- corrigé -

NOM PRENOM GROUPE

Graphes – 3MIC

Lire l'ensemble du sujet avant de commencer — Barème pouvant être modifié

Durée : 1h30 — Vendredi 17 mai 2019

EXERCICE 1. GRAPHE BIPARTI - PARCOURS — 7 POINTS

Un graphe non orienté est biparti si on peut séparer ses sommets en deux sous-ensembles S_1 et S_2 qui vérifient les propriétés suivantes :

- Le graphe est connexe
- Il n'y a aucune arête entre deux sommets de S_1
- Il n'y a aucune arête entre deux sommets de S_2

Toutes les arêtes du graphe relient donc un sommet de S1 à un sommet de S_2.

Question 1. Proposez une adaptation d'un **algorithme de parcours en largeur** permettant de déterminer si un graphe non orienté est biparti ou non. Quelle est sa complexité ? **(2 points)**

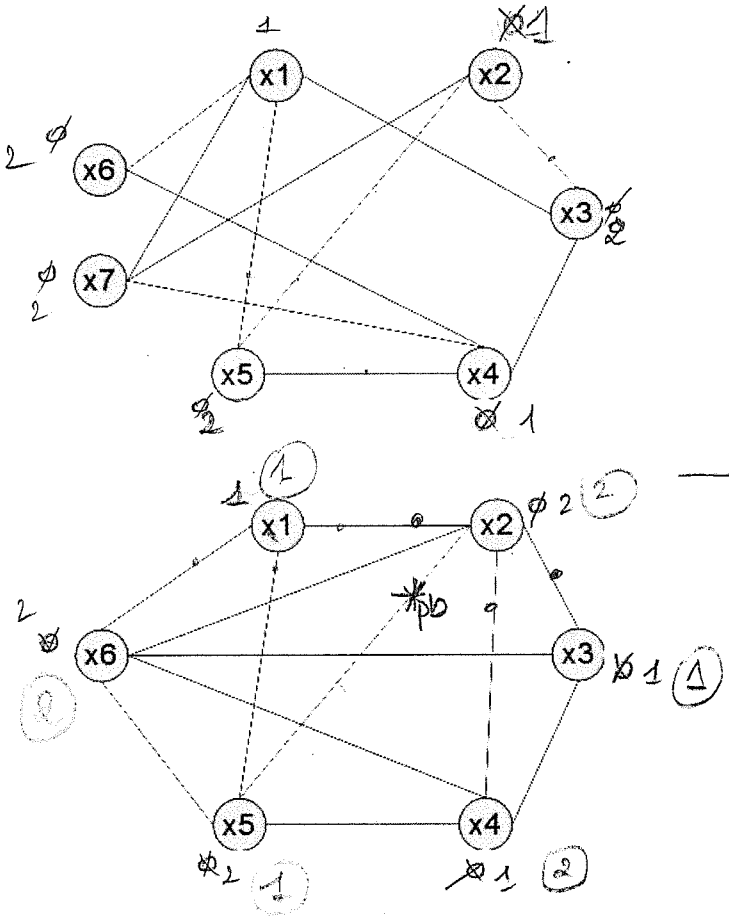
Algorithme de parcours en largeur à partir d'un sommet initial x_0 et utilisant une file F.

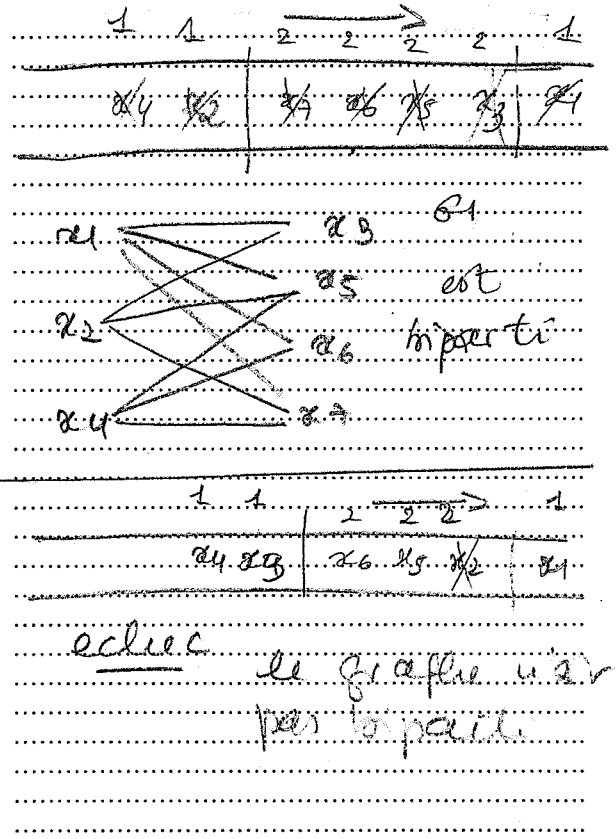
```
ETAT(x0) ← gris;
CRÉER_FILE(F); AJOUTER_FILE(F,x0);

while not EST_VIDE_FILE(F) do
      x ← RETIRER_FILE(F);  ETAT(x) ← noir;
      for tout y ∈ voisin(x) do
            if ETAT(y) == blanc then
                  AJOUTER_FILE(F,y)); ETAT(y) ← gris;
            endif
      endfor
endwhile
```

