Corrigé DS 4

Exercice 1

1)
```
let evalue c v = if (Array.length c = 0) then true
    else begin
    let n = Array.length c in
    let res = ref false in
    for i = 0 to n-1 do
        match c.(i) with
        | V(j) when v.(j) -> ref := true
        | NV(j) when (not (v.(j))) -> ref := true
        | _ -> ()
    done
    !res
    end
```

2) Si on renvoie None alors on n'a pas trouvé de valuation satisfaisant notre formule au bout de k tentatives mais cela ne signifie pas forcément que la formule n'admet pas de modèle car on ne les a pas tous testés. C'est un algorithme de Monte Carlo : la complexité est garantie mais pas la correction.

3) $f_1 = [\ ]$ formule vide

$f_2 = [[|\ |]]$ clause vide

$f_3 = [[| V0; V1; V2 |]; [| NV0; NV1; NV2 |]]$ formule sat.

$f_4 = [[| V0; V0 |]; [| NV0; NV0 |]]$ clauses avec répétition non sat.

$f_5 = [[| V0; NV0 |]; [| NV1 |]]$

4 l'appel récursif à test devrait se faire avec f pour entrée et non pas g.

5
```
let nb_sat f v =
  let rec aux g res = match g with
    | [] -> res
    | h::t when (evalue h v) -> aux t (res+1)
    | _::t -> aux t res
  in
  aux f 0
```

La fonction précédente compte le nombre de clauses satisfaites par v dans f.

```
let max_sat f n k =
  let vmax = ref (initialise n) in
  let maxi = ref (nb_sat f (!vmax)) in
  for i = 1 to k-1 do
    let v = initialise n in
    let nb = nb_sat f v in
    if (nb > (!maxi)) then
      begin
        vmax := v
        maxi := nb
      end
  done ;
  !vmax
```

Complexité: k tours de boucle appelant initialise ($O(n)$) et nb_sat ($O(|f|$ ~~xxx~~$)$) on obtient $O(k.|f|)$ avec $O(n) = O(|f|)$

⑥ Si on ~~a~~ avait un algorithme en temps polynomial pour Max-Sat alors on l'utiliserait pour obtenir une valuation pour laquelle il suffirait de vérifier si elle ~~sa~~ satisfait effectivement la formule et ainsi on aurait un algo polynomial pour CNF-SAT qui est NP-complet, or si un pb NP-complet était dans P, on aurait P = NP.

Exercice 2

① (a) (o_0, o_1, o_2) $(p_0 = 1, p_1 = 1, p_2 = 2)$

$(v_0 = 1, v_1 = 1, v_2 = 4)$

$pmax = 2$

L'algo va sélectionner o_0 et o_1 pour obtenir une valeur de 2 alors qu'on aurait pu prendre o_2 pour obtenir 4. Ce n'est donc pas optimal

(b) (o_0, o_1, o_2) $(v_0 = 3, v_1 = 2, v_2 = 2)$

$pmax = 2$

$(p_0 = 2, p_1 = 1, p_2 = 1)$

L'algo va sélectionner l'objet o au lieu de $\{o_1, o_2\}$

Ce n'est donc pas optimal.

(c) L'exemple du (a) prouve la non-optimalité.

(d) On va considérer: $(v_0 = 8, v_1 = 6, v_2 = 4)$ $pmax = 10$

$(p_0 = 9, p_1 = 6, p_2 = 4)$

(2) le pb de décision est le suivant :

<u>entrée</u> : $(v_i)_{1 \leq i \leq n}$ $(p_i)_{1 \leq i \leq n}$ p_{max} v_{seuil}

<u>Sortie</u> : true ssi il existe $(x_i)_{1 \leq i \leq n}$ t.q $\forall i,\ x_i \in \{0,1\}$ et

$$\sum_{i=1}^{n} x_i p_i \leq p_{max} \text{ et}$$

$$\sum_{i=1}^{n} x_i v_i \geq v_{seuil}.$$

Ce pb est dans NP car la donnée des $(x_i)_{1 \leq i \leq n}$ est un certificat de taille polynomiale et la vérification se réduit au calcul de deux sommes.

(3) Il y a 2^n choix possibles pour les valeurs des x_i à tester. On peut élaguer dès que la somme partielle des $x_i p_i$ dépasse p_{max} pour mettre en oeuvre une stratégie de backtracking

(4) Soit $(x_0, \dots, x_{n-1}, S)$ une instance de SubsetSum.

