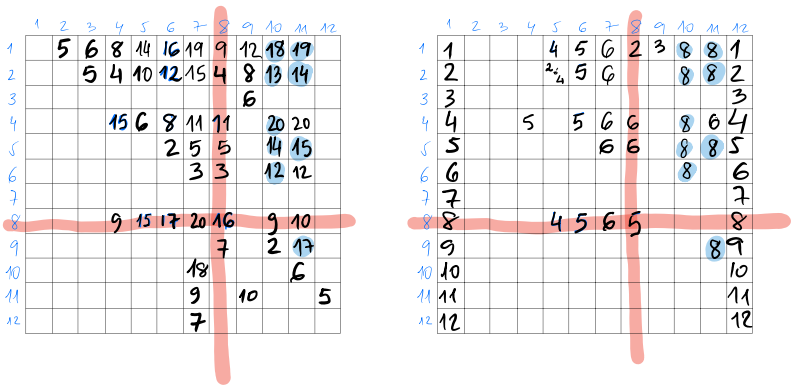


PASO 7



(sin cambios)

PASO 8



	1	2	3	4	5	6	7	8	9	10	11	12
1												
2												
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												

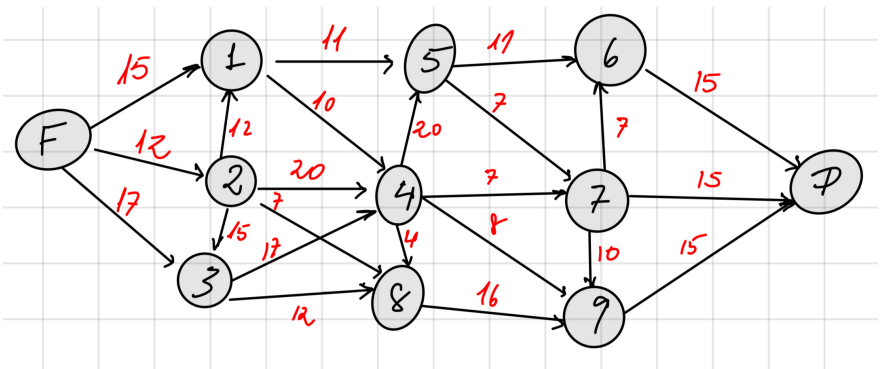
	1	2	3	4	5	6	7	8	9	10	11	12
1												
2												
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												

PROBLEMA 3: FLUJO MÁXIMO.

Para plantear este problema usaremos una red dirigida (N,G) con las siguientes propiedades:
a) Existe un nodo del que sólo salen arcos; lo llamaremos F (fuente). (Esto significa que no existe ningún arco de la forma (i,F) en G).
b) Existe un nodo al que sólo llegan arcos; lo llamaremos P (pozo).
c) Cada arco (i,j) en G tendrá asociado un número real no negativo C_{ij} .

Cada arco en la red (N,G) representa una tubería (acequia, ...) por el que puede circular agua (gas, petróleo, ...). La capacidad (volumen de agua por unidad de tiempo = flujo) de un arco es el número real mencionado anteriormente. El problema consiste en calcular el flujo máximo de agua que puede ser enviado desde la fuente al pozo a través de la red (N,G).

Ejemplo



Una solución factible X para este problema determinará el flujo de agua que se enviará por cada arco; por tanto, para cada arco (i,j), X_{ij} será un número real no negativo, acotado por la capacidad del arco. Supondremos que todo el agua que sale de la fuente llega al pozo. Esa cantidad sería el flujo asociado a la solución X. Para que se cumpla esa condición es necesario y suficiente que todo el flujo de agua que entra a un nodo coincida con el que sale de él.

El problema que habrá que resolver será el siguiente:

$$\text{Max } \phi(x) = \sum_{(F,j)\in G} X_{Fj}$$

$$\text{S.a. } \sum_{(i,k)\in G} X_{ik} = \sum_{(k,j)\in G} X_{kj}$$

$$\text{Si } k \in N \setminus \{F,P\}$$

$$0 \le X_{ij} \le C_{ij}\ ,\ (i,j)\in G$$

donde $\phi(x)$ es el flujo asociado a X.