Définir tableau : biparti[x] ∀ x sommet du graphe = (0, 1, 2)
bip ← true; biparti[x] ← 0; biparti[x0] ← 1;
while non est vide File(F) do
    x ← retirer File(F); Etat(x) ← noir
    for tout y ∈ voisin(x) do
        if biparti(y) = biparti(x) then bipar ← faux; exit;
        if biparti(y) = 0 then biparti(y) ← opp(biparti(x))
        if etat(y) == blanc then
            ajouter File(F,y), Etat(y) ← gris
        end if
    end for
end while
if bip then -- le graphe est biparti
    -- les sommets noir ont un n° (1 ou 2)
else : echec : le graphe n'est pas biparti
end if —> même complexité que Parcours

Question 2. Appliquez l'algorithme proposé pour déterminer si les deux graphes ci-dessous sont bipartis.
(2 points)





Question 3. Votre algorithme permet-il de détecter des graphes non connexes ? **(1 points)**

– Oui - Si tous les sommets ne sont pas visités, et qu'aura de valeurs 0 dans bipartie (x)

Question 4. Serait-il possible d'adapter un algorithme de parcours en profondeur ? Quelles seraient les similitudes / différences ? **(2 points)**

- A chaque extrémité des arêtes il faut vérifier s'il a déjà une valeur bipartie ou pas

Différence ⇒ parcourir tte les arêtes dans parcours en profondeur

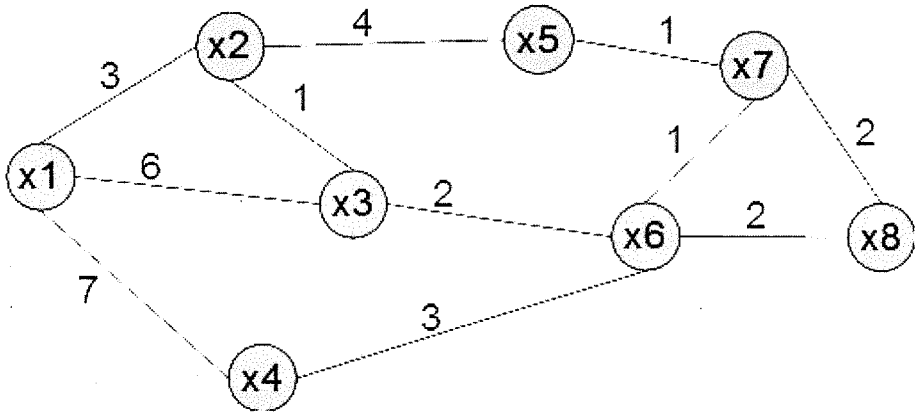
⇒ pas besoin d'un sommet de départ (le 1er non Blanc)

⇒ adaptation plus complexe et nécessaire probablement à des pertes en largeur.

EXERCICE 2. PLUS COURT CHEMIN — 7 POINTS

On souhaite appliquer l'algorithme de Dijkstra pour déterminer le plus court chemin entre une origine et une destination de manière bidirectionnelle : une partie des itérations s'effectue à partir de l'origine (sens forward) et une partie des itérations s'effectue à partir de la destination (sans backward).

On va appliquer cet algorithme bidirectionnel sur l'exemple ci-dessous pour calculer le plus court chemin de x1 vers x8.