On lui associe $(x_0, \dots x_{n-1}) \to$ valeurs

$$(x_0, \dots, x_{n-1}) \to \text{poids}$$

$$S \to v_{seuil}$$

$$S \to p_{max}$$

On remarque que $(x_0, \dots x_{n-1}, S)$ est une instance positive de SubsetSum ssi $((x_0, \dots, x_{n-1}), (x_0, \dots, x_{n-1}), S, S)$ est une instance positive du problème du sac à dos avec seuil. En effet si on assimile $I \subset [\![0, n-1]\!]$ à une valuation de n variables : $\sum_{i \in I} x_i = \sum_{i=0}^{n-1} x_i y_i$

avec $\forall i \in [\![0, n-1]\!], y_i \in \{0,1\}$ et ici $y_i = 1$ ssi $i \in I$

On a alors

$$\exists I \subset [\![0,n-1]\!] \text{ t.q } \sum_{i\in I} x_i = S$$

$$\text{ssi } \exists y_i \in \{0,1\}^n \text{ t.q } \sum_{i=0}^{n-1} x_i y_i = S$$

$$\text{ssi } \exists y_i \in \{0,1\}^n \text{ t.q } \overbrace{\sum_{i=0}^{n-1} x_i y_i}^{\text{poids}} \leq \underbrace{S}_{pmax} \text{ et } \underbrace{\sum_{i=0}^{n-1} x_i y_i}_{\text{valeurs}} \geq \underbrace{S}_{vseuil}$$

La construction est en temps polynomial donc SubsetSum $\leq_p$ Sac à dos avec seuil.

Exercice 3 : On a montré que le pb du sac à dos avec seuil est dans NP donc par la réduction précédente c'est un pb NP-complet.

1. Select Count(*) from ressources wher type = "cours"

2. Elle renvoie le nom de la ressource la plus récemment chargée avec l'utilisateur qui l'a chargée.

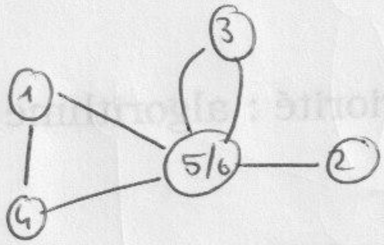
3.

```
select c1.id, c2.id from compte as c1 join
        chargement on id-u = c1.id join
        ressources on id-r = ressources.id join
        compte as c2 on owner = c2.id
```

4. On utilise intersect avec le schéma précédent.

Exercice 4

1



2 On montre par récurrence sur n, le nombre d'arêtes contractées que dans le graphe résultant G_n, on a les propriétés suivantes : (a) $\forall u \in S(G_n)$ $\forall (s,t) \in S_u$ il existe un chemin ds G_0 de s à t composé d'arêtes contractées

(b) $\forall u, v \in S(G_n)$ s'il y a une arête entre u et v alors il existe un chemin de G_0 entre chaque sommet de S_u et chaque sommet de S_v

$n = 0$: a. Les ensembles S_u sont des singletons.

b. évident car $G_n = G_0$

$n = 1$ a. deux sommets dans un supersommet admettent une arête les reliant ds G_0.

b. Pour tous les sommets de G qui sont dans G_0 c'est évident. Si $u = \{a, b\}$ est le nouveau sommet alors s'il y a une arête entre un sommet s et u alors s serait relié à a ou b eux mêmes reliés donc il existe bien un chemin entre s et a et un entre s et b.

Hérédité : G_n satisfait a et b. On considère u_1 et u_2 deux sommets de G_n et $G_{n+1} = G_n/u_1u_2$

a. Si $u \in G_n$ alors le résultat est vrai

Soit $u_1 u_2 \in S(G_{n+1})$ et $(s,t) \in S_{u_1 u_2} = S_{u_1} \cup S_{u_2}$

si $(s,t) \in S_{u_1}^2$ ou $\in S_{u_2}^2$ c'est évident

si $s \in S_{u_1}$, $t \in S_{u_2}$ alors comme u_1 et u_2 sont reliés dans G_n on a par HRb que s et t sont reliés dans G_0

b. Il suffit de regarder $s \in S(G_n)$ et $u_1 u_2$, s'il y a une arête entre deux tels sommets alors il y avait une ~~chemin~~ arête entre s et u_1 ou entre s et u_2 dans G_n et donc un chemin entre tout sommet de S_s et de S_{u_1} (ou S_{u_2}) dans G_0 (HRb), de plus u_1 et u_2 étant reliés dans G_n, par HRb on a aussi l'existence dans G_0 d'un chemin entre u_1 et u_2 et ainsi dans tous les cas on a un chemin entre tout sommet de S_s et tout sommet de S_{u_1} ainsi que de S_{u_2} et donc de $S_{u_1 u_2} = S_{u_1} \cup S_{u_2}$ dans G_0.

