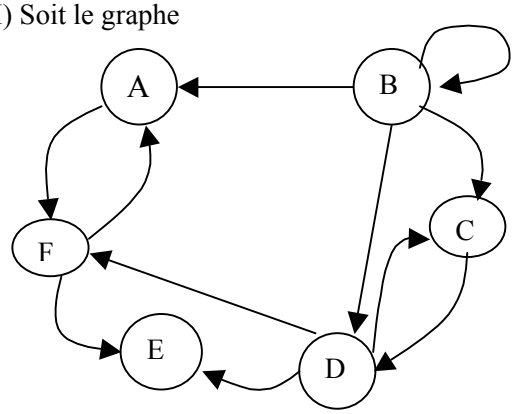
THEORIE DES GRAPHES

NOTIONS DE BASE

I) Soit le graphe



1) Donner $\Gamma^+(A), \Gamma^+(B), \Gamma^-(A), \Gamma^-(B)$.
2) Donner les demi-degrés intérieurs et extérieurs des sommets A et B. Donner les entrée(s) et les sortie(s) de G.
3) Donner un exemple de chemin simple mais non élémentaire.
4) Existe-t-il un circuit hamiltonien dans G ?
5) G est-il fortement connexe ? Justifiez votre réponse.

II)

Soit la matrice M binaire associé au graphe G :

	A	B	C	D	E	
	0	1	0	1	0	A
	0	0	1	0	0	B
	0	0	0	0	1	C
	0	0	1	0	1	D
	0	0	0	0	0	E

$= M$

a) Tracer le graphe représentatif de cette matrice.

b) Donner la matrice d'incidence de ce graphe.

c) Calculer M^2, M^3, M^4. Discuter la signification des coefficients non nuls de la matrice;

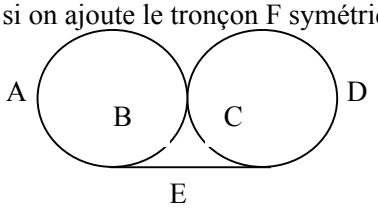
d) Calculez les puissances booléennes de M : $M^{[2]}$, $M^{[3]}$, $M^{[4]}$. Que pouvez vous en dire ?

e) Calculez ; $A = I \dot{+} M \dot{+} M^{[2]} \dot{+} M^{[3]} \dot{+} M^{[4]}$ (somme booléenne). Interpréter A.

III) Soit le schéma du circuit de train électrique de Laurent. Chaque aiguillage a deux positions possibles. Laurent a remarqué qu'au bout d'un certain temps, quelle que soit la position initiale du train, il n'emprunte jamais la partie E. (Le train n'utilise que la marche avant)

Pouvez-vous, à l'aide d'un graphe à 10 sommets, expliquez pourquoi?

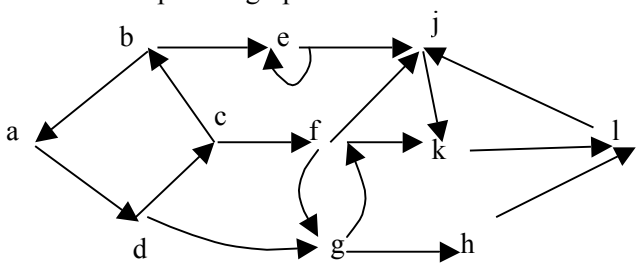
Qu'arrive-t-il si on ajoute le tronçon F symétrique de E ?



IV On appelle "graphe réduit" d'un graphe orienté G ayant m composantes fortement connexes : $C_1 \ldots C_m$; le graphe G' qui a m sommets $x_1 \ldots x_m$, tel qu'il existe un arc de x_i vers x_j dans G', s'il existe un sommet x dans C_i , un sommet y dans C_j et un arc (x, y) dans G.

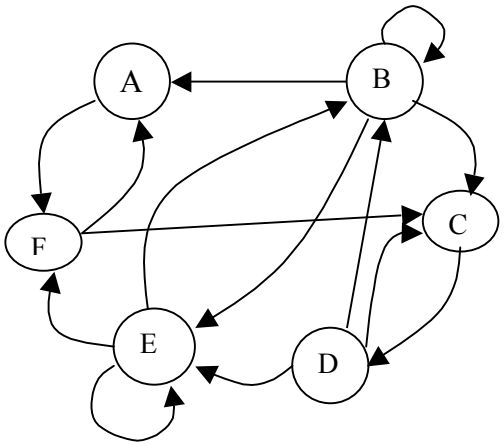
Montrer que le graphe réduit G' d'un graphe orienté G est sans circuit.

Déterminer G' pour le graphe orienté G suivant :



REPRESENTATION D'UN GRAPHE

Soit le graphe G



Représenter ce graphe sous forme :
a) matricielle
b) de liste d'adjacence

II) Soit le graphe G = (X, U) de n = 6 sommets (numérotés de I = 1 à I = 6) et de m = 7 arcs (numérotés comme ci-dessous de j = 1 à j = 7), valué par des coûts et défini par les 3 tableaux suivants :

a) Le tableau NARC (I) désigne le demi-degré extérieur du sommet I :

I	1	2	3	4	5	6
NARC(I)	1	2	1	1	2	0

Vous expliquerez pourquoi on a : $\sum_{i=1}^{n} \mathrm{NARC}(I) = m$

b) Le tableau EXTR(J) liste, dans l'ordre lexicographique, les successeurs du sommet 1, puis ceux du sommet2, etc…(s'ils existent) ; EXTR(J) désigne le numéro du sommet qui est l'extrémité terminale de l'arc d'indice J (J =1, 2,…,7)) : on a donc numéroté les arcs en commençant par ceux issus du sommet 1, puis du sommet 2 et ainsi de suite.

J	1	2	3	4	5	6	7
EXTR(J)	6	1	3	6	1	2	4

c) Le tableau COUT(J) désigne la capacité de l'arc d'indice J :

J	1	2	3	4	5	6	7
EXTR(J)	9	8	3	4	5	7	6

Tracer le graphe G, à grande échelle. Montrer qu'il comporte une entrée e et une sortie s.

I

Soit G = <S, A> un graphe de n sommets : S = (s_1, s_2,…,s_n)

Montrer que la somme des degrés de tous les sommets de G est un nombre pair.

Montrer que, pour tout graphe, le nombre de sommets de degré impair est un nombre pair.

II

Soit G un graphe connexe de n sommets.

Question 1 : Montrer que la plus courte chaîne reliant deux sommets est élémentaire. En déduire que la plus longue des plus courte chaînes de G, est de longueur au plus n-1.

Question 2 : Montrer que G a au moins n - 1 arêtes.

On dit que G est "k-régulier" si tout sommet est de degré k.

Question 3 : Montrer que si k est impair alors n est pair.

Un graphe connexe comportant n-1 arêtes est un "arbre" ($n \geq 2$).

Question 4 : Montrer que dans un arbre il y a au moins deux sommets de degré 1. En déduire l'ensemble des arbres k-réguliers.

Question 5 : Montrer qu'un arbre ne comporte pas de cycle.

Un graphe G est "biparti" s'il est possible de partionner l'ensemble de ses sommets en deux sous-ensembles tels que chaque arête de G n'ait pas ses deux extrémités dans un même sous-ensemble.

Question 6 : Montrer que les propositions suivantes sont équivalentes :

- G est biparti
- G n'a pas de cycle de longueur impaire.

Question 7 : En déduire qu'un arbre est biparti

Soit G un orienté graphe connexe de n sommets.

Question 8 : Montrer que si G est sans circuit, il existe un sommet sans prédécesseur ("entrée").

Question 9 : Montrer que la propriété de la question 8 peut-être fausse pour des graphes sans circuit, comportant un nombre infini de sommets, en exhibant un contre-exemple simple.

III

1) On rappelle les définitions suivantes :
- un sommet r d'un graphe H = (X,U) est une "racine" s'il existe dans H un chemin joignant r à x pour tout x ∈ X.
- Un graphe H = (X,U) est une arborescence de racine r si r est racine de H et si H est un arbre.

Démontrez que tout sommet x d'une arborescence, différent de la racine, possède un prédécesseur unique noté père (x).

2) 2) Soit A un alphabet de n lettres choisies parmi les 26 possibles ;
ex : A ={A, C, E}.
On considère le graphe complet non orienté K_n dont l'ensemble ses sommets est A

Soit un graphe partiel de K_n qui est un arbre. On transforme cet arbre en arborescence en choisissant une racine (puis en orientant les arêtes de manière convenable).

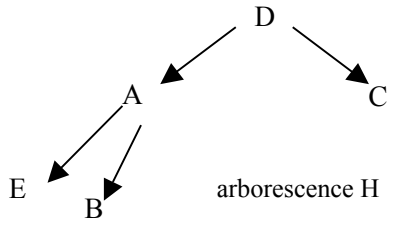
A cette arborescence, notée H, on associe un "mot" : m (H) sur l'alphabet A de la manière suivante :
a) Si H se réduit à un seul sommet, m(II) = ∅ (le mot est vide) ;

b) Sinon, examiner **dans l'ordre alphabétique** les sommets pendants de H et à chacun de ces sommets pendants associer la lettre associée à son prédécesseur appelé "père". On obtient ainsi un mot m' ayant autant de lettres que H a de sommets pendants.

