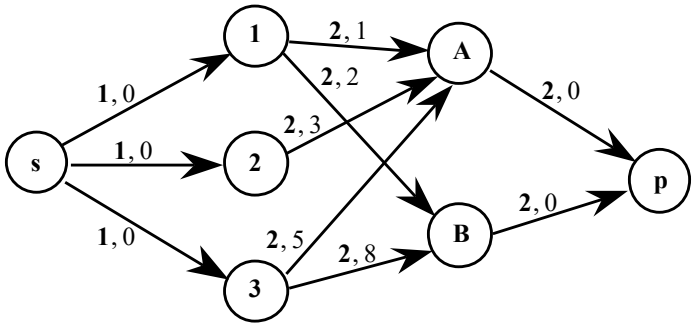
Exercice 1 : (3 points)

Faire tourner l'algorithme de ROY sur le graphe ci-dessous afin de déterminer un flot maximal à coût minimal. On rapportera les graphes d'écart et les chemins améliorants successifs, ainsi que le flot final et son coût.



notation (**capacité**, coût)

Premier flot : s-1-A-p (flot=1 et cout=1)

2eme flot : s-2-A-p (flot=1, cout=3)

3"m" flot : s-3-a-1-B-p (flot=1, cout=6)

Flot maximal = 3 et cout global = 10.

Exercice 2 (7 points)

Soit G=(X,U,c) un graphe dont les arcs sont valués par une valuation strictement positive représentant les capacités des arcs. On rappelle que la capacité d'un chemin est la plus petite valuation des arcs de ce chemin.

Question 1. Modifier l'algorithme de Dijkstra (rappelé ci-dessous) pour calculer les chemins de capacité maximale issus du sommet x0.

Rappel : Algorithme de Dijkstra modfifié (CAPACITE MAXIMALE)

Début
Poser S=(x0), λ_0=infini; λ_i=c(x_0, x_i) si (x_0, x_i) est un arc du graphe et λ_i=0, sinon.
Tant que S différent de X faire :
début
choisir x_i dans X-S de λ_i maximal;
poser S=S U (x_i);
pour x_j dans X-S et successeur de x_i faire : λ_j:=Max(λ_j, Min(λ_i, v_{ij})).
fin
Fin

Question 2. Appliquer l'algorithme sur le graph G et calculer les chemins de capacités maximales issus de x0. On rapportera le tableau analogue à celui de DIJKSTRA. Déterminer une arborescence de chemins de capacités maximales. Y-a-t-il unicité des chemins de capacités maximales? Justifier.

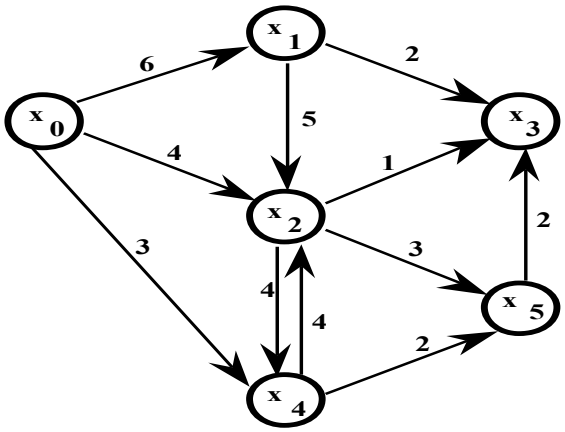
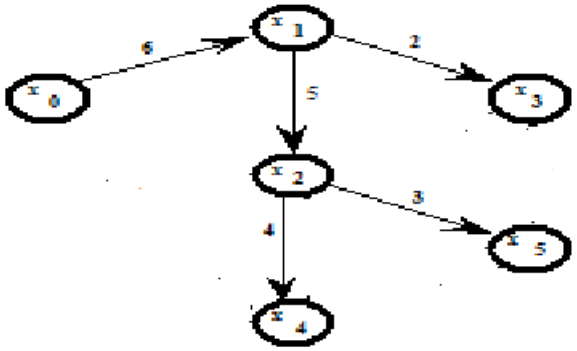


Tableau :

S\lambda	λ0	λ1	λ2	λ3	λ4	λ5
{0}	infini	6	4	0	3	0
+{1}	infini	6	5	2	3	0
+{2}	infini	6	5	2	4	3
+{4]	infini	6	5	2	4	3
+{3}	infini	6	5	2	4	3



Il n'y a pas d'unicité car la valeur de λ3 on peut l'obtenir aussi en venant du nœud x5 (et d'autres cas comme λ5).

Question 3. (Question indépendante de l'application) Proposer une méthode générale qui calcule le chemin le plus court parmi les chemins de capacités maximales.

Il suffit de garder les arcs avec des capacités supérieur ou égal à cette valeur de capacité max et de calculer dans le sous-graphe les chemins longueur min.

