Correction TD 11

LE PROBLEME DU SAC A DOS (d'après ROSEAUX)

Un randonneur, partant pour une longue excursion, détermine avec soin le contenu de son sac à dos. Compte tenu des équipements indispensables déjà chargés, le poids total de nourriture emportée ne devra pas excéder 16 kilogrammes. Il dispose, en quantités limitées, de trois types d'aliments, de valeurs nutritives variables, dont les poids unitaires sont différents. Les aliments sont conditionnés par unités non fractionnables.

Aliments	I	II	III
poids unitaires en kilo	7	5	2
quantités disponibles	4	3	4
valeurs nutritives	15	10	4

Le randonneur choisit la quantité de chaque aliment à emporter, de façon à maximiser la valeur nutritive totale, tout en tenant compte de la limite de 16 Kg qu'il s'est fixée.

Il va utiliser la "programmation dynamique" pour résoudre ce problème dit du "Knapsack" ou sac à dos. D'une manière générale, soit :

n, le nombre d'aliments différents;

p_i le poids unitaire de l'aliment i ($i \in [1,n]$);

q_i le nombre d'unités disponibles de l'aliment i ($i \in [1,n]$);

c_i la valeur nutritive de l'aliment i ($i \in [1,n]$);

x_i la quantité de l'aliment i emportée en nombre d'unités ($i \in [1,n]$) (ce sont les inconnues);

et P le poids maximal de nourriture qu'il peut emporter.

1) Donner une formalisation du problème en tant que programme linéaire en nombre entiers.

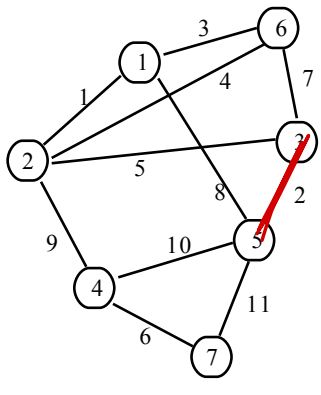
2) Tracer le graphe des décisions associé au problème, construit de façon analogue à l'exercice de TD précédent sur le problème de l'investissement.

3) Résoudre numériquement.

4) En notant $f_i(Q)$ la valeur nutritive maximale du sac à dos chargé optimalement avec uniquement les i premiers produits, et de poids total Q (Q inférieur ou égal à P). Donner les formules d'optimisation séquentielle, c'est-à-dire des formules reliant la fonction f_i à la fonction f_{i-1}.

5) Evaluer la complexité de la méthode en fonction de n et de P et expliquer son intérêt en la comparant à une méthode énumérative.

PROBLEME: ALGORITHME DE KRUSKAL:



L'algorithme de KRUSKAL permet de déterminer un arbre couvrant de coût minimal d'un graphe valué **connexe** G=(X,U, V) à N sommets et M arêtes. **Il consiste à partir d'un graphe réduit à ses sommets isolés, puis à ajouter pas à pas les arêtes en coûts croissants sous réserve que l'arête ajoutée ne crée pas de cycle.**

On supposera que les coûts sont tous distincts.

KRUSKAL

{par convention l'arête e_L s'écrira e_L =[I,J] avec I strictement plus petit que J}

```
Début
    Indicer les arêtes en ordre de coûts croissants: v(e1) ≤ v(e2) ≤ ... ≤v(eM)
    Pour H:=1..N faire  CONNEXE(H):=H;
    K:=1; {le graphe partiel est initialement vide}
    Pour L:=1..M faire
      Début
          soit eL=[I,J];
          Si  (CONNEXE(I) différent de CONNEXE(J)) alors
          Début
             AUXI:=CONNEXE(J); fK:=[I,J]; { fK  est mis dans le graphe partiel}; K:=K+1;
             Pour H:=1..N faire
             Si (CONNEXE(H)=AUXI) alors CONNEXE(H):=CONNEXE(I)
          Fin
      Fin
Fin
```

Question 1: Faire tourner l'algorithme sur le graphe ci-dessus: on dessinera les graphes partiels successifs et les onze valeurs successives du tableau CONNEXE obtenu avant chaque itération de la boucle externe.