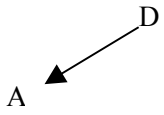
Soit H' l'arborescence obtenue à partir de H en supprimant les sommets pendants. On pose :

$$m(H) = m' \oplus m(H')$$

où ⊕ signifie "mise bout à bout" ou "concaténation".
Exemple : à l'arborescence représentée ci-dessous on associe le mot ADAD avec A = (A, B, C, D, E) . En effet la liste des sommets pendants de H est, en ordre alphabétique : B, C, E . B a pour pére A ; pour C c'est D et pour E c'est A ; ainsi m' = ADA. Supprimons B, C et E : on obtient l'arborescence H' et : m(H') = D ; m(H) = ADA ⊕ m(H'). Donc m(H) = ADAD

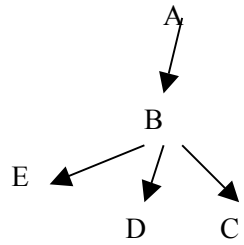


arborescence H



arborescent H'

Questions.
2.1) Si A = (A, B, C, D, E) trouver le mot associé à l'arborescence H représentée ci-contre :

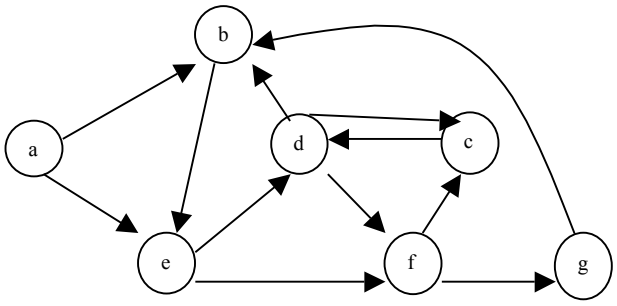


2.2) En utilisant le résultat de 1 déterminer la longueur du mot associé à toute arborescence H de n sommets déduite de K_n.
2.3) Proposer une méthode permettant de construire une arborescence de n sommets à partir d'un mot de (n-1) lettres (on considèrera les sommets pendants à chaque étape).
Rédiger avec soin cette méthode.
Application : Retrouver l'arborescence associée au mot FADDA déduit de
A = (A, B, C, D, D, F).
2.4) Facultatif Montrer que le nombre de graphes partiels du graphe non orienté complet sur n sommets qui sont des arbres est n^{n-2} (formule de Cayley).

PARCOURS DANS LES GRAPHES

I

I Représenter le graphe orienté ci-contre :
a) par une matrice associée.
b) Par une liste d'adjacence chaînée



a) On rappelle la procédure d'exploration « en profondeur d'abord » d'un graphe orienté, sous forme récursive :

```
Procédure rp (s : sommet) ; var t : sommet ;
début
état(s) : = exploré ;
Pour chaque sommet t dans L [s] faire
Si état (t) = inexploré alors
rp (t) ;
fin
```

NB : L(s) désigne ici l'ensemble des successeurs du sommet s.

L'appliquer au graphe ci-dessus.
Pour la forêt d'exploration obtenue précisez :

- les arcs d'arborescence
- les arcs directs
- les arcs rétrogrades
- les arcs traversiers

b) comment détecter la présence de circuits par un parcours « en profondeur d'abord » d'un graphe ?

II

On appelle "clique" d'un graphe non orienté G, tout sous-graphe complet de G. Montrer que s'il existe une clique C dans G et si l'on effectue un parcours en profondeur à partir d'un sommet s choisi arbitrairement dans C, lors de ce parcours tous les sommets de la clique apparaissent sur une même branche de l'arborescence associée au parcours. Sont-ils nécessairement consécutifs sur branche ?

III

On rappelle la procédure tri topologique inversé :

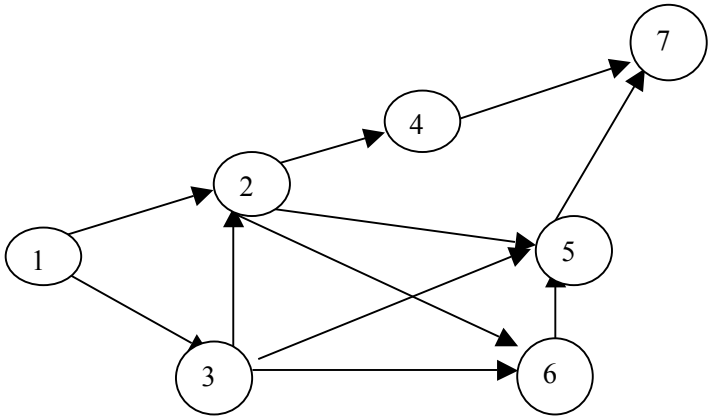
```
Procédure tritop( s : sommet) ;
/*tritop(s) affiche les sommets accessibles depuis s, par ordre topologique inversé*/
var t : sommet ;
début
état(s) :=exploré ;
pour chaque sommet t dans L(s) faire
si état (t) = inexploré alors tritop(t) ;
écrire (tritop)
fin
```

NB : L(s) désigne, ici, l'ensemble des sommets adjacents (voisins) du sommet s.

De quelle procédure générale est-elle issue ?

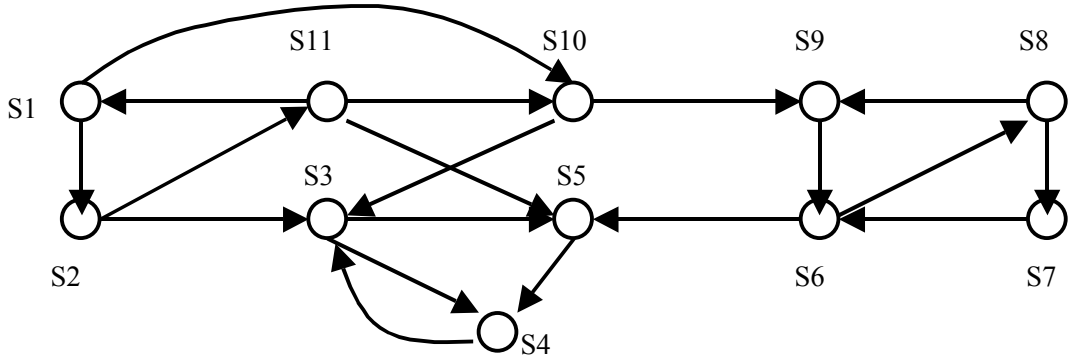
Rappelez la condition à laquelle le résultat de cette procédure est un tri topologique inversé

Appliquez la procédure au graphe ci-dessous, après avoir vérifié que cette condition est satisfaite.



IV

1°) Trouver les composantes fortement connexes du graphe ci-dessous en appliquant l'algorithme présenté en cours fondé sur le parcours en profondeur



2°) Combien y a-t-il de composantes fortement connexes dans un graphe orienté, sans circuit, de n sommets ?

Même question avec un graphe orienté de n sommets comportant un circuit élémentaire de longueur n.

3°) Dans le cas où tous les arcs d'un graphe orienté sont valués par un coût identique (égal à 1 par exemple), montrer que l'algorithme de parcours en largeur effectué à partir d'un sommet x, permet d'obtenir un plus court chemin de x vers tout autre sommet accessible depuis x.

EVALUATION DE COMPLEXITE D'ALGORITHMES

I Pour les deux algorithmes suivants, calculer la complexité dans le pire des cas, dans le meilleur des cas et en moyenne.

1) Soient A = (aij) et B = (bij) deux matrices n x n à coefficients dans R. L'algorithme suivant calcule les coefficients cij de la matrice C, où C = A x B, selon la formule classique :

$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$ pour i et j compris entre 1 et n.

```
type matrice : array [1…n, 1…n] of integer ;
procedure MULTMAT (a, b : matrice ; var c : matrice );
var i, j, k : integer ;
begin
        for i : = 1 to n  do
                for j : = 1 to n  do
                        begin
                                c(i,j) : = 0 ;
                                for j: = 1 to n  do
                                        c(i, j) : = c(i, j)  + a(i, k).b(k, j) ;
                        end
end MULTMAT
```

2) Algorithme de recherche séquentielle d'un élément X dans un tableau L.

```
var L : array (1…n] of element
        X : element ;
        j : integer ;
begin
        j : = 1 ;
        while j ≤ n and L (j) ≠X
                do j : = j + 1 ;
        if j > n then print "X not in L"
        else print "X is in L"
end
```

3) programme rudimentaire de tri d'un tableau Tab à n éléments (par transpositions successives).

```
Pour i allant de 1 à (n – 1)
        pour j allant de (i + 1 ) à n
                si Tab(i) < Tab(j) alors
                        élt := Tab(i)
                        Tab(i) := Tab(j)
                        Tab(j) := élt
                        fin si
                fin si
        fin pour (j)
fin pour (i)
```

FERMETURE REFLEXO-TRANSITIVE : ALGORITHME DE ROY-WARSHALL

Le problème, étant donné un graphe G = (X, U) de n sommets (indicés de 1 à n), est de déterminer la fermeture réflexo-transitive de G : le graphe $\hat{G} = (X, \hat{U})$; rappelons que (x, y) est un arc $\hat{G}$ de si et seulement si x = y ou bien s'il existe un chemin de x vers y dans G.
Soit M = $[m_{ij}]$ la matrice binaire associée à G (et $\hat{M}$ pour $\hat{G}$) .