Pour cela, on va créer 2 ensembles de labels : des labels Forward pour le parcours depuis l'origine et des labels Backward pour le parcours depuis la destination.
A chaque itération, l'algorithme commence par sélectionner le sommet de cout minimal. Si ce sommet est dans l'ensemble Forward (respectivement Backward), l'algorithme le marque et l'étend en mettant à jour éventuellement d'autres labels Forward (respectivement Backward). Les labels dans le sens inverse ne sont pas modifiés.

Le tableau ci-après donne les premières étapes de l'algorithme :

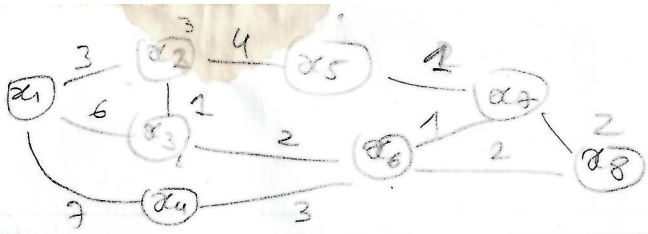
- Initialisation : les coûts des labels Forward et Backward sont initialisés (0 pour le point de départ, c'est à dire x1 pour les labels Forward et x8 pour les labels Backward.
- Etape 1 : le sommet de plus petit coût est sélectionné (en cas d'égalité on choisit les labels de sens Forward). Ici le label du sommet x1 Forward est sélectionné et son coût permet d'actualiser ses voisins. Les labels Backard sont juste recopiés
- Etape 2 : le sommet de plus petit coût est sélectionné. Il s'agit du sommet x8 des labels Backward, son coût permet d'actualiser ses voisins. Les labels Forward sont juste recopiés.
- Etape 3 : le sommet de plus petit coût est sélectionné. Il s'agit du sommet x7 des labels Backward, son coût permet d'actualiser ses voisins. Les labels Forward sont juste recopiés.

Pour améliorer la lisibilité, le sens de parcours et noté en grisé en chaque itération ainsi que le label marqué. Ce label marqué est également noté entre parenthèse et recopié lors des itérations suivantes.

Question 1. Poursuivre le déroulement de l'algorithme. On supposera que la condition d'arrêt est qu'un même sommet soit marqué à la fois dans les labels Forward et dans les labels Backward (il peut y avoir plus de lignes que nécessaire dans le tableau). **(3 points)**

Donner la valeur du plus court chemin obtenu : 8

Donner la liste des sommets de ce plus court chemin : $x_1 - x_2 - x_3 - x_6 - x_8$



→ 8

→ x1 - x2 → x3 → x6 → x8

		X1	X2	X3	X4	X5	X6	X7	X8
Init	Fwd	0	∞	∞	∞	∞	∞	∞	∞
	Bckwd	∞	∞	∞	∞	∞	∞	∞	0
1	Fwd	(0)	3, x1	6, x1	7, x1	∞	∞	∞	∞
	Bckwd	∞	∞	∞	∞	∞	∞	∞	0
2	Fwd	(0)	3, x1	6, x1	7, x1	∞	∞	∞	∞
	Bckwd	∞	∞	∞	∞	∞	2, x8	2, x8	(0)
3	Fwd	(0)	3, x1	6, x1	7, x1	∞	∞	∞	∞
	Bckwd	∞	∞	∞	∞	3, x7	2, x8	(2, x8)	(0)
4	Fwd	(0)	3, x1	6, x1	7, x1	∞	∞	∞	∞
→	Bckwd	∞	∞	4 x6	5 x6	3, x7	(2, x8)	(2, x8)	(0)
5 →	Fwd	(0)	(3, x1)	4 x2	9 x1	7 x2	∞	∞	∞
	Bckwd	∞	∞	4 x6	5 x6	3 x7	(2, x8)	(2, x8)	(0)
6	Fwd	(0)	(3, x1)	4 x2	7 x1	7 x2	∞	∞	∞
→	Bckwd	∞	7 x2	4 x6	5 x6	(3, x7)	(2, x8)	(2, x8)	(0)
7 →	Fwd	(0)	(3, x1)	(4, x2)	7 x1	7 x2	6, x3	∞	∞
	Bckwd	∞	7 x2	4 x6	5 x6	(3, x7)	(2, x8)	(2, x8)	(0)
8	Fwd	(0)	(3, x1)	(4, x2)	7 x1	7 x2	6 x3	∞	∞
(Stop) →	Bckwd			(4, x6)		(3, x7)	(2, x8)	(2, x8)	(0)
9	Fwd								
	Bckwd								
10	Fwd								
	Bckwd								