Question 2: Evaluer en fonction de N et M la complexité de l'algorithme. On pourra distinguer deux types d'itérations dans la grande boucle.

Question 3 (cette question est indépendante de l'algorithme)

On rappelle que l'on est dans le cas où tous les coûts sont distincts.

On va démontrer le théorème suivant :

« L'arbre de cout minimal d'un graph connexe existe et il est formé des arêtes de cout minimal des cocycles de G. »

a) Soit A un arbre couvrant de coût minimal et ω(Y) un cocycle de G. Montrer par l'absurde que A rencontre ω(Y) en son arête de coût minimal. Un arbre de coût minimal existe puisque le graphe est connexe.

b) Réciproque : Soit A un arbre couvrant de coût minimal et [i,j] une arête de A, montrer que [i,j] est l'arête de coût minimal d'un cocycle de G que l'on identifiera.

c) En déduire une méthode pour construire un arbre de coût minimal.

Question 4: Montrer que l'algorithme détermine l'arbre couvrant de coût minimal. On montrera qu'à chaque itération l'algorithme choisit l'arête minimale d'un cocycle de G. Que ferait l'algorithme si le graphe n'était pas connexe ?

Q1 Formulation PLNE: notons xi la variable de décision qui exprime la quantité de l'aliment i placé dans le sac. La fonction objectif maximise la valeur nutritive du sac sous contraintes de capacité maximale du sac et de disponibilité des aliments.