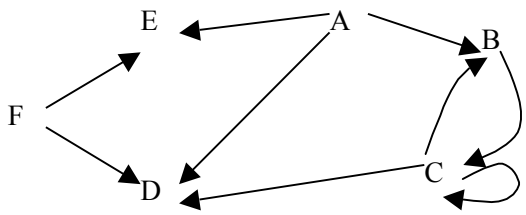
On rend G réflexif [1] par ajout d'une boucle en tout sommet n'en comportant pas :
on suppose donc désormais que $m_{ii} = 1$ pour tout i = 1, 2,…,n.

Considérons l'opérateur $è_r$ qui, appliqué à G, fournit le graphe $è_r(G)$: par définition $è_r(G)$ comporte les arcs de U ainsi que tout arc (i, j) (si celui-ci n'appartenait pas à G) tel qu'existent dans G les arcs (i,r) et (r, j). Autrement dit, $è_r(G)$ enrichit G des arcs (i,j) (si ceux-ci n'y existaient pas) tels q'il existe un chemin de longueur deux de i vers j passant par r.

L'algorithme de ROY-Warshall consiste) :

-appliquer $è_1$ à G , puis $è_2$ à $è_1(G)$, puis $è_3$ à $è_2(è_1(G))$, …
puis $è_n$ à $è_{n-1}(è_{n-2\ldots1}(G))$; le dernier graphe obtenu est $\hat{G}$

1. Appliquer cet algorithme au graphe G ci-dessous, après avoir établi M.



2. <u>Preuve de l'algorithme</u>

2.1. Soit $M^{(r)}$ la matrice binaire associée à $è_r(G)$: $M^{(r)} = [m_{ij}^{(r)}]$.

Montrer que :

$$m_{ij}^{(r)} = m_{ij} + m_{ir} . m_{rj}.$$

2.2 Montrer que les opérateurs $è_r$ et $è_s$ $(1 \le s \le n)$ commutent ; on notera $m_{ij}^{(sor)}$ l'élément générique de la matrice binaire associée à $è_s(è_r(G))$.

2.3. Montrer que l'opérateur $è_r$ est "idempotent" : $è_r(è_r(M) = è_r(M)$

2.4. Montrer que la composition des opérateurs considérés est associative.

2.5. Prouver que $è = è_1 \_ è_2 \_ \_ è_n$ a au moins pour effet d'ajouter à G les arcs (i, j) (n'existant pas dans G) tels qu'il existe un chemin de longueur 2 de i vers j dans G ; que peut-on en déduire pour $è^2 = è \_ è$?

Prouver alors que $è^{n-2}(G) = \hat{G}$

2.6. En déduire que $è(G) = \hat{G}$

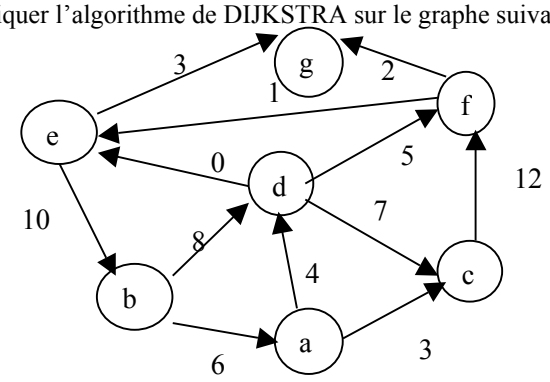
2.7. Evaluer la complexité de cet algorithme.

[1] Pour calculer la fermeture transitive de G, on omettrait ce point.

CHEMINS OPTIMAUX

I

1°) a) Appliquer l'algorithme de DIJKSTRA sur le graphe suivant, en choisissant le sommet a comme sommet initial :



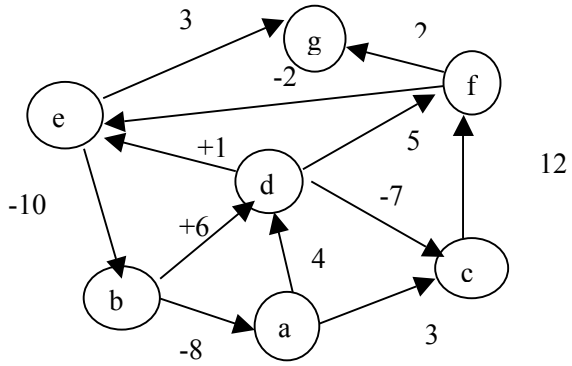
2°) Dans certains problèmes de communication, on a besoin de connaître le point (ou les points) le moins éloigné de tous les autres points d'un réseau . On note par d(x, y) la valeur du plus court chemine allant de x à y. On appelle "excentricité" d'un sommet x, dans un graphe G = (X,U,), orienté et valué par des coûts positifs la quantité :

$$exc(x) = \max_{y \in X} \{ppc(x,y)\}$$

exc(x) est donc la distance maximale nécessaire pour atteindre chaque autre sommet depuis le sommet x. On appelle "centre" du graphe G, un sommet d'excentricité minimale.

a) Montrer comment on peut trouver le centre d'un graphe en utilisant l'algorithme FLOYD.
b) Trouver le centre du graphe donné dans l'exercice n°1

3°) En utilisant l'Algorithme de FORD, trouver le chemin de valeur <u>maximale</u> allant du sommet a vers tout autre sommet dans le graphe suivant :



II

UNE PROCEDURE DE ROUTAGE DANS UN RESEAU D'ORDINATEURS A COMMUTATION DE PAQUETS

Dans un réseau à commutation de paquets, il doit exister une règle (parfois abusivement appelée "algorithme") qui indique, en chaque nœud, le canal sur lequel un paquet (qui vient d'arriver en ce noeudmais qui n'est pas encore parvenu à destination) doit être acheminé, connaissant, bien sûr, cette destination ; pour simplifier, on a supposé ici que le routage d'un paquet n'est pas aléatoire (pour aller du nœud i au nœud j l'itinéraire du paquet est donc toujours le même) et l'on exclut les défaillances du réseau (et donc les routage adaptatifs).

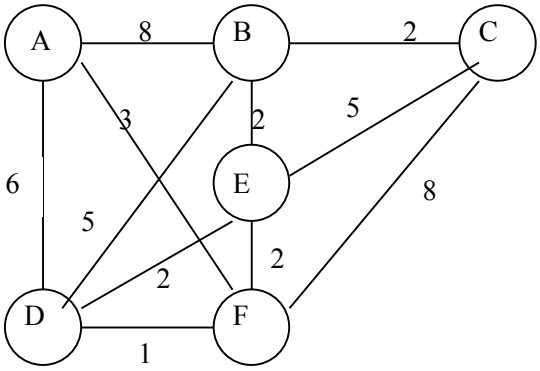
Le problème est donc -ici- pour toute paire de nœud, d'établir un chemin, composé d'un ou plusieurs canaux, permettant d'acheminer les paquets.

Il existe, en général, plusieurs façons de classer les différentes procédures de routage ; la première consiste à distinguer : les procédures déterminées, fixées à l'avance (comme ici), les procédures aléatoires, ou encore celles adaptatives. Il n'est pas commode de comparer les règles de routage entre elles, à cause du grand nombre de critères susceptibles d'être retenus pour juger des performances ; ainsi :

- le temps moyen de transmission
- un temps de réponse optimal pour des trafics variables,
- la façon dont la règle s'adapte en cas de défaillance du réseau,
- le coût des calculs et le temps de retransmission des informations.

On s'intéresse ci-dessous uniquement à une procédure directive : "l'acheminement de poids minimal" : on va affecter à chaque canal un coefficient spécial ou "poids" tenant compte du coût, de la longueur physique du temps de propagation, du trafic prévu, du nombre d'erreurs détectées.

1) l'exemple ci-dessous donne la topographie d'un réseau à six nœuds où chaque canal a été affecté d'un poids (ou valeur) ; les canaux sont bidirectionnels (avec le même poids dans chaque sens)



Après avoir remplacé chaque arête par une paire d'arcs, déterminer par application de l'algorithme de DANTZIG les chemins de valeur minimale du nœud A vers les autres nœuds.