Question 2. Lors des itérations, un même sommet peut être atteint par les deux parcours (label non infini dans le sens Forward et le sens Backward). **(2 point)**

A quelle étape cela se produit-il pour la première fois ? Etape 4

Quel(s) est (sont) les chemin(s) correspondant ? Donnez également le(s) coû(s) associé(s) :

x1 - x2 - x6 - x8 - coût 10 (6+4)

x1 - x4 - x6 - x8 - coût 12 (7+5)

Question 3. Si la condition d'arrêt de l'algorithme bidirectionnel était : arrêt lors un même sommet est atteint par les deux sens de parcours, est-ce que le résultat obtenu serait correct ? Justifiez la réponse.

(2 points)

Non, on le voit à la question précédente (contre-exple)

Pourquoi : la 5ème étape trouve un meilleur coût des sommets mais au FW et BW est 4 au coût des chemins trouvés avec les sommets atteint dans un 2 sens

1 étape (4) Σ = 6

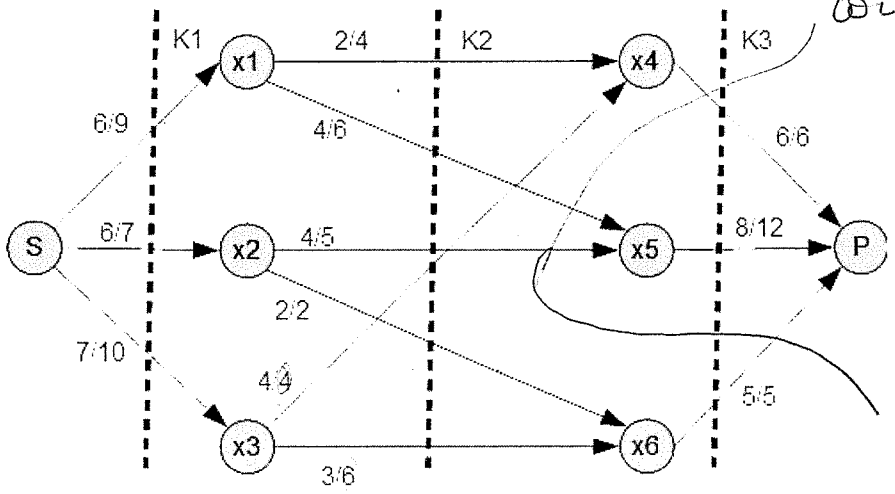
Meilleur chemin connu. 10)

(Il peut exister un chemin de + petit coût passant par les sommets)

EXERCICE 3. FLOT MAXIMUM — 7 POINTS

On considère le problème de flot maximum représenté par le graphe ci-dessous.

coupe dernière itération



Question 1. **(2 points)**

Les valeurs des flots sont-elles admissibles ?

oui (flot ≤ capacité ; Σ flots entrants = Σ flots sortant sauf source et puits ; Σ sorties source = Σ entrantes puits)

Quel est le débit du flot ? 19

Quelles sont les capacités des 3 coupes (K1, K2 et K3) ?

K1 de capacité 26 (= 9+7+10)

K2 de capacité 27 (= 4+6+5+2+4+6)

K3 de capacité 23 (= 6+12+5)

Question 2. Déterminez le flot maximum à l'aide des schémas ci-après (utilisez un parcours en largeur et ordre alphabétique) et donnez les résultats demandés. **(3 points)**

Valeur du flot maximum : 22

Coupe établie lors de la dernière itération : {(S), (x1), (x2), (x3), (x4), (x5)} | {(x5), (P)}

Capacité de cette coupe : 6 + 6 + 5 + 5 = 22

Question 3. Vous avez la possibilité d'augmenter la capacité **d'un seul arc** de la valeur que vous souhaitez afin d'augmenter le flot circulant actuellement dans le graphe. **(2 points)**

Quel est l'arc choisi ? Quel est sa nouvelle capacité ? Justifiez.

choisir un arc sortant de la coupe et flot entrant en x4

x4 → P : on ne peut augmenter que de 1 (d'après le graphe d'écart)