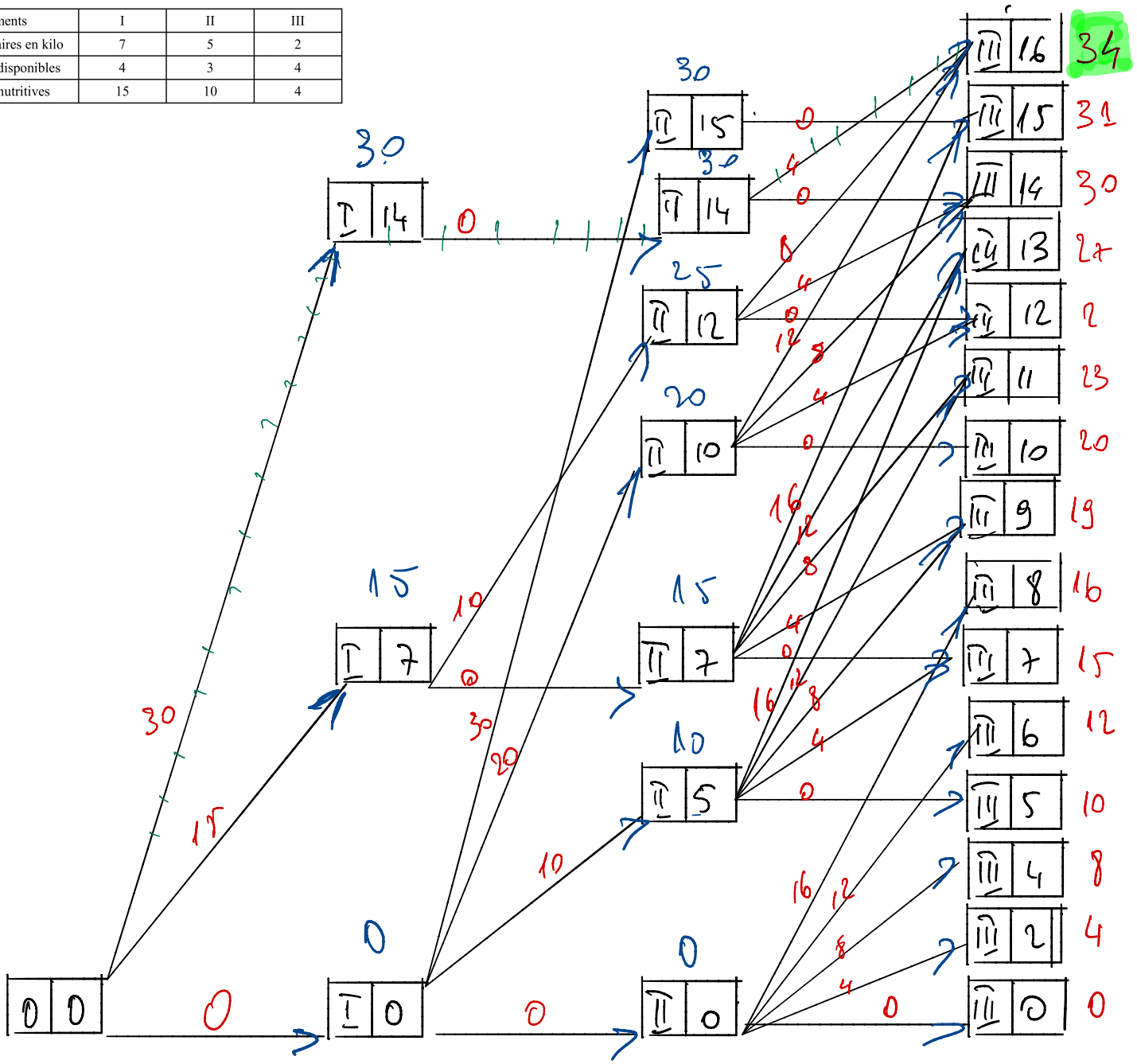
$$\text{Max} \sum_{i=1}^{n} c_i x_i$$

$$\text{S.C.: } \sum_{i=1}^{n} p_i x_i \leq P$$

$$0 \leq x_i \leq q_i, \quad x_i \in \mathbb{N} \quad \forall \text{ aliment } i.$$