2) En fait, on a besoin de rechercher ces chemins entre tout couple de nœuds.

Quelle(s) méthode(s) envisagez-vous d'appliquer ? (leur application est facultative).

3) Montrez qu'on peut supprimer cinq canaux avant de procéder à cette recherche :; on pourra justifier cette suppression à l'aide du principe d'optimalité de la programmation dynamique.

Après ces suppressions, le graphe partiel obtenu étant très simple, vous pourrez déterminer visuellement les chemins cherchés.

Etablissez la matrice des poids des chemins optimaux ; vérifiez qu'elle est symétrique.

4) Etablir la table de routage $[x_{ij}]$ où x_{ij} désigne pour un paquet passant en i le prochain nœud à atteindre, sachant que sa destination est j et qu'il suit le chemin de poids minimal de i vers j. Remarquez que cette matrice n'est pas symétrique. Commentez l'usage de cette table pour un paquet transitant du nœud A au nœud C

	1	2	3	4	5	6	
1	-						A
2		-					B
3			-				C
4				-			D
5					-		E
6						-	F
	A	B	C	D	E	F	

III

Dans un graphe G = (X, A), on associe à tout arc (i, j) de A une "fiabilité" rij c'est à dire une valeur comprise entre 0 et 1 : $0 < r_{ij} \leq 1$.

La fiabilité mesure la probabilité que l'arc soit opérationnel (c'est-à-dire non défaillant).

On définit la fiabilité d'un chemin P du graphe G, comme le produit des fiabilités des arcs qui forment ce chemin :

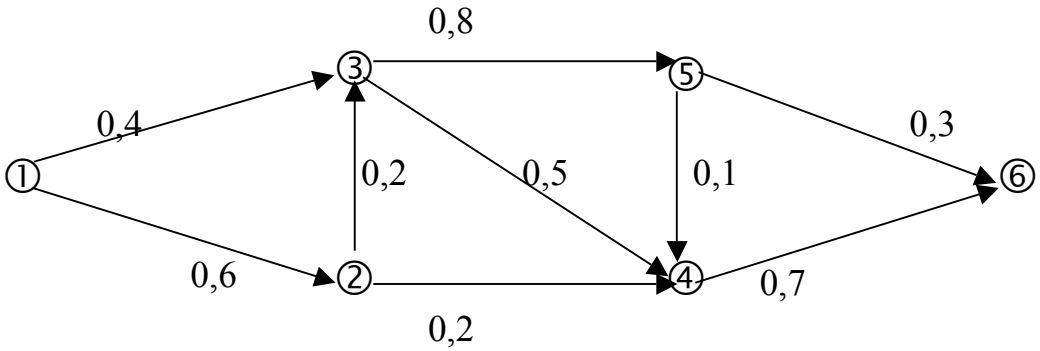
$$r(P) = \prod_{(i,j) \in P} r_{ij}$$

Le problème est de déterminer le (ou les) chemin(s) de fiabilité maximale issus d'un sommet source s vers tout autre sommet du graphe.

1) Montrer qu'en employant les logarithmes, on peut ramener le problème du chemin de fiabilité maximale à celui du plus court chemin. On rappelle que pour $0 < r_{ij} \leq 1$, on a : $\log r_{ij} \leq 0$.

2) Supposez que vous ne soyez par autorisé à utiliser les logarithmes (parce qu'ils peuvent conduire à des nombres irrationnels). Quelles sont alors les modifications que l'on doit apporter à l'algorithme de DIJKSTRA pour qu'il puisse résoudre ce problème ?

3) Appliquer alors l'algorithme de DIJKSTRA modifié sur le graphe ci-dessous :



4) Si certains des coefficients rij étaient strictement plus grand que 1 (ce qui est impossible si r_{ij} est une fiabilité), la méthode proposée en 3) reste-t-elle valable ?

<u>Rappel</u>

On rappelle l'algorithme de DISKSTRA vu en cours (pb des plus courts chemins issus du sommet s dans le graphe G = (X, A). Notations : cij est la valuation de l'arc (i, j) ; si (i, j) $\in$ A, alors i $\in$ préd(j) et j $\in$ succ(i). Les sommets sont indicés de 1 à n.

Algorithme DIJKSTRA

début

$S := \emptyset$ et $\overline{S} := X$;

$d(s) := 0$ et préd(s) := 0;

d(i) := pour chaque sommet $i \in X,\ i \neq s$;

tant que card S < n faire

début

Soit $i \in \overline{S}$ tel que $d(i) = \min_j \{d(j) : j \in \overline{S}\}$;

$S := S \cup \{i\}$;

$\overline{S} := S - \{i\}$;

pour chaque j $\in$ succ (i) faire

si $d(j) > d(j) + c_{ij}$ alors $d(j) := d(i) + c_{ij}$ et préd (j) := i ;

fin

fin

NB : en fin d'exécution de l'algorithme, préd(j) désigne le précesseur du sommet j sur le chemin optimal de s à j obtenu.

ORDONNANCEMENT

Problème 1

On s'intéresse à l'ordonnancement de la production de deux éléments A et B constitutifs d'une plateforme off-shore.

Chacun de ces éléments doit être usiné successivement sur les deux machines M1 puis M2.

On estime les temps d'exécution (en jour) sur chaque machine à :

	M_1	M_2
A	2	3
B	2	1

Avant de procéder à la fabrication de chaque élément sur une machine, une phase de réglage est nécessaire.

Les temps de réglage en jours sont donnés ci-dessous :

	M_1	M_2
A	1	1
B	2	6

Ainsi, par exemple, la durée fabrication de B sur M1 est de 2 jours, sachant que M1 a subi préalablement un réglage de 2 jours.

On suppose qu'il est possible d'utiliser ou de régler les deux machines simultanément. Pour des raisons techniques impératives, l'élément A doit être réalisé avant B sur chaque machine.

1. Préciser sous forme de tableau les huit tâches (opérations élémentaires) qu'il faudra exécuter, leur contraintes de précédence ainsi que leur durée respective.

On notera (i, j, k) chacune de ces tâches , où :

i _ {R. U} (R : réglage U : usinage)

j _ {1, 2} (1 : machine M1, 2:M2)

k _ {A,B}

2. Tracer un graphe PERT associé à ce problème.

   Il sera nécessaire d'introduire au moins une tâche fictive;

3. Déterminer la durée minimale nécessaire pour produire les deux éléments.
4. Quelles sont les opérations (tâches) sur lesquelles il vous semble pertinent d'exercer un contrôle particulier si l'on souhaite réaliser ce projet au plus vite.

Problème 2

On rappelle que le graphe Potentiel-Etapes (graphe Pert) associé à un problème d'ordonnancement est un graphe sans circuit G = (X, U) dans lequel :

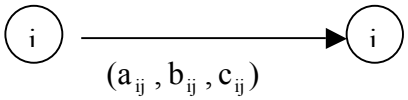
- l'ensemble X des sommets correspond aux étapes (événements) de réalisation du projet :
l'ensemble U des arcs correspond aux différentes tâches. Chaque tâche est désignée par le couple (i, j) des sommets respectivement origine (étape début) et extrémité (étape fin) de l'arc u correspondant. Ces tâches peuvent être des tâches réelles du projet ou bien des tâches fictives.

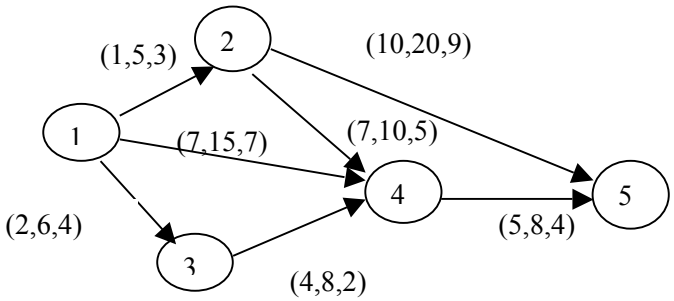
On s'intéresse ici à une extension des problèmes d'ordonnancement dans lesquels au lieu d'avoir des tâches de durée fixe et connue, la durée dij de chacune des tâches (i,j) peut varier continûment entre deux limites :

$$a_{ij} \leq d_{ij} \leq b_{ij}$$

où a_{ij} est la durée d'exécution de la tâche (i, j) en "urgence" et b_{ij} est la durée d'exécution de la tâche (i, j) "normale".
A chaque tâche (i, j) est associée un coût unitaire cij dont la signification est la suivante ; cij est l'augmentation du coût de réalisation du projet (surcoût) pour chaque unité de temps gagnée sur la réalisation de la tâche (i,j) considérée par rapport à sa durée normale b_{ij}.
On remarquera que pour chaque tâche (i,j) le surcoût est nul pour une durée de réalisation de la tâche égal à la durée normale b_{ij}. Par suite, la contribution d'une tâche quelconque u = (i, j) à l'augmentation du coût total C du projet est égale à $c_{ij} . (b_{ij} – d_{ij})$ pour $a_{ij} \leq d_{ij} \leq b_{ij}$.