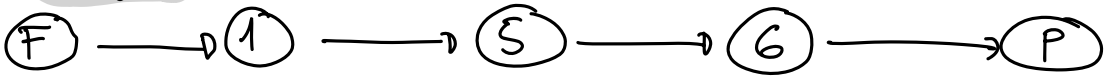
Se trata, efectivamente, de un problema de programación lineal que podríamos resolver usando el algoritmo del Símplex. Aquí lo vamos a resolver usando otro algoritmo:

ALGORITMO DE FORD-FULKERSON

Para iniciar el algoritmo necesitamos una solución factible X. Si no se tiene otra, podemos comenzar con X=0, que es evidentemente factible. En cada paso del algoritmo se mejorá la solución obtenida en el paso anterior mediante la creación de un camino que va de la fuente al pozo. Utilizaremos dos tipos de caminos:
a) Caminos cuyos arcos son recorridos en el sentido de su flecha (diremos 'hacia adelante') ('Caminos Dirigidos').
b) Caminos que pueden incluir arcos recorridos 'hacia atrás'. ('Caminos No dirigidos')

Ejemplos usando la red anterior:

Dirigido:



No dirigido

Describamos las reglas que se han de cumplir para poder usar esos caminos.
Supongamos que estamos en un paso cualquiera del algoritmo. Como mencioné antes, dispondremos de una solución factible X. Atendiendo a esta solución, se podrán definir los caminos mencionados anteriormente:

(i) Un arco (i,j) podrá ser recorrido hacia adelante y, por tanto, pertenecer a un camino dirigido o no, si $X_{ij} < C_{ij}$.
(ii) Un arco (i,j) podrá ser recorrido hacia atrás y, por tanto, pertenecer a un camino no dirigido, si $X_{ij} > 0$.

El algoritmo termina cuando no pueda encontrar un camino de la fuente al pozo en las condiciones dadas.

PROPOSICIÓN 1. Consideremos una solución factible X. El flujo que sale de la fuente coincide con el flujo que llega al pozo.

Se tiene que

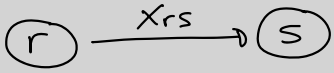
$$\phi(x) = \sum_{(F,j)\in G} X_{Fj}$$

$$\sum_{(i,K)\in G} X_{iK} = \sum_{(K,j)\in G} X_{Kj} \quad \text{si } K \in N \setminus \{F,P\}$$

Sumamos todas estas ecuaciones

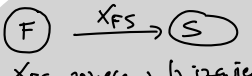
$$\phi(x) + \sum_{K\neq F,P} \sum_{(i,K)\in G} X_{iK} = \sum_{(F,j)\in G} X_{Fj} + \sum_{K\neq F,P} \sum_{(K,j)\in G} X_{Kj}$$

$r, s \neq F, P$



X_{rs} está en el miembro de la izquierda para $K=s$ y en el de la derecha para $K=r$. Por tanto, se cancelan

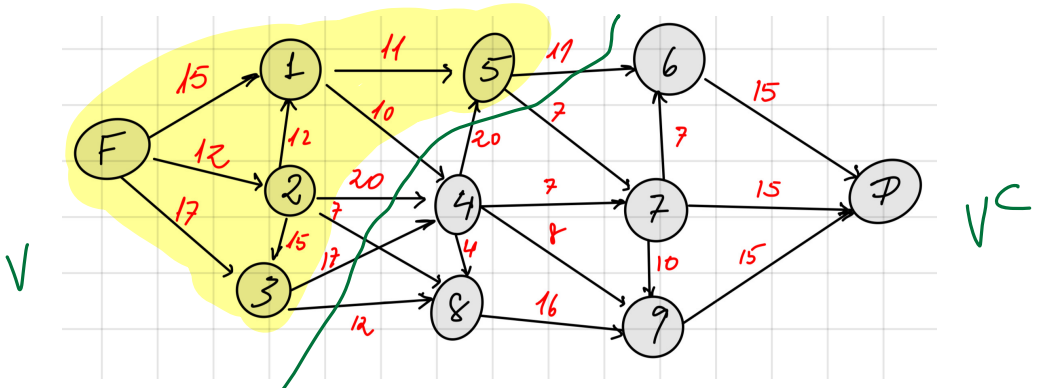
$s \neq P$



X_{FS} aparece a la izquierda para $K=s$. Se cancelan

$$\phi(x) = X_{FP} + \sum_{\substack{K\neq F \\ (K,P)\in G}} X_{KP} = \sum_{(K,P)\in G} X_{KP}$$

DEFINICIÓN (Corte). En la red (N,G), llamaremos "corte V" a una partición del conjunto de nodos formada por dos subconjuntos, V y su complementario, tal que la fuente está en V y el pozo en el complementario. Llamaremos capacidad del corte V a la suma de las capacidades de los arcos que salen de V y llegan a su complementario.