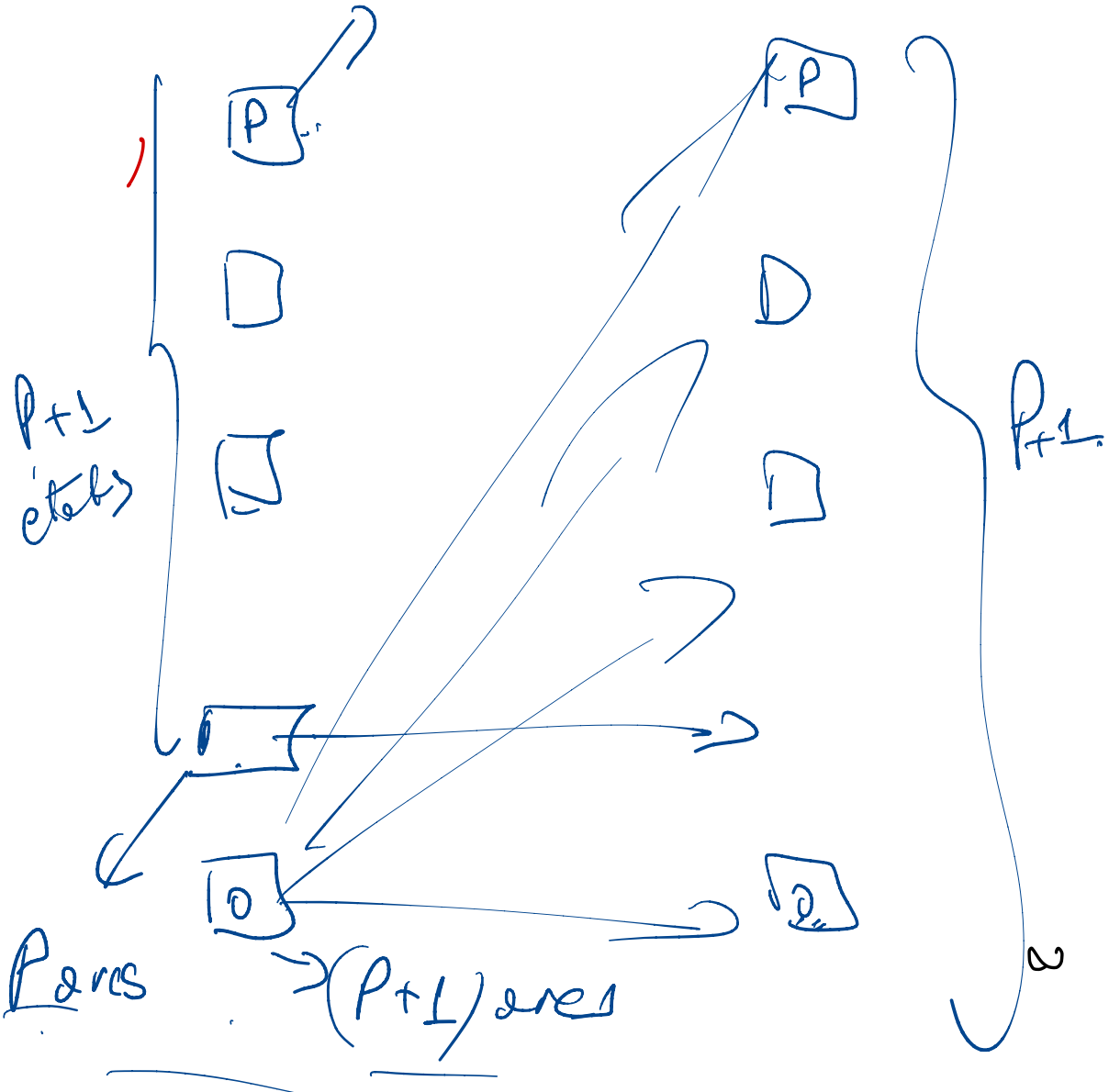
Q2

Aliments	I	II	III
poids unitaires en kilo	7	5	2
quantités disponibles	4	3	4
valeurs nutritives	15	10	4



Q3. Solution optimale : $x_1 = 2$, $x_2 = 0$, $x_3 = 1$, valeur nutritive 34.

Nombre d'arcs pour une étape



$$1+2+3+\dots+(P+1) = \frac{(P+1+1)}{2} * (P+2)$$

$$\Theta(P^2)$$

Nombre d'arcs pour une étape

$\Rightarrow$ Complexité globale $O(n P^2)$

Q4

avec $y_i = p_i x_i + y_{i-1}$ et $y_0 = 0$

$f_1(y_1) = c_1/p_1 \cdot y_1$ pour $y_1 = 0, p_1, 2p_1, \ldots k_1 p_1$ ($k_1 = \min(q_1, [P/p_1])$

$f_2(y_2) = \text{MAX}[f_1(y_1) + c_2/p_2.(y_2 - y_1)]$

Le domaine de variation de y_1 est fixé par le fait que x_2 est un entier positif ou nul et inférieur à q_2 ; d'autre part on a :

$y_1 \geq 0$ soit $y_2 - p_2 x_2 \geq 0$ donc $[y_2/p_2] \geq x_2$

posons $k_2 = \min(q_2, [y_2/p_2])$ le calcul de $f_2(y_2)$ revient à :

$f_2(y_2) = \text{MAX}[f_1(y_2 - p_2 x_2) + c_2 x_2]$ pour $x_2 = 0, 1, \ldots, k_2$

Et d'une manière générale :