3) Soit $S_1 \cup S_2 = S(G_{/st})$ une coupe minimum du graphe contracté. Supposons sans perte de généralité que le supersommet $st \in S_1$ et considérons $S_1^{\setminus\{st\}} \cup \{s,t\} = S_1'$ et $S_2' = S_2$, on a alors $S_1' \cup S_2' = S$

S_1' et S_2' sont non triviaux et $S_1' \cap S_2' = \emptyset$ ainsi

S_1', S_2' donne une coupe pour G.

$$|(S_1', S_2')| = |(S_1, S_2)| \geqslant \text{min cut}(G)$$

4 N_{st} = taille de coupe mini de $G_{/st}$

N = ______________________ G.

- **Supposons:** $N_{st} > N$

soit (S_1, S_2) une coupe minimale pour G
ne contenant pas l'arête (s,t) alors s et t
sont tous les deux dans S_1 ou tous les deux
dans S_2 et si c'est dans S_1 par exemple alors

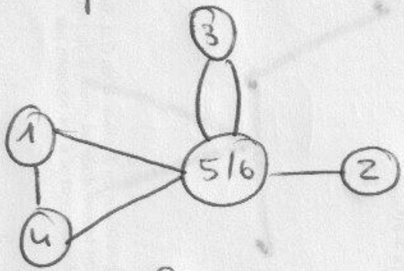
$\left((S_1 \setminus \{s,t\}) \cup \{st\} , S_2 \right)$ est une coupe

pour $G_{/st}$ de même taille que (S_1, S_2)

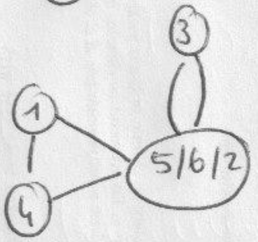
et ceci contredit le fait que $N_{st} > N$.

- Réciproquement, si (s,t) est une arête de toutes les
coupes minimum de G, alors toute coupe où s et t
sont dans le même ensemble n'est pas minimale et
la coupe construite à la q° 3 est de cardinal
$N_{st} > N$ car ne peut pas être minimale dans G.

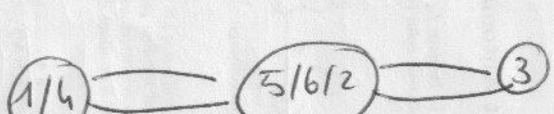
5 étape 1:



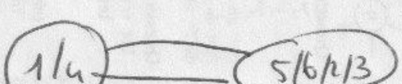
étape 2:



étape 3:



étape 4:



On obtient la coupe $\{1,4\},\{2,3,5,6\}$ de taille 2 alors qu'il existe la coupe $\{2\}$ $\{1,3,4,5,6\}$ qui est plus petite de taille 1. On n'a donc pas obtenu de coupe minimum

6 Chaque sommet apparait dans exactement deux arêtes, ainsi sommer les degrés revient à compter chaque arête exactement deux fois.

7 Soit (S_1, S_2) une coupe minimum de taille k. Chaque sommet de S a un degré $\geq k$ car sinon en l'isolant on obtiendrait une coupe de taille son degré.

Ainsi $2|A| = \sum_{s \in S} d(s) \geq k|S| \Rightarrow k \Rightarrow \frac{2|A|}{|S|}$

8 On peut majorer cette probabilité par le nombre d'arêtes de $|C|$ sur celui de $|G|$ or le nombre d'arêtes de $|C|$ vaut k donc on a au pire $\frac{k}{|A|} \leq \frac{2}{|S|}$

9 $P_R(\text{algo renvoie } C) = P_{|S|}$

$= P(a_1 \notin C \wedge a_2 \notin C \wedge \dots \wedge a_k \notin C)$

$= P(a_1 \notin C) P(a_2 \notin C \mid a_1 \notin C) \cdots P(a_k \notin C \mid a_1 \wedge \dots \wedge a_{k-1} \notin C)$

$\geq \left(1 - \frac{2}{|S|}\right)\left(1 - \frac{2}{|S|-1}\right) \cdots \left(1 - \frac{2}{\underbrace{|S|-k}_{=3}}\right)$

$= \frac{|S|-2}{|S|} \frac{|S|-3}{|S|-1} \cdots \frac{2}{4} \frac{1}{3}$

$= \frac{2}{|S|(|S|-1)}$

(10) Si le multigraphe dispose de $k > \frac{|S|(|S|-1)}{2}$ coupes minimum ≠ alors d'après la q° précédente, chacune a une chance $\geq \frac{2}{|S|(|S|-1)}$ d'être renvoyée par l'algo et alors la proba de renvoyer une coupe minimale est strictement plus grande que 1.