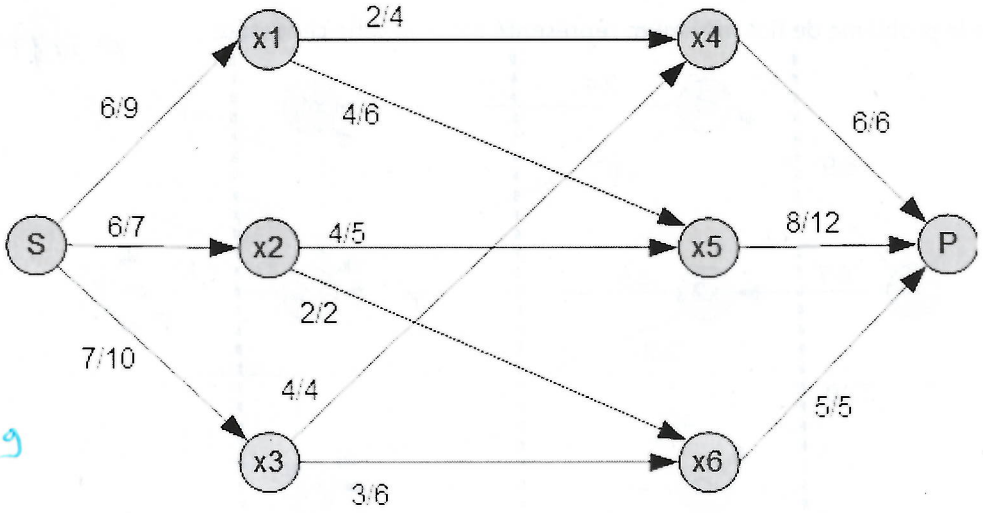
x1 → x5 :

x2 → x5 : on ne peut pas augmenter

Quel est le flot maximum pouvant alors circuler sur le graphe ? 25

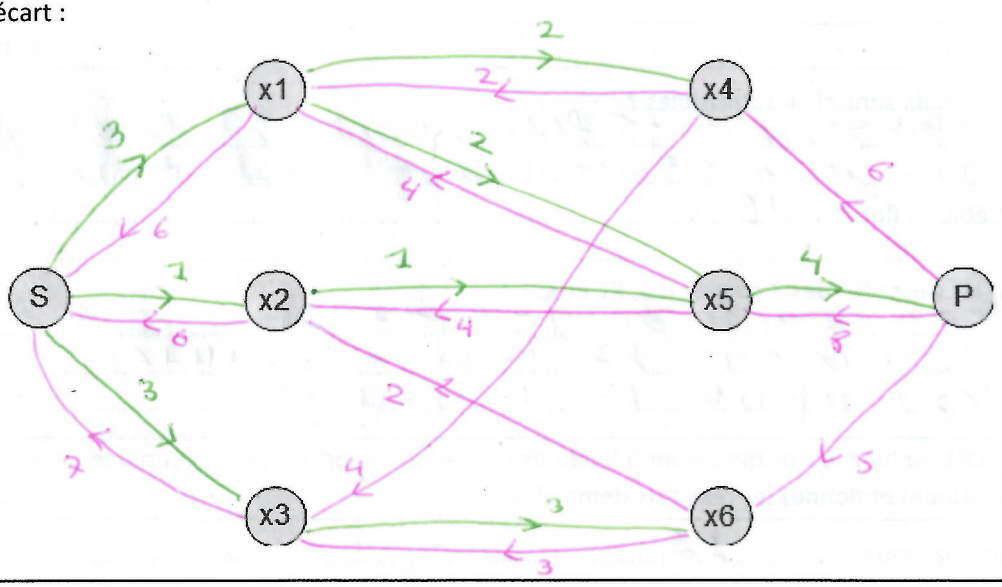
x6 → P : on peut ↑ le flot entrant en x6 de 3 ; il suffit donc d'augmenter de 3 la capacité de x6 → P

Graphe de flot :

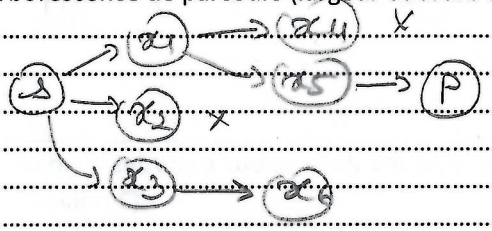


Flot= 19

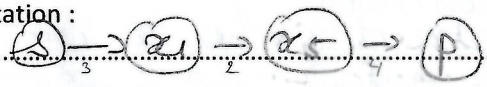
Graphe d'écart :