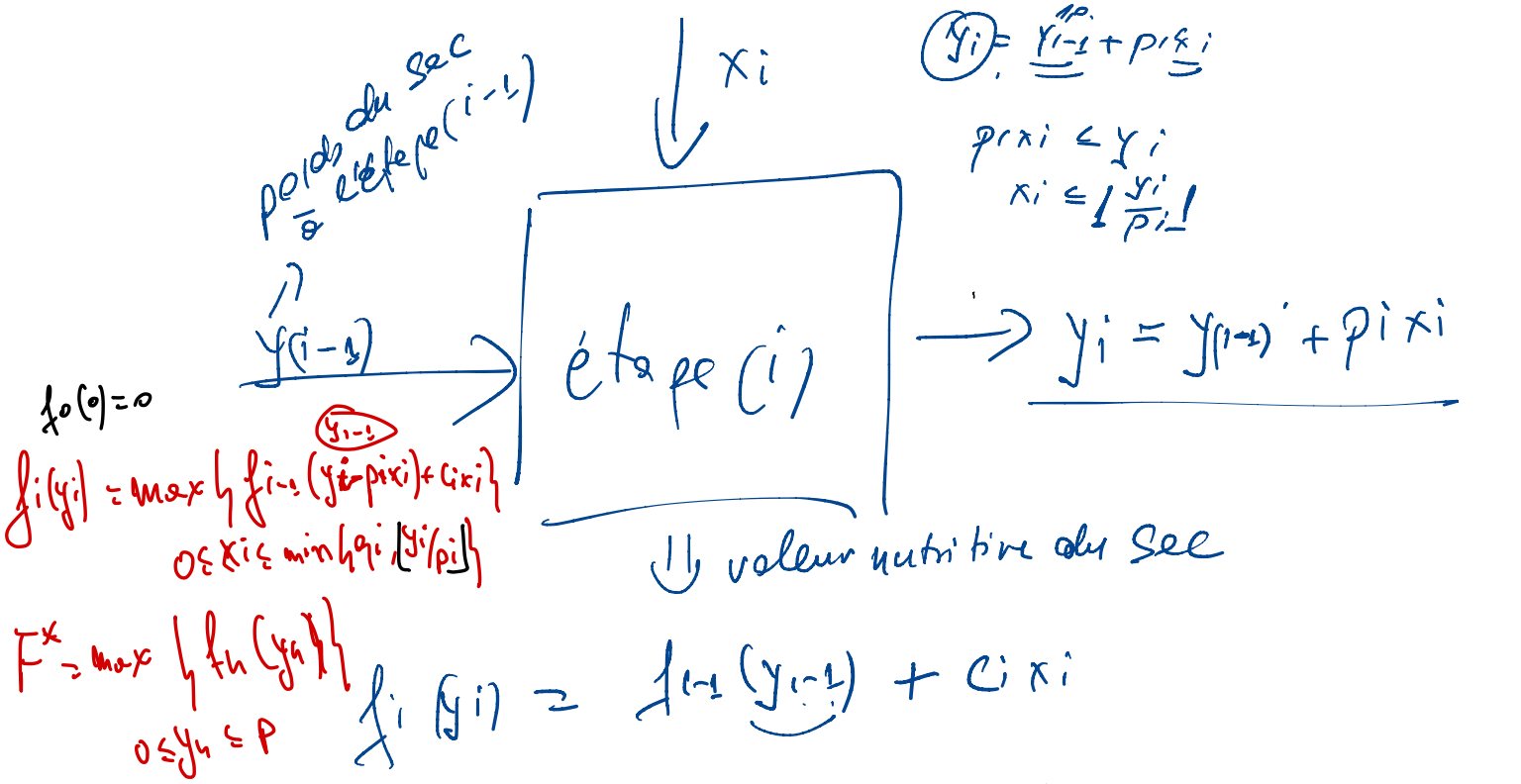
$f_i(y_i) = \text{MAX}[f_{i-1}(y_i - p_i x_i) + c_i x_i]$ pour $x_i = 0, 1, \ldots, k_i$ et $k_i = \min(q_i, [y_i/p_i])$

Pour i = n, la valeur nutritive maximale du sac chargé à y_n Kg est :

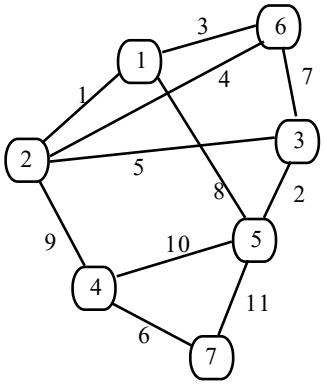
$G^*(y_n) = \text{MAX}[f_{n-1}(y_n - p_n x_n) + c_n x_n]$ pour $x_n = 0, 1, \ldots, k_n$

Q5.