(11) On a vu qu'à chaque tour, la probabilité d'avoir une coupe minimum vaut au moins $\frac{2}{|S|(|S|-1)}$

ainsi la probabilité que l'algo 2 ne renvoie pas une coupe minimum vaut ~~[illegible]~~ la proba qu'à chaque tour il ne trouve pas une coupe minimum ce qui arrive à chaque tour avec proba au plus $1 - \frac{2}{|S|(|S|-1)}$; les tours sont indépendants donc la proba cherchée est inférieure ou égale à $\left(1 - \frac{2}{|S|(|S|-1)}\right)^N$.

(12)

La proba de trouver une coupe minimum vaut au moins $1-\left(1-\frac{2}{|S|(|S|-1)}\right)^N$ avec

$$1-\frac{2}{|S|(|S|-1)} \leq e^{-\frac{2}{|S|(|S|-1)}}$$

et donc $$1-\left(1-\frac{2}{|S|(|S|-1)}\right)^N \geq 1-e^{-\frac{2N}{|S|(|S|-1)}}$$

Soit $c>0$, on veut $1-e^{-\frac{2N}{|S|(|S|-1)}} \geq 1-\frac{1}{N^c}$

$$\iff e^{-\frac{2N}{|S|(|S|-1)}} \leq \frac{1}{N^c}$$

$$\iff e^{2N/|S|(|S|-1)} \geq N^c$$

$$\iff \frac{2N}{|S|(|S|-1)} \geq c \log N$$

↳ Je pense qu'il y a une erreur d'énoncé et qu'on ne devrait pas avoir le même N.

(13) Les deux algos sont de type Monte Carlo car peuvent renvoyer un résultat faux mais ont une complexité qui ne dépend pas de l'exécution.

(14)

```
struct Arete{
        int s1;
        int s2;
};
typedef struct Arete  Arete;
struct Graphe {

        int nb_sommets;
       int nb_ar;
     Arete* aretes; };

typedef struct Graphe Graphe;
```

(15)

```
int contracteArete(Graphe G, subset subsets[], Arete a){
     int s1 = a.s1;
    int s2= a.s2;
    int p1= Trouver (subsets, s1);
   int p2= Trouver (subsets, s2);
   if( p1 != p2) {

         Unir (subsets, s1, s2);
         return (-1);
   }
   return 0;
}
```

(16)
```
int compteAretesCoupe (Graphe G, subset subsets[]){

    int taille_c = 0;
    int n = G.nb_arete;
    for (int i=0 ; i<n ; i++) {
        Arete a = G.aretes[i];
        if ( Trouver (subsets, a.s1) == Trouver(subset, a.s2)){
                    taille_c ++
        }
    }
    return taille_c;
}
```

(17) Il faut contracter en comptant le nombre de contractions réussies jusqu'à n'avoir plus que deux parties.

```
int KargerMinCut (Graphe G0) {
        int ns = G0.nb_sommets;
        int na= G0.ar;
        subset* part= (subset*) malloc(ns * sizeof(subset));
        for (int i=0; i< ns; i++) {
            part[i].parent= i;
            part [i]. rang=0;
        }
        int nb_cont = 0;
```

```
while (nb_cont != ns-2) {
    int indice = rand() % na;
    Arete a = G0.aretes[indice];
    int res = contracteArete(G0, part, a);
    if (res == -1)
        nb_cont++;
}
int taille = compteAretesCoup(G0, part);
free(part);
return taille;
}
```

18 J'ai laissé la question mais pour moi cette question n'a pas vraiment de sens car l'algo implémenté n'est pas exactement celui de Karger : on peut tirer des arêtes supposées contractées avec la représentation choisie, ainsi on ne peut pas contrôler le nombre d'appels à contracteArete (qui peut même être infini → rien ne garantit l'arrêt). On a un algo de type Las Vegas avec erreur... en gros il est temps que ça s'arrête car rien ne va plus.