$V = \{F, 1, 2, 3, 5\}$ $V^C = \{4, 6, 7, 8, 9, P\}$

$Cap(V) = C_{56} + C_{57} + C_{14} + C_{24} + C_{28} + C_{34} + C_{38} = 84$

PROPOSICIÓN 2. Consideremos una solución factible X y un corte V. El flujo asociado a la solución X es menor o igual que la capacidad del corte V.

Esta demostración es similar a la de la proposición 1.

Se tiene que

$$\phi(x) = \sum_{(F,j)\in G} X_{Fj}$$

$$\sum_{(i,K)\in G} X_{iK} = \sum_{(K,j)\in G} X_{Kj} \quad \text{si } K \in N \setminus \{F,P\}$$

En lugar de sumar todas las ecuaciones de equilibrio, sólo lo haremos con aquellas en las que $k \in V$.

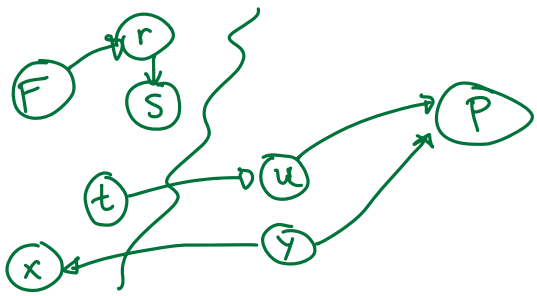
$$\phi(x) + \sum_{\substack{K\neq F,P \\ K\in V}} \sum_{(i,K)\in G} X_{iK} = \sum_{(F,j)\in G} X_{Fj} + \sum_{\substack{K\neq F,P \\ K\in V}} \sum_{(K,j)\in G} X_{Kj}$$

$r, s \in V$, $r, s \neq F, P$

r —x_{rs}→ s

x_{rs} está en el miembro de la izquierda para $K=s$ y en el de la derecha para $K=r$. Por tanto, se cancela

$$\phi(x) + \underbrace{\sum_{\substack{K\neq F \\ K\in V}} \sum_{\substack{(i,K)\in G \\ i\notin V}} X_{iK}}_{V/\ 0} = \sum_{\substack{(F,j)\in G \\ j\notin V}} X_{Fj} + \sum_{\substack{K\neq F \\ K\in V}} \sum_{\substack{(K,j)\in G \\ j\notin V}} X_{Kj}$$

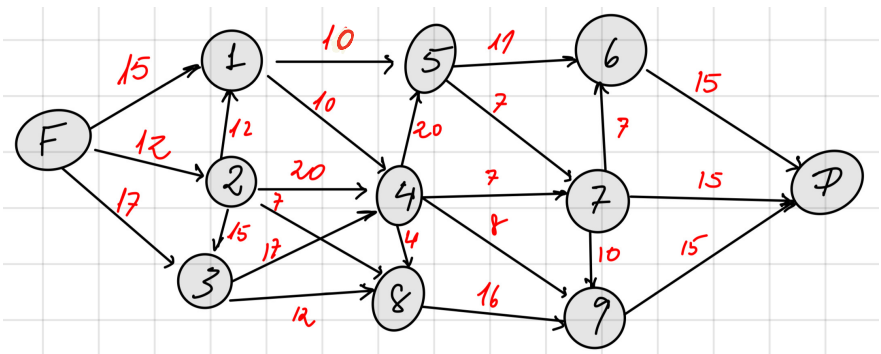


$$\phi(X) \le \sum_{k \in V} \sum_{\substack{(k,j)\in G \\ j \notin V}} x_{kj} \le \sum_{k \in V} \sum_{\substack{(k,j)\in G \\ j \notin V}} c_{kj} = cap(V)$$

COROLARIO (Criterio de Optimalidad). Consideremos una solución factible X y un corte V tales que el flujo de uno coincide con la capacidad del otro. Entonces, la solución X es óptima y el corte es el de capacidad mínima.