La méthode revient à calculer le chemin de valeur maximale dans un graphe sans circuits. On peut ainsi appliquer l'algorithme de Bellman ce qui revient à une complexité de O(n*P^2). La méthode énumérative nécessiterait de numéroter l'ensemble des solutions réalisables (remplissage du sac avec les aliments) ce qui revient à une complexité de O(P^n).



PROBLEME: ALGORITHME DE KRUSKAL:



L'algorithme de KRUSKAL permet de déterminer un arbre couvrant de coût minimal d'un graphe valué **connexe** G=(X,U, V) à N sommets et M arêtes. **Il consiste à partir d'un graphe réduit à ses sommets isolés, puis à ajouter pas à pas les arêtes en coûts croissants sous réserve que l'arête ajoutée ne crée pas de cycle.**

On supposera que les coûts sont tous distincts.

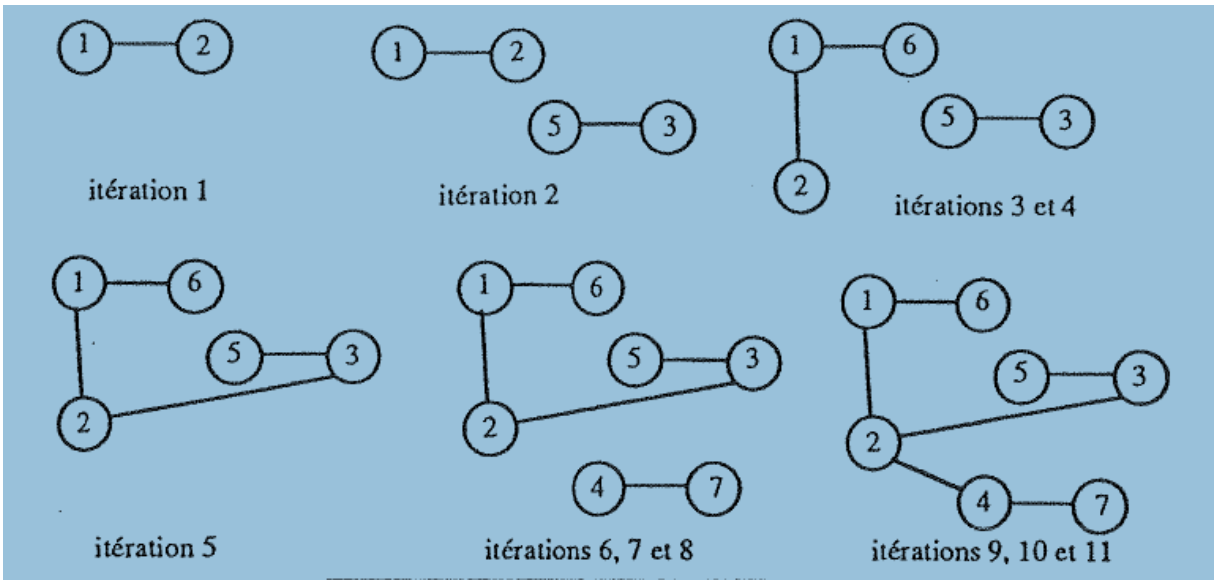
KRUSKAL

{par convention l'arête e_L s'écrira e_L =[I,J] avec I strictement plus petit que J}

```
Début
    Indicer les arêtes en ordre de coûts croissants: v(e1) ≤ v(e2) ≤ ... ≤v(eM)
    Pour H:=1..N faire  CONNEXE(H):=H;
    K:=1; {le graphe partiel est initialement vide}
    Pour L:=1..M faire
      Début
          soit eL=[I,J];
          Si  (CONNEXE(I) différent de CONNEXE(J)) alors
          Début
             AUXI:=CONNEXE(J); fK:=[I,J]; { fK  est mis dans le graphe partiel}; K:=K+1;
             Pour H:=1..N faire
             Si (CONNEXE(H)=AUXI) alors CONNEXE(H):=CONNEXE(I)
          Fin
      Fin
Fin
```

PROBLEME: ALGORITHME DE KRUSKAL – correction

Question 1: Faire tourner l'algorithme sur le graphe ci-dessus: on dessinera les graphes partiels successifs et les onze valeurs successives du tableau CONNEXE obtenu avant chaque itération de la boucle externe.