Rappel : Algorithme de Dijkstra

(i) Poser S ={x_0}, $\lambda_0 = 0$, $\lambda_i = v_{0i}$, si $(x_0\ x_i) \in U$, $\lambda_i = \infty$,sinon.
(ii) Tant que S ≠ X faire

(a) Choisir $x_i \in X - S$ de λ_i minimum.
(b) Poser $S = S + \{ x_i \}$.
(c) Pour tout $x_j \in (X - S)$, successeur de x_i , poser: $\lambda_j = \min(\lambda_i + v_{ij}, \lambda_j)$.

Exercice 3. Le problème du Bin-Packing (10 points)

Dans ce problème, on a n articles u1, u, …, u de taille s(u1), s(u2), …, s(un) comprise entre 0 et 1. Il s'agit de mettre ces articles dans un nombre minimum de boites de capacité 1. Mathématiquement, il faut déterminer une partition U1, U2, Uk de l'ensemble des articles telle que :

- Les contraintes de capacités soient satisfaites
  $\sum_{u \in Uj} s(u) \leq 1$ (j=1, .., k)
- Le nombre k de boites soit minimum.

Question 1. Etude de l'Algorithme FIRST FIT

{on suppose qu'on dispose d'un nombre infini de boite de capacité 1 ; ces boites sont numérotées par les entiers naturels : on met l'article i dans la boite admissible de plus petite indice.}

FIRST FIT

Début

{Les articles sont indicés selon une liste quelconque}

Pour i=1 à n faire

Début

Déterminer le plus petit numéro de boite j pour lequel la place restant disponible est supérieur ou égale à s(ui) ;

Affecter l'article ui à la boite j.

Fin.

FIN.

A. (4 points)
   1. Appliquer l'algorithme FIRST FIT à la donnée suivante : on a 10 articles de tailles : s(u1)=0,5 ; s(u2)=0,1 ; et ainsi de suite 0,3 ; 0,4 ; 0,7 ; 0,2 ; 0,6 ; 0,6 ; 0,1 et le dernier s(u10)=0,5.
      Ca donnera : u1-> 1 ; u2-> 1 ; u3-> 1 ; u4-> 2 ; u5-> 3 ; u6 -> 2 ; u7-> 4, u8-> 5, u9-> 1 ; et u10-> 6.
   2. Evaluer la complexité de l'algorithme en fonction de n. Toute réponse non justifiée sera considérée comme fausse.
      Complexité $O(n^2)$…
   3. Que proposez-vous pour améliorer le résultat de l'algorithme FIRST FIT ? Appliquer l'amélioration que vous proposez sur l'exemple ci-dessus.

On pourra examiner les articles dans l'algorithme FIRST FIT dans l'ordre décroissant de leurs valeurs. Cela donnerait alors seulement 4 boites.

B. (4 points) On se place dans le cas général. D dénote une donnée quelconque de FIRST FIT. On veut démontrer que FF(D) < 2 F*(D), où FF(D) est le nombre de boites non vides utilisés par FIRST FIT, et F*(D) le nombre optimal de boites.
   1. Démontrez qu'au cours de l'algorithme le nombre de boites non vides, dont le contenu est inférieur où égal a 0,5, vaut au plus 1.

La réponse est évidente car si on avait deux boites avec au plus une valeur de 0,5, tous les articles de la deuxième boite auraient du être placés dans la première car dans celle-ci il y avait toujours de la place disponible.

Dans la suite, on note a le contenu de la boite la moins chargée.

2. Cas a > 1/2
   Justifier FF(D)/2 < $\sum_{i=1}^{n} \mathrm{s(ui)}$ (i=1, …, n) et F*(D) >= $\sum_{i=1}^{n} \mathrm{s(ui)}$, et en déduire le résultat.
   Réponse : La première inégalité est vraie car la somme des contenus de toutes les boites donnerait $\sum_{i=1}^{n} \mathrm{s(ui)}$ qui est forcément > le nombre de boites * une borne inférieure des contenances de boites = FF(D) * 1/2 (car le contenu de chaque boite est > 1/2). La deuxième inégalité est immédiate. Des deux inégalités en déduit directement le résultat de 2.
3. Cas a <= 1/2
   Utiliser les résultats démontrés en 1) et 2) pour démontrer le résultat.
   Réponse : notons d'abord que la contenance de toutes les boites sauf la moins chargée est strictement supérieure à (1-a). On en déduit alors que la valeur moyenne de contenance de l'ensemble des boites est > 1/2 et donc 1/2 donne une borne inférieure de cette valeur moyenne. On applique alors le raisonnement suivi en 2., et on obtient le résultat désiré.

Question 2. Proposer un model PL pour ce problème. (2 points)

Variables

Yj est binaire = 1 si boite j est utilisée.

Xij est binaire = 1 si élément i est placé dans la boite j.

Min $\sum_j yj$

$\sum_i (xij * s(ui)) \leq yj$ pour tout j

$\sum_j xij = 1$ pour tout i

yi et xij binaire