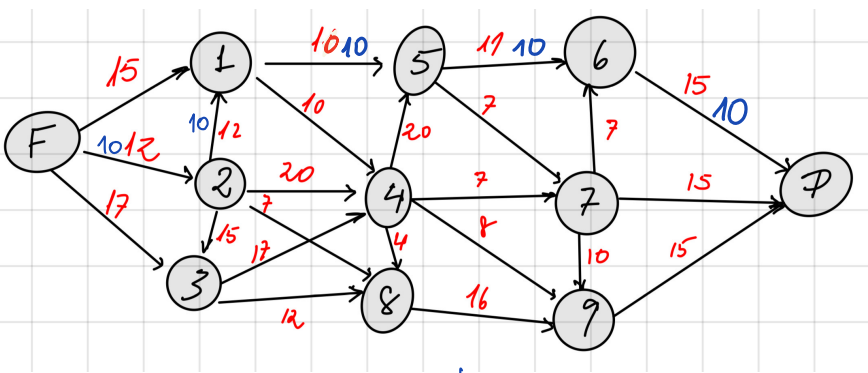
La demostración se deja al lector.

Ejemplo de aplicación del algoritmo



En azul iremos denotando los caminos desde (F) a (P) que cumplen las condiciones dadas:

PARTIMOS de la solución $X=0$, $\phi(X)=0$



$\phi = 10$

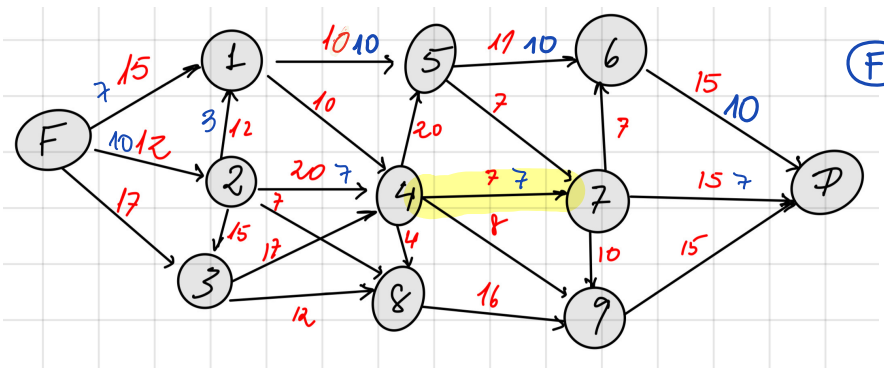
Paso 1

$a = \min\{C_{ij} - X_{ij},$ si (i,j) en el camino y recorrido hacia adelante$\}$

$a = 10$

$X'_{ij} = X_{ij} + a$ si $(i,j) \in$ camino

$X'_{ij} = X_{ij}$ si $(i,j) \notin$ camino



$\phi = 17$

Paso 2

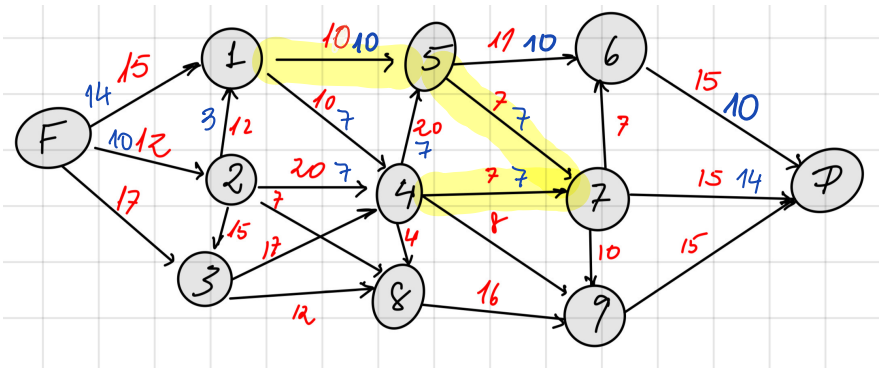
$a = \min \begin{cases} C_{ij} - X_{ij}, & \text{si (i,j) en el camino y recorrido hacia adelante} \\ X_{ij}, & \text{si (i,j) en el camino y recorrido hacia atrás} \end{cases}$

$a = 7$

$X'_{ij} = X_{ij} + a$ si $(i,j) \in$ camino (hacia adelante)