Arborescence de parcours (largeur et ordre alphabétique) :

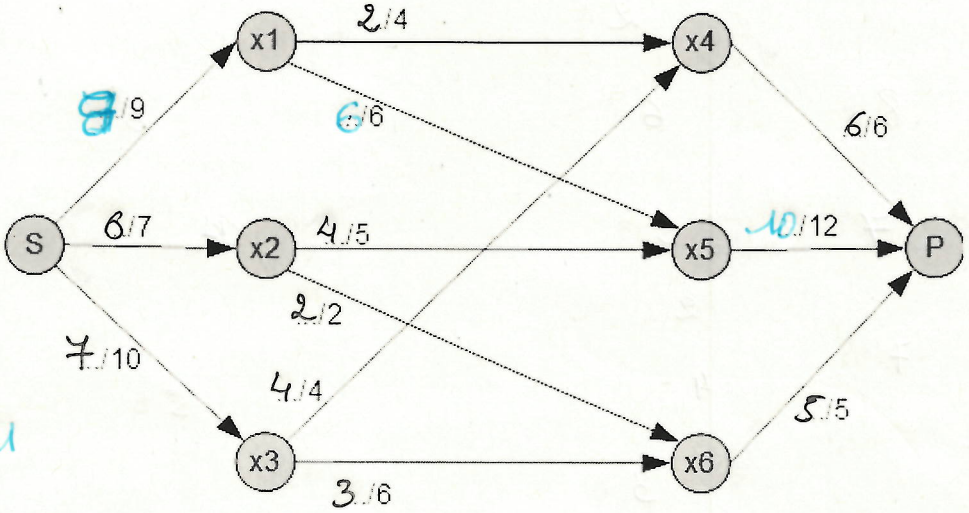


Chemin d'incrémentation :



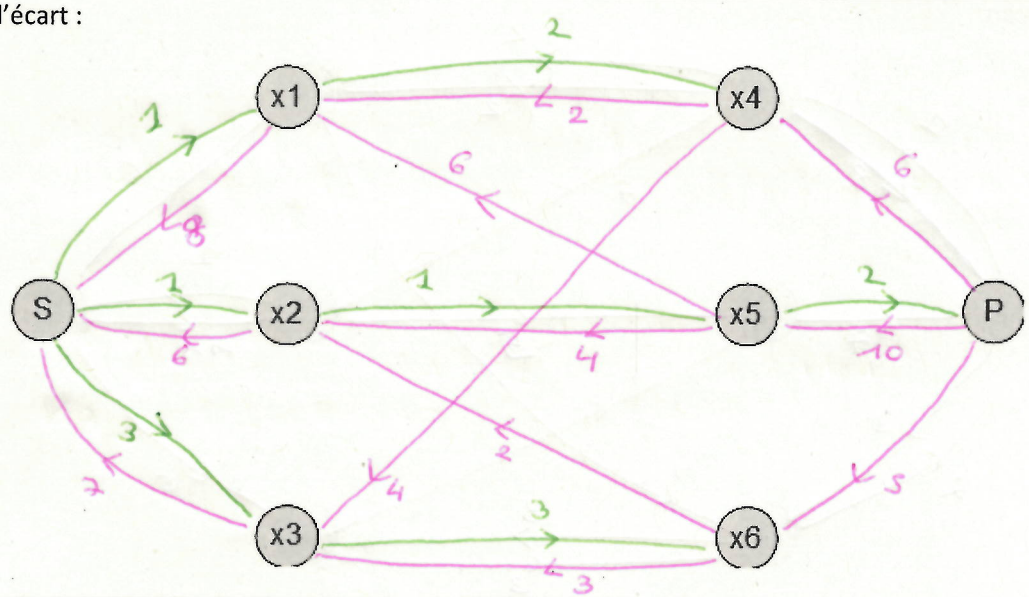
Valeur d'incrémentation : 2

Graphe de flot :