Dans la suite de l'exercice, on considère le graphe Potentiel-Etapes suivant :





- L'étape 1 correspond au début de l'ensemble du projet.
- L'étape 5 correspond à la fin de l'ensemble du projet.

N° de la tâche :	1	2	3	4	5	6	7
(i;j)	(1.2)	(1.4)	(1.3)	(2.5)	(2.4)	(3.4)	(4.5)

Ainsi, par exemple, si la tâche 3 est réalisée en 6 unités de temps, le surcoût est nul. Si elle est réalisée en 5 unités de temps, le surcoût est de 4, si elle est réalisée en 4 unités de temps, le surcoût est de 8, etc…

Question 1 : Sur l'exemple ci-dessus, et en supposant que l'on soit prêt à payer tout les surcoûts nécessaires, déterminer la durée minimale θ_{min} de réalisation du projet. Donner le graphe PERT du problème d'ordonnancement classique auquel on est confronté dans ce cas et expliquer la procédure employée pour déterminer le chemin critique et la durée θ_{min} de ce chemin critique.

Question 2 : Toujours dans l'exemple, et en supposant que l'on impose au projet d'être réalisé dans la durée θ_{min} déterminer à la question 1. Donner la durée de réalisation de chacune des tâches permettant une réalisation minimisant le surcoût total.

Question 3 : En supposant maintenant que l'on n'est prêt à payer aucun surcoût, quelle serait la durée (notée θ_{max}) de réalisation du projet ? Expliciter le chemin critique correspondant.

Question 4 : En pratique, ce à quoi l'on s'intéresse, c'est de voir comment évolue le surcoût total en fonction de θ (temps total de réalisation du projet) pour θ variant entre θ_{min} et θ_{max}. On peut représenter le problème de la détermination du surcoût minimal pour un achèvement du projet en θ unités de temps, sous la forme d'un "programme linéaire" ($\theta_{min} \leq \theta \leq \theta_{max}$).
On considère alors le problème général sur un graphe G = (X, U) quelconque, sans circuit, et on utilise les notations définies ci-dessous pour les variables du problème :

dij = durée de la tâche correspondant à l'arc (i, j) :
$a_{ij} \leq d_{ij} \leq b_{ij}$
π_i = instant de réalisation de l'étape i associée au sommet du graphe numéroté i : en particulier π_D est l'instant de début et π_F est l'instant de fin de l'ensemble du projet.

La formulation du programme linéaire correspondant à ce problème se ramène alors :

Maximiser $Z = \sum_{(i,j) \in U} c_{il}\, d_{ij}$

sous les contraintes pour tout $(i,j) \in U$ et pour tout $i \in X$:

$$d_{ij} \geq 0$$
$$\pi_i \geq 0$$
$$d_{ij} + \pi_i - \pi_j \geq 0 \quad \text{(type 1)}$$
$$\pi_F - \pi_D = \theta \quad \text{(type 2)}$$
$$d_{ij} \leq c_{ij}$$
$$-d_{ij} \leq -a_{ij}$$

Expliquer :

- La forme de la fonction économique Z et son lien avec la fonction économique du problème initialement posé qui consistait à minimiser le surcoût global.
- La raison d'être des contraintes, en particulier celles de type 1 et leur lien avec les problèmes classique d'ordonnancement et celle du type 2.

Problème 3

Un contrat a été signé pour un projet informatique à réaliser dans une durés maximale de 13 mois. Ce projet a été décomposé en tâches confiées à des ingénieurs. Ces tâches ont entre elles des contraintes de précédence représentées par le tableau ci-dessous. Ce tableau indique également la durée prévue des tâches ainsi que le nombre des ingénieurs qui doivent être affectés à chaque tâche.

Tâches	Durée en mois	Tâches préalables	nombres d'ingénieurs
A	3	--	8
B	8	A	4
C	5	A	4
D	1	C, G	2
E	4	D	6
F	2	C	6
G	6	--	4

1) En faisant abstraction du nombre d'ingénieurs disponibles :

1.1) Donnez la durée minimale du projet dans le meilleur des cas en appliquant la méthode du graphe potentiel-tâches (méthode MPM de B. ROY) directement sur le graphe.

Indiquez les dates de début au plus tôt et au plus tard de chacune des tâches.

Précisez les tâches critiques.

1.2) Calculez les marges totales et marges libres des tâches B, C et G.

2) En réalité, la société ne dispose que de 12 ingénieurs pour ce projet :

2.1) En appliquant la méthode "sérielle", dans laquelle on donne la priorité à la tâche disponible de plus petite date au plus tard, peut-on obtenir une solution qui respecte le délai du contrat (13 mois) ? Illustrer la solution par un diagramme de GANTT (en abscisse le temps, en ordonnée le nombre d'ingénieurs).

Résumez clairement la solution donnée par la méthode sérielle en indiquant pour chaque tâche sa date de début.

2.2) Donnez la solution obtenue en appliquant la méthode sérielle uniquement sur les tâches non critiques, après avoir placé les tâches critiques de telle sorte qu'elles débutent à leur date au plus tôt.

Comparez la solution ainsi obtenue à la précédente. Que peut-on en conclure ?

I

Soit le graphe G = (X, U) de n = 5 sommets et m = 7 arcs, valué par des "capacités", défini par les tableaux suivants :
a) Le tableau DEG (I) désigne le demi-degré extérieur du sommet I (nombre d'arcs ayant I comme extrémité initiale) :

I	1	2	3	4	5
DEG(I)	1	2	2	0	2

Rappeler pourquoi $\sum_{i=1}^{n} DEG(I) = m$

b) Le tableau SUCC(J) liste d'abord les successeurs (s'ils existent) du sommet 1, puis ceux du sommet 2, et ainsi de suite.
SUCC(J) désigne l'indice du sommet qui est l'extrémité terminale de l'arc d'indice J (on a numéroté les arcs en commençant par ceux issus du sommet 1, puis ceux du sommet 2, etc)

J	1	2	3	4	5	6	7
SUCC(J)	4	1	4	2	5	1	2

c) Le tableau CAPA(J) désigne la capacité de l'arc d'indice J.

J	1	2	3	4	5	6	7
CAPA(J)	1	4	7	3	5	2	2

1. Tracer, très clairement, le graphe G (inutile de reporter les indices des arcs ; noter x1 le sommet d'indice 1, etc ; reporter sur chaque arc sa capacité notée entre crochets)..

Montrer <u>en détail</u> que G est un réseau de transport.

2. Vérifier que le tableau FLUX(J) ci-dessous définit bien un flot sur le réseau de transport G :

J	1	2	3	4	5	6	7
FLUX(J)	1	1	4	3	2	0	2

Ce flot est-il complet ? Optimal ? Sinon l'optimiser en mettant en évidence une chaîne améliorante. Donner la coupe optimale associée au flot maximal.

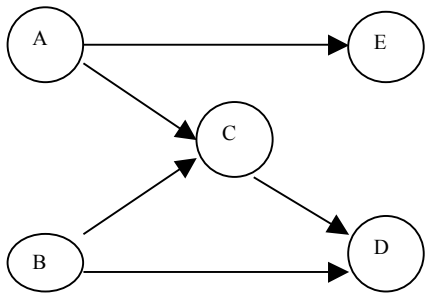
3. Donner le graphe d'écart associé au flot donné par le tableau FLUX(J) ; à quoi correspond la chaîne améliorante ?
De même donner le graphe d'écart associé au flot optimal ; à quoi, dans le graphe d'écart, correspond une coupe minimale dans le graphe G initial ?

II

Un transporteur veut expédier une certaine marchandise depuis les villes A et B vers les villes D et E.

Les liaisons (i, j) représentant les transports possibles entre villes sont données par les arcs du graphe ci-dessous (C est une ville intermédiaire).

Les quantités de marchandise disponibles en A et B sont respectivement de 7 et 12 unités. Les quantités demandées en D et E sont respectivement de 15 et 9 unités.



Les tableaux suivants : CAPA(i, j) et COUT (i,j) désignent respectivement les capacités de transport offertes sur ces liaisons (i, j), ainsi que leur coût unitaire de transport :

arc (i, j) :	(A, C)	(A, E)	(B, C)	(B, D)	(C, D)
CAPA (i, j) :	5	14	7	5	6
COUT (i, j)	1	4	3	2	0

1) Déterminer, en faisant abstraction des coûts de transport, quelle est la quantité maximale de marchandise que le transporteur peut expédier des villes A et B vers les villes D et E ?

Pour cela, on devra d'abord compléter le graphe ci-dessus et appliquez ensuite un algorithme approprié vu en cours.

Préciser en détail la solution optimale ainsi obtenue et justifier en détail son optimalité. On rendra **obligatoirement** un flot initial tel que le flux sur (A, C) soit égal à 2 et qui soit complet.