$X'_{ij} = X_{ij} - a$ si $(i,j) \in$ camino (hacia atrás)

$X'_{ij} = X_{ij}$ si $(i,j) \notin$ camino

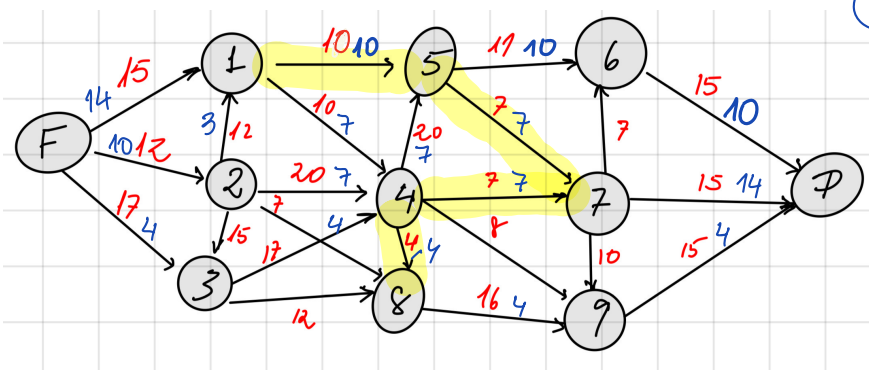


Paso 3

F → 1 → 4 → 5 → 7 → P

$a = 7$

$\phi = 24.$

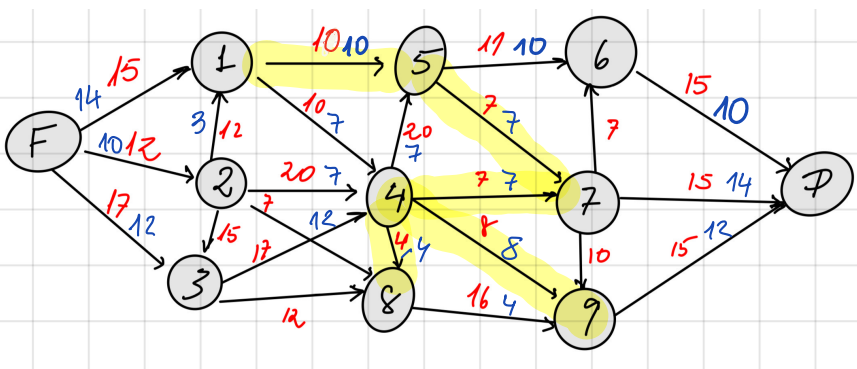


Paso 4

F → 3 → 4 → 8 → 9 → P

$a = 4$

$\phi = 28$

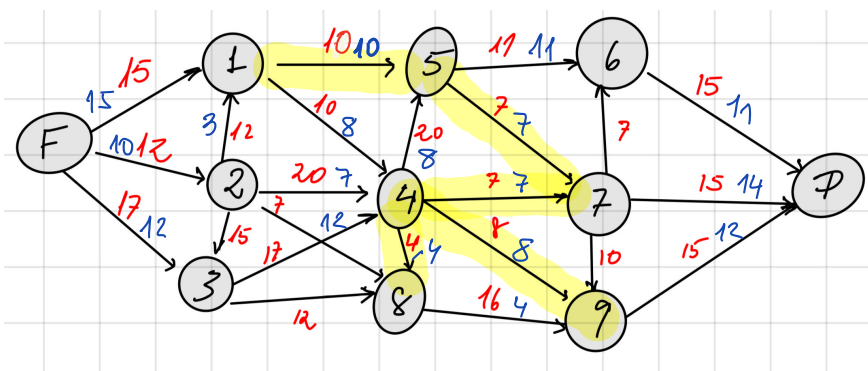


Paso 5

F → 3 → 4 → 9 → P

$a = 8$

$\phi = 36$

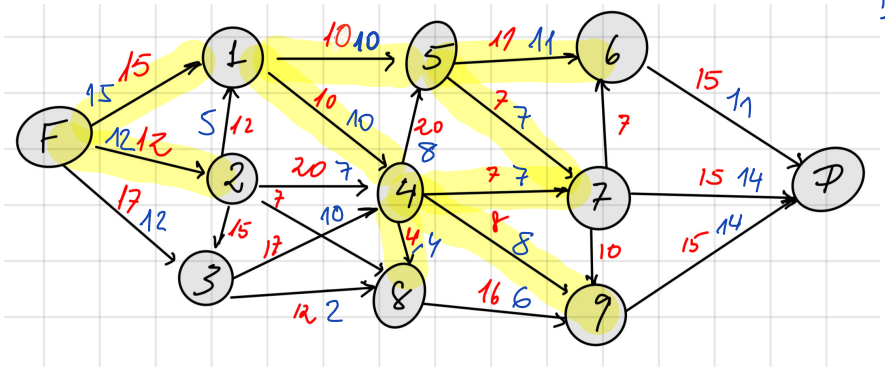


Paso 6

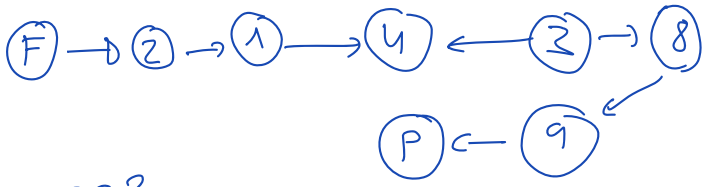
F → 1 → 4 → 5 → 6 → P

$a = 1$

$\phi = 37$

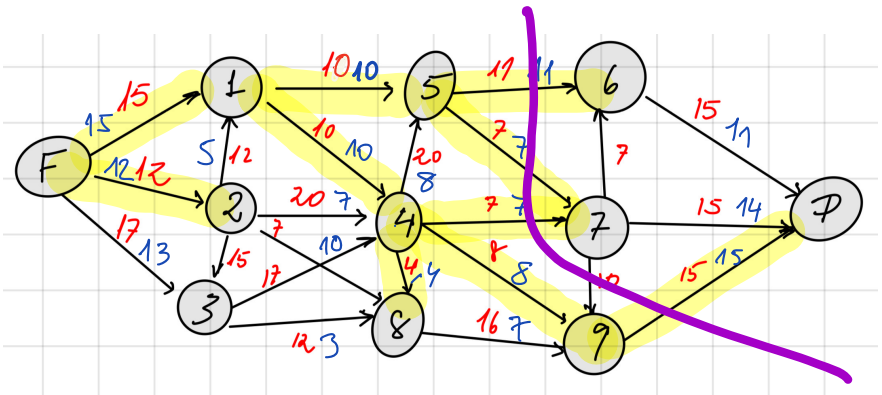


Paso 7



$a = 2$

$\phi = 39$



Paso 8

$a = 1$

$\phi = 40$

Corte:

$$V = \{F, 3, 4, 8, 2, 1, 5, 9\} \quad V^C = \{6, 7, P\}$$

$$Cap(V) = \sum_{\substack{(i,j)\in G \\ i\in V \\ j\in V^C}} C_{ij} = C_{56} + C_{57} + C_{47} + C_{9P} =$$

$$= 11 + 7 + 7 + 15 = 40$$

Como $\phi(X) = Cap(V) \Rightarrow X$ es óptima.

la solución actual