Question 2: Evaluer en fonction de N et M la complexité de l'algorithme. On pourra distinguer deux types d'itérations dans la grande boucle.

La phase « Indicer les arêtes en ordre de coûts croissants: $v(e_1) \leq v(e_2) \leq \ldots \leq v(e_M)$ »

COUTE $O(M\log(M)) \sim O(M\log(N^2)) \sim O(2*M\log(N)) \sim O(M\log(N))$

La boucle principale contient M itérations dont seulement N-1 rentrent dans la boucle intérieure de complexité O(N). Donc, complexité de la boucle principale :

M-N+1 itérations de complexité O(1) + N-1 itérations de complexité O(N) ce qui donne $O(M+N^2)$ ou $O(N^2)$, donc Complexité globale : $O(N^2)$ + $O(M\log(N))$

Question 3 (cette question est indépendante de l'algorithme)

On rappelle que l'on est dans le cas où tous les coûts sont distincts.

On va démontrer le théorème suivant :

« L'arbre de cout minimal d'un graph connexe existe et il est formé des arêtes de cout minimal des cocycles de G. »

a) Soit A un arbre couvrant de coût minimal et $\omega(Y)$ un cocycle de G. Montrer par l'absurde que A rencontre $\omega(Y)$ en son arête de coût minimal. Un arbre de coût minimal existe puisque le graphe est connexe.

Preuve : notons avec A l'arbre de cout minimum et B l'ensemble des arêtes de cout min de tous les cocycles du graphe. Démontrer 3.a) revient à démontrer que B est inclu dans A.

Soit w(Y) un cocycle quelconque et (i,j) son arête de cout min, donc (i,j) fait partie de B. Raisonnons par l'absurde, supposons que (i,j) ne fait pas partie de A. Rajouter alors (i,j) à A formerait un cycle qui couperait w(Y) dans un autre arc du cycle (et donc de A) que nous notons (p,q). Il est facile alors de constater que le cout de A et strictement supérieur du cout de A'= A + (i,j) - (p,q), donc en contradiction avec la minimalité de A.

b) Réciproque : Soit A un arbre couvrant de coût minimal et [i,j] une arête de A, montrer que (i,j) est l'arête de coût minimal d'un cocycle de G que l'on identifiera.

Vice-versa, démontrons que A est inclus dans B. Soit (i,j) un arête dans A. Enlever (i,j) de A donne une partition de l'arbre dans deux parties, dont une est notée Y. Il est facile de constater que (i,j) fait partie de w(Y) et il est nécessairement de cout min car sinon on remplacerait dans A (i,j) par un autre arc de cout inférieure et on obtiendrait un autre arbre de cout inférieure à celui de A.

c) En déduire une méthode pour construire un arbre de coût minimal.

Il suffit de construire B…pour cela prendre un cocycle quelconque et prendre son arête de valeur min, ensuite une autre et ainsi de suite jusque' il y a n-1 arêtes différents, ou que cela forme un arbre (ce qui est équivalent dans ce cas).

Question 4: Montrer que l'algorithme détermine l'arbre couvrant de coût minimal. On montrera qu'à chaque itération l'algorithme choisit l'arête minimale d'un cocycle de G.

Reponse : Considérons une itération de l'algo ; soit (i,j) l'arête choisi dont Connexe [i] est différente de Connexe [j]. Notons Y= sommets de la composante du Connexe [i] et remarquons que (i,j) est l'arête de cout min de w(Y) et cela est du a l'ordre d'examen des arêtes. Donc l'algo choisit des arêtes dans B et s'arrete quand on forme un arbre couvrant. La validité découle de la question 3.

Que ferait l'algorithme si le graphe n'était pas connexe ?

On l'applique a chaque composante connexe.