On pourra ultérieurement modifier, si nécessaire, le flux sur (A, C).

2) Vérifier que le tableau suivant FLUX(i, j) représente effectivement un flot □ sur le réseau de transport construit en (1).

arc (i, j) :	(A; C)	(A, E)	(B, C)	(B, D)	(C; D)
FLUX (i, j)	5	2	1	5	6

Donner la valeur v ainsi que le coût de ce flot □

Tracer le graphe d'écart associé au flot □ : G^e {□}

3)On admettra que ce flot □ de valeur v est bien de coût minimal parmi tous les flots de valeur v. En déduire alors le flot maximal (de valeur v*), de coût minimal, c'est à dire la quantité maximale de marchandise que le transporteur peut expédier, au moindre coût, des villes A et B vers les villes D et E , (On utilisera **obligatoirement** l'algorithme approprié vu en cours)

III
PLACEMENT DE PROCESSUS SUR LES PROCESSEURS D'UNE MACHINE PARALLELE.

Soit T : $\{t_1, t_2, …., t_n\}$ un ensemble de n processus (ou tâches) d'un programme parallèle.

Soit P = $\{p_1, p_2, …p_p\}$ un ensemble de p processeurs d'une architecture donnée;

Le placement du programme sur cette architecture consiste à placer chacune des n tâches dur l'un des p processeurs il y a alors p^n placements possibles (si toute tâche peut être placée sur tout processeur).

En pratique deux types de contraintes se présentent :

- le placement de certaines tâches sur un (ou des) processeur (s) donné(s) disposant de ressources particulières, peut être limité.
- on peut imposer des regroupements de tâches et/ou limiter les routages.

On considère ci-dessous le placement statique des tâches sur les processeurs ; il est réalisé à la compilation ou au chargement du programme;

Le coût d'un placement est la somme de 2 coûts :

Les coûts d'exécution des tâches placées : le coût d'exécution de la tâche t_j sur le processeur p_i, noté q_{ij}, et connu

et

Les coûts de communication entre tâches. Si deux tâches t_j et t_k sont exécutées par le même processeur, ce coût de communication est nul ; sinon ce coût vaut c_{jk}, qui est connu. Ce coût est symétrique : $c_{jk} = c_{kj}$.

Le problème est donc de trouver un placement, de coût minimal, des tâches sur les processeurs.

Cas de p = 2 processeurs : P = $\{p_1, p_2\}$

On va modéliser le problème par un graphe valué G = (X, U) défini comme suit :

Chaque processeur et chaque tâche sont représentés par un sommet : X = P ∪ T ; le sommet p_1 est une entrée de G, le sommet p_2 est une sortie de G.

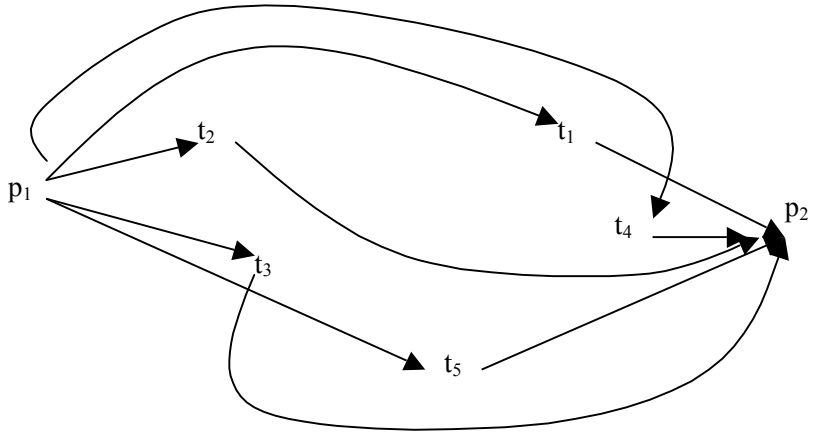
On figure un arc du sommet p_1 vers tout sommet t_j vers le sommet p_2 valué par q_{2j} (et non pas par q_{1j}). Symétriquement on figure un arc de tout sommet tj vers le sommet p_2 valué par q_{1j} (et non pas par q_{2j}), ceci pour j = 1, 2, …, n.

On figure aussi entre deux sommets quelconques distincts t_j et t_k , un arc (t_j, t_k) et un arc $(t_k, t_j,)$ si c_{jk} est non nul (et ces deux arcs sont valués par c_{jk}) : si c_{jk} est nul, on ne figure aucun arc entre t_j et t_h (j, k = 1, 2, …, n et $j \neq k$).

1) pour l'exemple ci-dessous (p = 2 processeurs, n = 5 tâches), construire et valuer le graphe ainsi défini, en complétant le tracé ci-dessous, que vous reproduirez à grande échelle.

	t_1	t_2	t_3	t_4	t_5
p_1	2	3	5	1	7
p_2	4	3	2	3	3

	t_1	t_2	t_3	t_4	t_5
t_1	-	2	0	1	0
t_2	2	-	5	0	0
t_3	0	5	-	0	1
t_4	1	0	0	-	1
t_5	0	0	1	1	-



On attribue à chaque arc une capacité égale à son coût : montrer que G est un réseau de transport.

2).a Soit la coupe séparant l'ensemble des sommets $\{p_1, t_2, t_1\}$ des sommets $\{t_3, t_4, t_5, p_2\}$.

Quelle est la capacité de cette coupe ?

On lui associe le placement suivant : t_1 et t_2 exécutés sur p_1 ; t_3, t4 et t_5 sur p_2. Quel est le coût de placement et la capacité de la coupe ? Que remarquez-vous ?

2)b) Réciproquement à un placement donné, associer une coupe G. Quelle est la relation entre le coût de placement et la capacité de la coupe ? Justifiez.

En déduire qu'il y a bijection entre les coupes et les placements.

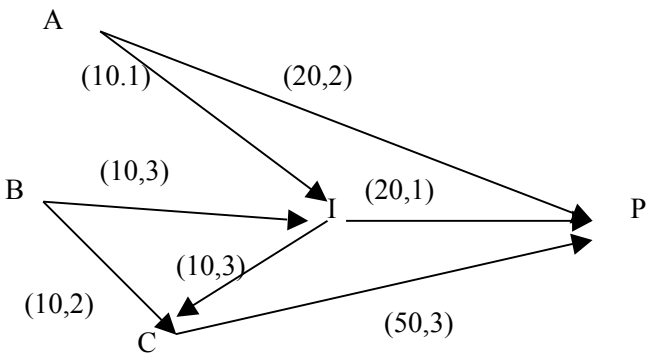
3) Pour l'exemple ci-dessus, déterminer par l'algorithme approprié vu en cours, la coupe minimale ; tracer celle-ci, en déduire le placement optimal.

4) Justifiez la polynormialité du problème avec p = 2 processeurs (pour $p \geq 3$, on montre que le problème est NP-difficile).

IV

Un transporteur est chargé d'organiser en urgence un convoi des combustibles irradiés depuis trois centrales nucléaires A, B et C vers l'usine de retraitement P.(I est un nœud intermédiaire).

Les liaisons ci-dessous représentent les transports possibles entre ces sites ; elles sont données par les arcs du graphe G ci-dessous. Chaque arc (x, y), sera valué par (c_{xy}, d_{xy}) : c_{xy} indiquant la capacité de la liaison (x, y) et d_{xy} le coût unitaire de transport associé (en euros). On désignera par f_{xy}, le flux sur l'arc (x, y).



Les quantités de combustibles irradiés déposés dans les piscines des centrale A,B et C sont respectivement de 30, 50 et 40 unités. La capacité de l'usine de retraitement de ces déchets est de 100 unités (on ne dispose pas de centre de stockage intermédiaire).

1) Déterminer, en faisant abstraction des coûts de transport, le flot maximal (f*) de déchets que le transporteur peut acheminer des trois centrales nucléaires A, B et C vers l'usine de retraitement P en respectant l'ensemble des contraintes de la situation.

Pour cela, vous devrez **compléter** le graphe ci-dessus puis vous constituerez un flot initial dans lequel $f_{BI} = f_{IC} = 10$ unités et $f_{BC} = 0$ unité.
Marquez d'un trait double les arcs saturés.

Appliquez ensuite un algorithme approprié vu en cours. Précisez en détail la solution optimale (f*) ainsi obtenue et justifier son optimalité puis calculer le coût de transport (en euros) associé à ce flot f*.
2) En se basant sur la coupe minimale déduite de la question 1, en déduire quel serait le meilleur investissement, pour améliorer la quantité qu'on peut faire parvenir à l'usine de retraitement.

3) Tracer $G_e(f^*)$: le graphe d'écart associé au flot associé au flot maximal (f*) obtenu à la question 1. Ce flot f* est-il alors de coût minimal ? Justifier votre réponse en raisonnant sur $G_e(f^*)$. On rappelle que f* est de coût minimal si et seulement si G_e (f*) ne comporte pas de circuit de valeur négative.