PROBLEMA 4. GESTIÓN DE PROYECTOS.

Un proyecto es un conjunto ordenado de actividades (o trabajos), $(A, \leq)$.

$A \leq B \Leftrightarrow$: la actividad A tiene que estar terminada antes de que empiece a ejecutarse la B. Diremos que A es antecesora a B.

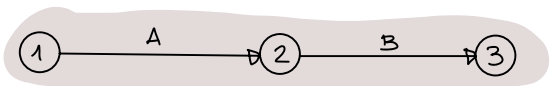
Cada actividad A tiene asociado un nº real positivo, t_A , que es el tiempo que dura la ejecución de A. El primer problema que trataremos será el de encontrar el tiempo mínimo para realizar el proyecto (sin otros condicionamientos que los ya expuestos). Realizar el proyecto produce un gasto que no interviene en el cálculo de ese tiempo mínimo.

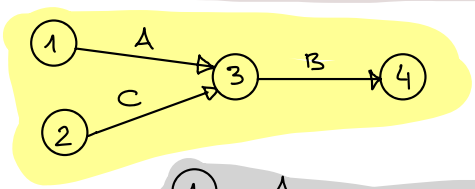
Luego, abordaremos el problema de reducir ese tiempo mínimo de ejecución del proyecto con un presupuesto de gastos adicional. Ese presupuesto adicional se usará para acortar la duración de algunas de las actividades.