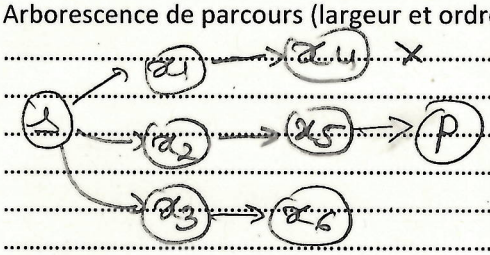


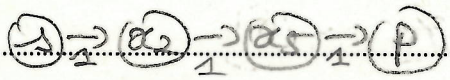
Flot = 21

Graphe d'écart :



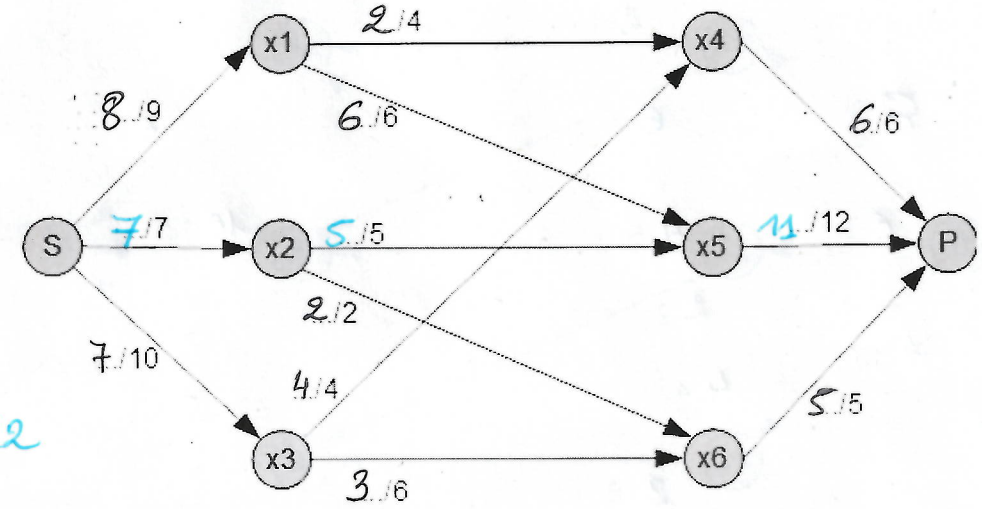
Arborescence de parcours (largeur et ordre alphabétique) :



Chemin d'incrémentation : 

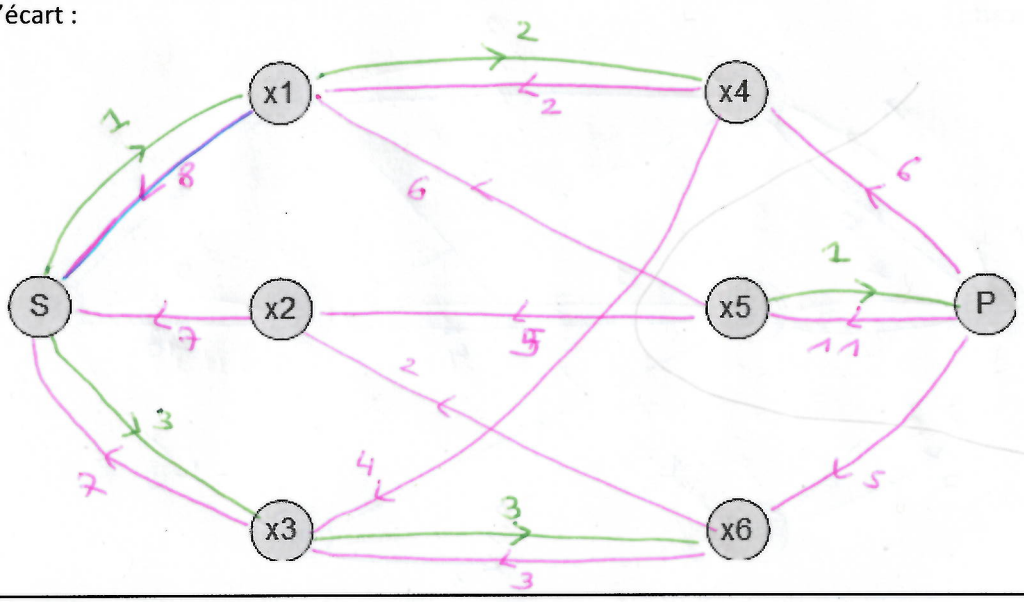
Valeur d'incrémentation : 1

Graphe de flot :