V

Soit un réseau R de transmission à créer : on connaît les nœuds, les liaisons et leur capacité (une unité de capacité correspondant à 300 bauds) ; on connaît également le coût unitaire de transmission sur les différentes liaisons.

arc	(a,c)	(a,d)	(b,c)	(c,d)	(c,e)	(d,T)	(c, T)
Capacité disponible (par unité de 300 bauds)	3	2	2	2	2	6	5
Coût de transmission unitaire	2	1	2	3	4	3	3

a et b sont des nœuds d'entrée de l'information dans le réseau : c, d et e sont des noeuds intermédiaires : T est un nœud de réception de l'information.

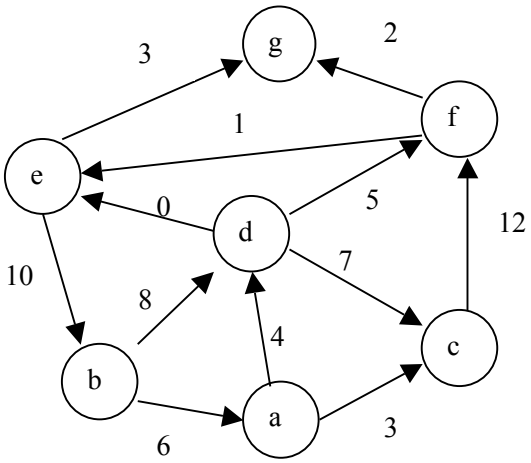
1) Représenter le réseau R à l'aide d'un graphe en précisant pour chaque arc sa double valuation : (capacité, coût)

2) On souhaite déterminer le débit maximal du réseau, défini comme la quantité d'information reçue par seconde au nœud T, ceci tout en minimisant le coût global de transmission;

Compléter le graphe et appliquer un algorithme approprié pour répondre à cet objectif.

ARBRES DE POIDS MINIMUM

1) Donner par les algorithmes de KRUSKAL, PRIM et SOLLIN, un arbre couvrant de valeur minimale du graphe suivant :



2) Une banque désire installer un réseau de transmission de données entre ce centre situé à ANTIBES (SOFIA ANTIPOLIS) et sept de ses succursales. Il s'agit d'un réseau arborescent composé de lignes privées point à point à 400 bauds avec des possibilités de concentrateurs. Le coût de construction d'un ligne entre 2 villes est donné par le tableau suivant :

VILLES	**ANTIBES**	**PARIS**	**LYON**	**MARSEILLE**	**BORDEAUX**	**LILLE**	**RENNES**	**STRASBOURG**
ANTIBES	-	923	471	188	808	1151	1246	960
PARIS		-	462	777	560	219	147	488
LYON			-	315	548	679	775	489
MARSEILLE				-	648	995	1090	804
BORDEAUX					-	777	439	1040
LILLE						-	564	524
RENNES							-	827
STRASBOURG								-

Les coûts ont été déterminés en fonction des distances inter villes et du chiffre d'affaires de chaque succursale).

1) Reconnaître la nature du problème posé par la recherche du réseau coût minimal.

2) Résoudre ce problème par l'algorithme de PRIM.

RESEAUX DE PETRI

On considère un système avec attente (attente à un disque par exemple) pour lequel :

a) La population potentielle (nombre de programmes pouvant utiliser le système) est finie et égale à 5 :
b) La taille maximale de ce système avec attente (nombre de tampons en attente ou en cours d'accès) est finie et égale à 3

-A-

On désire modéliser ce système à l'aide d'un réseau de Petri. Ce réseau comportera trois places. La marque de la première (p_1) représente le nombre de clients à l'extérieur du système : la marque de la seconde (p_2) le nombre de clients en attente ou en cours de service ; la marque de la troisième (p_3) le nombre de places encore disponibles dans le système.

On définit deux transitions de la façon suivante (ici le mot « place » signifie place dans la file d'attente du système et non place du Réseau de Petri) :

T 1 s'il y a au moins une place libre dans le système et un client au moins à l'extérieur du système.
Alors faire entrer un client dans le système ; il occupera une place.

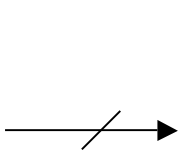
T 2 s'il y a au moins un client dans le système
Alors ce client peut sortir du système et donc libérer une place.

A1. Tracer le réseau de Petri associé à ce système

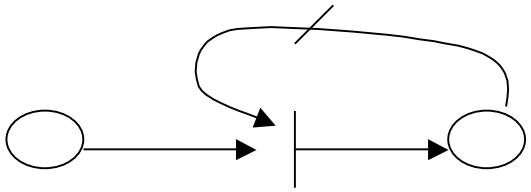
A2. Donner le marquage initial associé au réseau sachant qu'initialement le système d'attente est vide. Tracer les graphes de marquages conséquents de ce réseau (on constatera que le système ne comporte que 4 états).

-B-

On désire modéliser explicitement la discipline "premier arrivé, premier servi" (FIFO). Dans ce but, on va remplacer la place P_2 du réseau de Petri précédent par trois nouvelles places modélisant les positions dans le système d'attente et ne pouvant contenir qu'un jeton. On indique que le sous-réseau associé à deux places consécutives d'une file FIFO est le suivant :



Représente un arc inhibiteur.



Donner le nouveau réseau de Petri associé au problème. La place p_3 du réseau initial est-elle encore utile ? Pourquoi ? Donner le marquage initial du nouveau réseau dans les mêmes conditions qu'en A

Un atelier comporte deux machines identiques, utilisées en permanence dès lors qu'elles sont en état de fonctionner.

Lorsqu'une panne survient, on appelle, à son siège, un réparateur qui vient effectuer la réparation après un certain délai. Un seul réparateur est disponible.

Lorsqu'une réparation est terminée, la machine réparée est aussitôt remise en fonctionnement, et le réparateur rentre à son siège (en un temps négligeable), même si une autre machine est en attente de réparation.

On désire représenter le fonctionnement à l'aide d'un réseau de Petri comportant quatre places ; P_1, P_2, P_3 et P_4.

La marque P_1 représentera le nombre de machines en fonctionnement.

P_2 sera marquée lorsque le réparateur est à son siège : la marque de P_3 sera le nombre de machines en panne et en attente de réparation ; P_4 sera le marque lorsqu'une réparation en en cours.

Trois transitions t_1, t_2 et t_3 seront franchies respectivement, à l'occurrence d'une panne, au début d'une réparation (coïncidant avec l'arrivée du réparateur), à la fin d'une réparation.

1) Tracer le réseau de Petri associé, indiquer son marquage initial sachant qu'initialement les deux machines fonctionnent et que le réparateur est présent au siège.

Tracer le graphe des marquages conséquents du marquage initial.

Le réseau est-t-il vivant ? borné ? pour ce marquage initial.

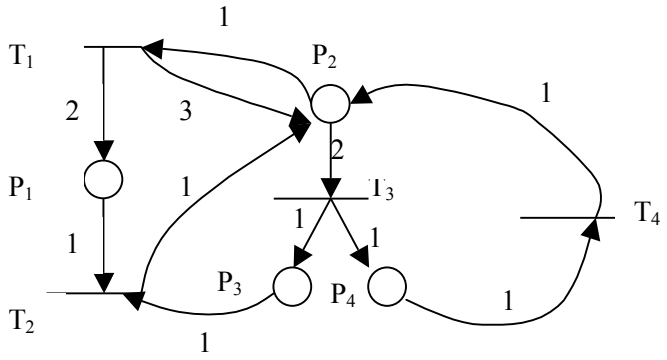
2) En remarquant que le nombre de machines est constant et égale à 2, associer un invariant de marquage du type $m(P_i) + m(P_j) + m(P_k) = 2$ en précisant la valeur de i, j, et k.

Faire de même en remarquant que le nombre de réparateurs est constant et égal à 1.

Donner la matrice C d'incidence du réseau.

Associer un P-semi flot à chacun des invariants de marquage ci-dessus.

On considère le réseau généralisé de la figure suivante :



1) Donner les matrices C^{+}, C^{-} et C de ce réseau.

2) Résoudre le système $f^T . C=0$ en nombres entiers positifs ou nuls.
En déduire un ensemble générateur minimal, de support minimal, des p-semi flots.
Que conclure sur la marque des places ?

3) Résoudre le système $C.s = 0$
En déduire un ensemble générateur minimal de support minimal des t-semi flots.
Que conclure sur la vivacité des transitions ?

4) On considère le marquage initial :

$$M_0 = \begin{pmatrix} 0 \\ 2 \\ 0 \\ 0 \end{pmatrix}$$

a) Calculer le ou les invariants $f^T.M = f^T.M_0$
b) Conclure sur les propriétés de borne et de vivacité du réseau pour M_0

5) On considère le marquage initial :

$$M'_0 = \begin{pmatrix} 0 \\ 3 \\ 0 \\ 0 \end{pmatrix}$$

a) Calculer le ou les invariants $f^T.M = f^T.M_0$
b) Construire l'arborescence de Karp et Miller ;
c) Déduire de l'arborescence le graphe de couverture associé (on identifie dans l'arborescence les sommets associées à des marquages identiques)
d) Vérifier l'invariant de marquage et conclure sur les propriétés de borne pour M'_0.