MÉTODO DEL CAMINO CRÍTICO

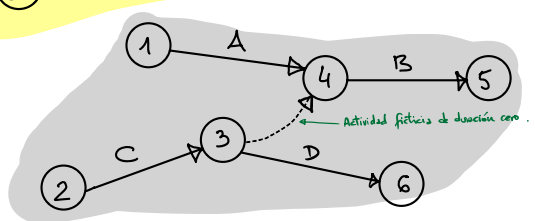
Empezaremos creando una red dirigida (N,G) que represente al proyecto. Las actividades del mismo formarán parte de G, pero es posible que haya que incluir algunos arcos adicionales que llamaremos ficticios. La duración de estas actividades ficticias será de 0. Los nodos serán instantes en el tiempo y en la red tendremos que hacer notar el orden establecido entre las distintas actividades. Así,

$A \leq B$ lo representaremos así:





$A \leq B, C \leq B$:



$A \leq B, C \leq B, C \leq D$:

De esta forma se puede crear una red (N,G), donde G está formada por las actividades que conforman el proyecto y las ficticias necesarias para establecer las preferencias, según se ha explicado más arriba. Además, crearemos un nodo inicial, tipo fuente, del que partan todas las actividades que no tiene predecesoras, usando actividades ficticias, y también un nodo final, tipo pozo, donde incidirás las actividades finales.

Los nodos representan instantes en el tiempo. Así, el nodo inicial lo consideraremos el instante en que se comienza a ejecutar el proyecto y el nodo final, su culminación.

Lo último que haremos para determinar la red será darle nombre a los nodos. Lo haremos con números naturales de forma que si existe un arco que vaya desde el nodo i al nodo j, i sea menor que j. Para conseguir ese podemos seguir el siguiente proceso. A la fuente le damos el 1. Si la eliminamos de la red, así como los arcos que salen de ella, habrá un nodo fuente en la subred resultante. Ese será el 2. Si quedasen varias fuentes, digamos r fuentes, asignaríamos los nombres 2, 3, ... , r+1 de cualquier manera. Este proceso se sigue hasta llegar al pozo.

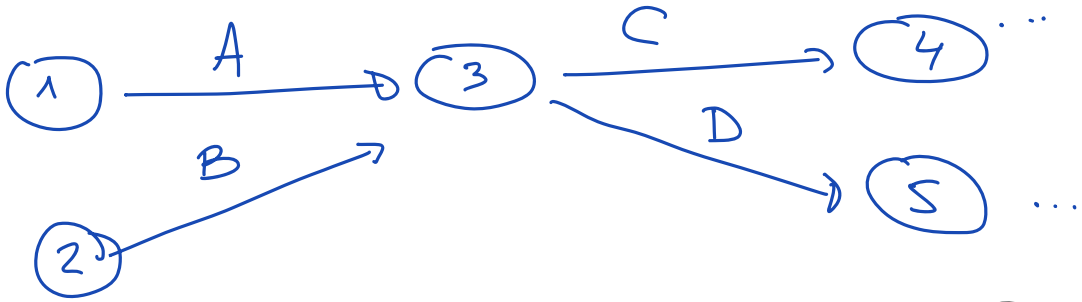
El algoritmo, por fin:

A cada nodo i, vamos a asignar dos números reales (que denotaremos ETi y LTi), calculados del siguiente modo:
El orden de cá´culo de los ET´s serán calculados según el orden natural:
ET1 =0. Una vez calculados los k-1 primeros, ETk=max

Sea (N, G) la red (de precedencias) construida.

G será la unión de A con un conjunto de actividades ficticias y N el conjunto de nodos necesario para establecer ese orden de precedencia entre las actividades en A.

Habrá actividades que no tengan predecesoras. Esas las representaremos de modo que sus arcos salgan de un mismo nodo que establecerá el inicio del proyecto. De hecho, todos los nodos determinan un instante de tiempo:



El instante de tiempo asociado al nodo (3) es aquel en el que A y B ya se han ejecutado y C y D pueden empezar a hacerlo.

Es fácil conseguir, tal vez con la ayuda de actividades ficticias, que haya un nodo del que sólo salen flechas

y otro nodo al que sólo llegan flechas. El primero indica el inicio de la ejecución del proyecto y el último, su finalización.