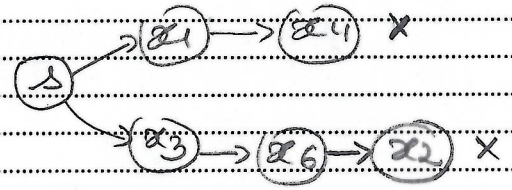


Flot = 22

Graphe d'écart :



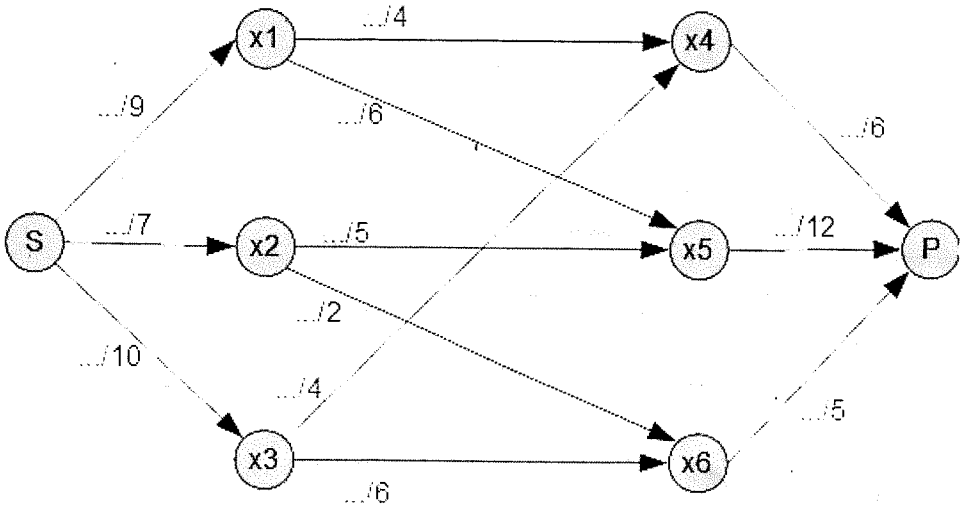
Arborescence de parcours (largeur et ordre alphabétique) :



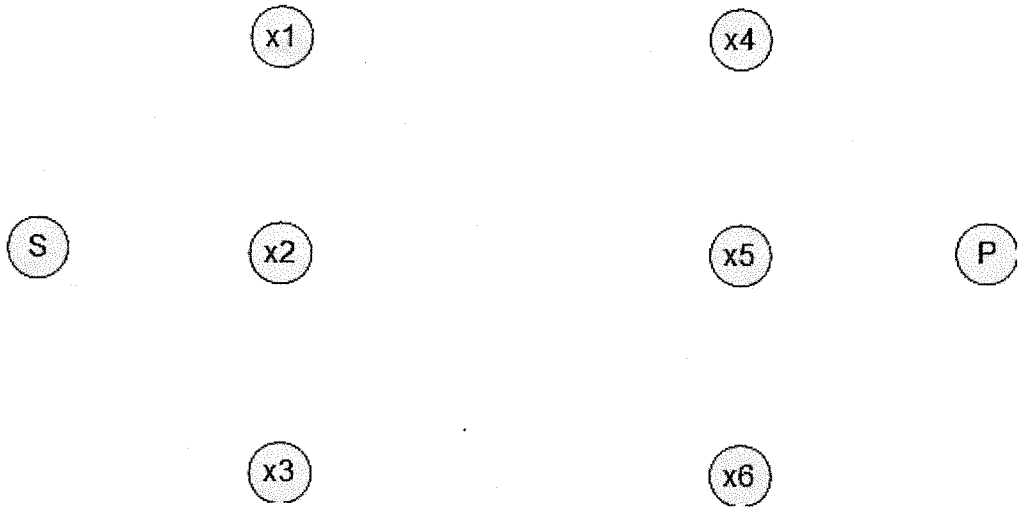
Chemin d'incrémentation : aucun

Valeur d'incrémentation : .....

Graphe de flot :



Graphe d'écart :



Arborescence de parcours (largeur et ordre alphabétique) :

.....

.....

.....

.....

.....

.....

Chemin d'incrémentation :

.....

.....

Valeur d'incrémentation : .....

.....

.....