6) Montrer à l'aide des résultats précédents que les 2 énoncés suivants sont faux :
a) le fait qu'une place ne soit pas couverte par un p semi-flot serait une condition suffisante pour qu'elle ne soit pas bornée pour tout marquage initial
b) Le fait qu'une transition soit couverte par un t semi flot serait une condition suffisante pour qu'elle soit vivante.

MODELISATION D'UN SYSTEME D'ATTENTE EN PRODUCTION

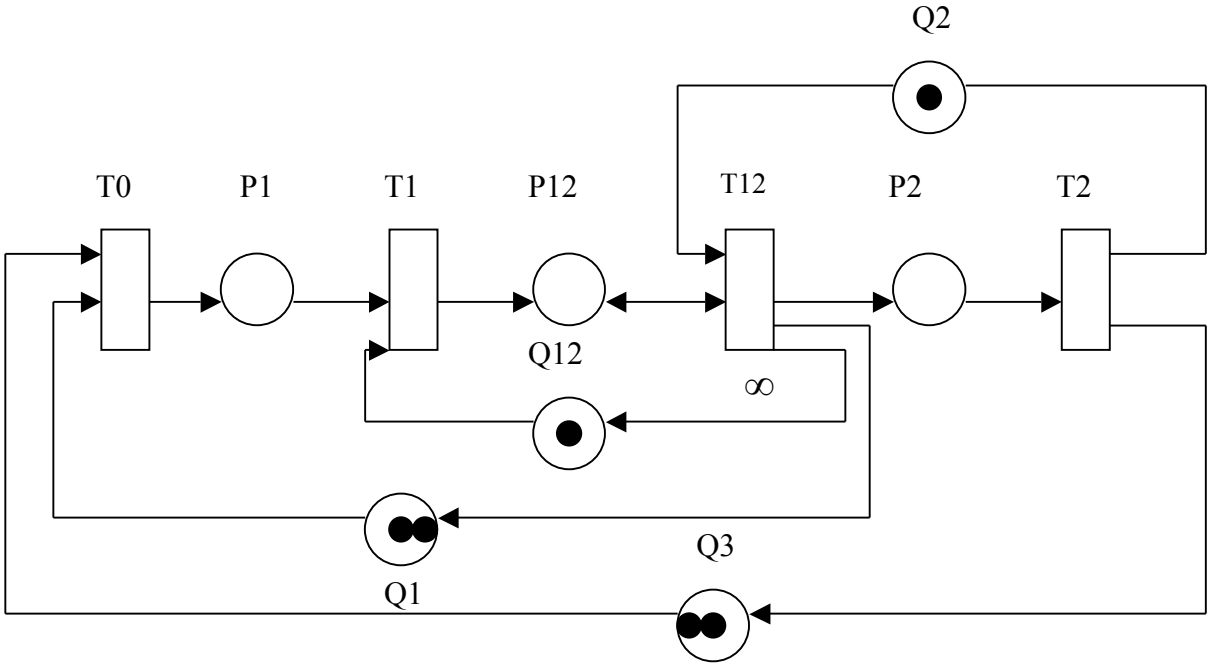
Nous allons étudier le fonctionnement dynamique (en univers aléatoire) d'un îlot flexible d'un système de production complexe.

Un nombre très important de pièces se déplaçant à l'aide de convoyeurs, passent en permanence devant plusieurs unités de fabrication(ou îlots) flexibles. Leur déplacement est contrôlé par un ordinateur superviseur. L'ordre de passage devant ces différents îlots est quelconque, mais il est nécessaire que chaque pièce passe devant les îlots prévus par sa gamme de fabrication.

Nous nous intéressons à l'un de ces îlots (voir le schéma suivant). A l'aide d'un détecteur, l'ordinateur supervisant le processus vérifie qu'il n'y a pas plus de deux pièces à la fois dans cet îlot. Si deux pièces y sont déjà présentes, toute pièce destinée à l'îlot est automatiquement véhiculée vers un autre îlot. Le flux des arrivées de pièce à l'entrée de l'îlot suit une loi de Poisson de taux λ.

L'îlot étudié est composé de deux machines en série (M1 puis M2) sur lesquelles les durées aléatoires de fabrication suivent des lois exponentielles de taux respectifs $\mu 1$ pour la machine M1 et $\mu 2$ pour la machine M2. Chaque machine contient au maximum 1 pièce ; il y a une place d'attente devant M1 mais aucune devant M2.

On a modélisé le fonctionnement de l'îlot par le réseau de Petri avec le marquage initial M0 ci-dessous :



T0 est la transition franchie lorsqu'une pièce entre dans l'îlot ;

T1 est la transition franchie lorsqu'une pièce termine sa fabrication sur M1 ;

T12 est la transition franchie lorsque le blocage de M1 cesse ;

T2 est la transition franchie lorsqu'une pièce termine sa fabrication sur M2.

A chacune de ces transitions est associé un taux ; pour T0, T1 et T2, ce taux est celui de la loi exponentielle associée (respectivement : , ì1 et ì2) ; pour T12, ce taux est infini ;: si les pré-conditions associées à T12 sont satisfaites, alors T12 est franchie instantanément.

Dans tout marquage M :

- M(P1) désigne le nombre de pièces sur M1 et sa file d'attente (ici réduite à une seule phrase) en cours ou en attente de fabrication (mais pas bloquées sur M1) ; si M(P1) = 0, aucune pièce n'est sur la machine M1 ; si M(P1) = 1, une pièce est en fabrication sur M1 ; Si M(P1) = 2, une pièce est en fabrication et l'autre en attente, sur M1.
- M(P12) = 1 ssi M1 est bloquée par une pièce qui y a terminé sa fabrication et attend que M2 se libère.

- M(P2) désigne le nombre de pièces sur M2.

- La place Q1, avec M0(Q1) = 2, assure qu'au plus 2 pièce sont sur M1 et sa place d'attente (la capacité de M1 égale 2).
- La place Q12, avec M0(Q12) = 1, assure qu'au plus 1 pièce est bloquée sur M1.
- La place Q2, avec M0(Q2) = 1, assure qu'au plus 1 pièce est bloquée sur M2 (la capacité de M2 égale 1).
- La place Q3, avec M0(Q3) = 2, assure qu'au plus deux pièces sont présentes à la fois dans l'îlot (ce que réalise le superviseur).

2) Donner la matrice d'incidence places-transitions du réseau, en respectant l'ordre suivant ;

Pour les places : P1, P12, P2, Q1, Q2, Q3, Q12

Pour les transitions : T0, T1, T12, T2.

Déterminer un générateur des P-semi-flots et interpréter concrètement , en termes d'invariant, chaque P-semi-flot fondamental. On posera ft = [a, b, c, d, e, f, g] et l'on exprimera a, b et c en fonction de d, e, f et g pour déterminer les P-semi-flots fondamentaux.

2)a) Avec le marquage initial indiqué sur le réseau de Petri ci-dessus :
M0(P1) = M0 (P12) = M0(P2) = 0 ; M0(Q2) = M0(Q12) = 1 ; M0(Q1) = M0(Q3) = 2.

Tracer le graphe des marquages accessibles depuis M0, comme suit : on figurera dans ce graphe un seul sommet pour un marquage instantané, et le marquage successeur de celui-ci.
Ainsi pour le marquage M tel que :

M(P1) = 0, M(P12) = 1, M(P2) = 0 ; M(Q1) = 1, M(Q12) = 0; M(Q2) = 1, M0(Q3) = 1,
noté en abrégé : P12 + Q1 + Q2 + Q3, la transition T12 est franchie instantanément et conduit au marquage :
M(P1) = 0 , M(P12) = 0, M(P2) = 1 ; M(Q1) = 2, M(Q12) = 1; M(Q2) = 0, M(Q3) = 1,
noté : P2 + 2Q1 – Q12 +Q3.

Ces deux marquages seront donc représentés par le même sommet.
Valuer les arcs associés aux transitions T0, T1 et T2 par le taux de ces transitions.
Que constatez-vous pour ce graphe ?
Commentez alors l'intérêt de la modélisation par réseau de Petri.
2)b) Le réseau est-il vivant ? Borné ?

3) Si l'on avait 1 place d'attente devant M1 et 2 places d'attente devant M2 (comme au A-3) pour une capacité totale de 5 pièces pour l'îlot, quelle(s) modification(s) devrait-on introduire au réseau de Petri ?
Il n'est pas demandé de retracer le graphe des marquages accessibles.
Réinterpréter alors chaque P-semi-flot fondamental dans ce nouveau contexte.