ALGORITMO DEL CAMINO CRÍTICO

Paso 0: Enumerar los nodos de modo que si una actividad está representada por el arco (i,j), entonces $i<j$.

$ET_1 = 0$.

Una vez obtenidos ET_2 hasta ET_{K-1},

$$ET_K = \max \{ ET_i + t_{iK} : (i,K) \in G \}$$

En el paso n habremos calculado ET_n.

$$LT_n = ET_n$$

Una vez obtenidos LT_{n-1} hasta LT_{K+1},

$$LT_K = \min \{ LT_j - t_{Kj} : (K,j) \in G \}$$

PROPOSICIÓN

(a) ET_n es la duración mínima del proyecto

(b) $ET_i \leq LT_i$, $\forall i \in N$.

(c) ET_i es el instante de tiempo en que, como muy pronto, se pasa por el nodo i

(d) LT_i es el instante de tiempo en que, como muy tarde, se puede pasar por el nodo i para terminar

el proyecto en el tiempo mínimo ET_n.

DEFINICIÓN Un camino crítico para el proyecto es un camino dirigido de ① a (n) tal que para cada (i,j) del camino se verifica que $ET_i = LT_i$, $ET_j = LT_j$ y $t_{ij} = ET_j - ET_i$

PROPOSICIÓN Existe un camino crítico.

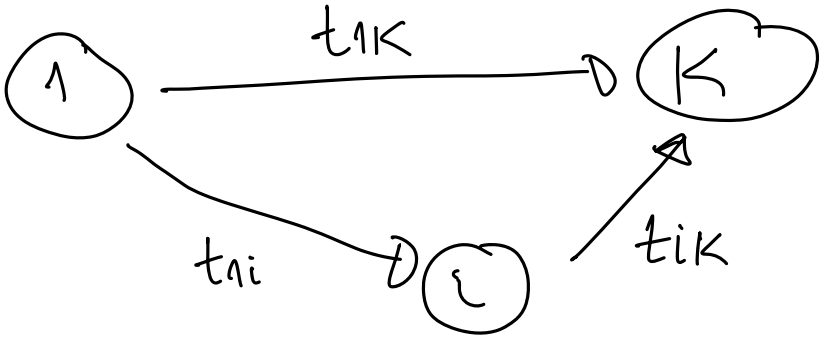
Sea K un nodo tal que $LT_1 = LT_K - t_{1K}$.

Supongamos que $LT_1 = a \geq 0$, esto es, $LT_1 - ET_1 = a$.

Queremos probar que $LT_K - ET_K = a$.

Si $ET_K = ET_1 + t_{1K}$. Dado que $LT_K = LT_1 + t_{1K} =$ $= a + t_{1K}$, entonces $LT_K - ET_K = LT_1 - ET_1 = a$

Si $ET_K > ET_1 + t_{1K}$, entonces, entre otras situaciones similares posibles, existe un nodo i tal que $ET_K = ET_i + t_{iK} = ET_1 + t_{1i} + t_{iK}$, siendo $t_{1i} + t_{iK} > t_{1K}$



Así: $a = LT_1 = LT_K - t_{1K} > LT_K - (t_{1i} + t_{iK})$

y $\left\{ LT_i \leq LT_K - t_{iK} \right.$

$$\left\{ LT_1 \leq LT_i - t_{1i} \leq LT_K - t_{iK} - t_{1i} = LT_K - (t_{1i} + t_{iK}) \right.$$

Contradicción

En definitiva, a partir del nodo (1), hemos encontrado otro nodo (K) tal que $\begin{cases} ET_1 + a = LT_1 \\ ET_K + a = LT_K \end{cases}$ y $t_{1K} = ET_K - ET_1$

Aplicando un razonamiento similar, encontraríamos otro nodo (J) tal que

$$\begin{cases} ET_j + a = LT_j \\ t_{Kj} = ET_j - ET_K \end{cases}$$

Así encontraríamos un camino dirigido de (I) a (h)

donde cada arco (i,j) verifica que:

$$\begin{cases} ET_i + a = LT_i \\ ET_j + a = LT_j \\ t_{ij} = ET_j - ET_i \end{cases}$$

En particular, $ET_h + a = LT_h$, así que $a = 0$.

C.Q.D.

PROPOSICIÓN Si los tiempos t_{ij} fueran distancias, un camino crítico resuelve el problema de encontrar el camino más largo.