Sujet Centrale 2015 : correction

I. A. 1)

```
let conflit a,b c,d = max a c < min b d;;
```

I. C. 1) Laissé à votre sagacité.
1. C. 2)

```
let construit_graphe couples = let n = Array.length couples in let g = Array.make n [] in
   for i = 0 to n-1 do
      for j = 0 to i-1 do
         if conflit couples.(i) couples.(j) then ( g.(i) <- j::g.(i); g.(j) <- i::g.(j))
      done
   done;
   g;;
```

I. D. 1)
Pour le problème a, la coloration $(0, 1, 2, 0, 1, 0, 0)$ est optimale ; pour le problème b, $(0, 1, 2, 0, 1, 0, 2, 1, 0)$ l'est.
I. D. 2) a)

```
let rec appartient l x = match l with
   | [] -> false
   | y::q -> x=y || appartient q x
```

b)

```
let plus_petit_absent l = let n = ref 0 in
   while appartient l !n do
      n:=!n+1
   done;
   !n;;
```

c)

```
let couleurs_voisins aretes couleurs i =
   let rec aux l acc = match l with
      | [] -> []
      | x::q when couleurs.(x)=-1 || appartient acc couleurs.(x) -> aux q acc
      | x::q -> aux q (couleurs.(x)::acc) in aux aretes.(i) [];;
```

d)

```
let couleur_disponible aretes couleurs i = plus_petit_absent (couleurs_voisins aretes couleurs i);;
```

I. E. 1)
a) $\chi(G) = 1$ (sans arête, il n'y a pas de conflit) et $\omega(G) = 1$ (dès qu'il y a deux sommets, il n'y a pas d'arête entre ceux-ci donc ils ne forment pas une clique).
b) Soit c une coloration. Pour tout $i \neq j \in [\![0, n-1]\!]$, $\{i, j\}$ est une arête donc $c(i) \neq c(j)$ donc c est injective donc il faut au moins n couleurs. Inversement, on put bien colorier un graphe avec n couleurs. On a donc $\chi(G) = n$. En outre, le graphe étant lui-même une clique, $\omega(G) = n$.
I. E. 2) Une coloration sur G induit, par restriction, une coloration sur toute clique donc sur toute clique maximale C. On a donc $\chi(G) \geq \chi(C)$. Une clique maximale est un graphe complet donc, d'après ce qui précède $\chi(C) = \mathrm{Card}(C) = \omega(G)$. Ainsi $\omega(G) \leq \chi(G)$.
I. E. 3)

```
let est_clique aretes xs =
   let rec aux1 l = match l with
      | [] -> true
      | t::q -> aux2 aretes.(t) q && aux1 q
   and aux2 l1 l2 = match l2 with
      | [] -> true
      | y::q -> appartient l1 y && aux2 l1 q
   in aux1 xs;;
```

II. A. Il s'agit de $(0, 1, 2, 0, 1, 0, 2, 1, 0)$.
II. B.

```
let coloration segments aretes = let n = Array.length segments in let c = Array.make n -1 in
   for k = 0 to n-1 do
      c.(k) <- couleur_disponible aretes c k
   done;
   c;;
```

II. C. 1) Elle appartient à au moins c intervalles. Sinon, ce segment n'était en conflit qu'avec $c-1$ segments au plus. Il y aurait donc au plus $c-1$ couleurs non disponibles donc, comme $\mathrm{Card}([\![0, c-1]\!]) = c$, au moins une couleur disponible dans l'intervalle $[\![0, c-1]\!]$; la couleur choisie n'aurait pu être c.
II. C. 2) Soit i, j deux éléments strictement inférieurs à k tels que I_i et I_j sont adjacents à I_k. Notons a_l le minimum du segment I_l et b_l son maximum (pour tout l). Les segments étant par ordonnée par leurs extrémités gauche, $a_i \leq a_k$ et $a_j \leq a_k$. I_i étant en conflit avec I_k, $b_i > a_k$; de même $b_j > a_k$. Ainsi, $\max(a_i, a_j) \leq a_k < \min(b_i, b_j)$ donc I_i et I_j sont en conflit. Ainsi, pour tout $i \neq j$ tel que I_i et I_j sont en conflit avec I_k, I_i et I_j sont en conflit donc sont adjacents. Ainsi, pour tout i, j adjacents à k, il y a une arête entre i et j. Ceci étant vrai pour $i = k$, tous ces éléments en conflit avec I_k forment une clique. En outre, ces intervalles I_i en conflit avec I_k sont au moins c ; la clique qu'ils constituent avec I_k a donc au moins $c+1$ éléments.
II. C. 3) Comme l'ensemble précédent est une clique d'au moins $c+1$ éléments, $\omega(G) \geq c+1$ donc, en vertu de I.E.2, $\chi(G) \geq c+1$.
II. C. 4) D'après les questions précédentes, si c est une couleur utilisée, $\chi(G) \geq c+1$. En particulier, si c_m est la plus grande couleur utilisée, $\chi(G) \geq c_m + 1$.
Reste à montrer que nous avons bien obtenu une coloration : si cet algorithme affecte la même couleur à deux segments I_i et I_j, avec, par exemple, $i < j$, alors I_j aurait dû recevoir pour couleur la plus petite couleur disponible parmi ses voisins, I_i compris ; il était donc impossible de colorier I_j comme I_i en suivant l'algorithme, la couleur de I_i étant indisponible. L'algorithme effectue donc un coloriage avec $c_m + 1$ couleurs où c_m est le plus grand nombre utilisé pour colorier. Comme $\chi(G) \geq c_m + 1$, cet algorithme effectue un coloriage optimal et $\chi(G) = c_m + 1$.
II. D. L'utilisation de "couleurs voisins" sur un sommet x ayant d voisins vérifie pour chaque voisin s'il est dans la liste en construction avant de l'ajouter ; la complexité est donc $\sum\limits_{v} O(d) = O(d^2)$.
L'utilisation de "plus petit absent" sur une liste de taille n effectue une boucle taille au plus n donc chaque passage est en $O(n)$; elle est donc en $O(n^2)$.
La fonction "couleur disponible" sur un sommet x prend donc de l'ordre de $O(d^2)$ opérations. La fonction de coloration applique à chaque k entre 0 et $n-1$ cette dernière fonction. Au total ce fait donc $\sum\limits_{x \in S} O(d(x)^2) = O(nm^2)$.
III. A. $(x_0, x_1, x_4, x_2, x_5, x_7, x_6, x_3)$ est un ordre d'élimination parfait.
III. B. 1)

```
let voisins_inferieurs aretes x =
   let rec aux l x acc = match l with
      | [] -> acc
      | t::q -> if t<x then aux q x (t::acc) else aux q x acc in aux aretes.(x) x [];;
```

III. B. 2)

```
let est_ordre_parfait aretes = let boo= ref true and k = ref 0 in
   while !boo && !k<Array.length aretes do
      boo:=est_clique aretes (!k::(voisins_inferieurs aretes !k));
      k:= !k +1
   done;
   !boo;;
```

III. C. Ceci a été démontré lors que la question II. C. 2.
III. D. 1)
a) x_0 a couleur 0, x_1 couleur 1, x_2 0, x_3 1, x_4 2, x_5 3, x_6 2, x_7 0.
b) Prenons l'exemple $(x_0, x_1, x_4, x_2, x_5, x_7, x_6, x_3)$. Alors x_0 aura la couleur 0, x_1 1, x_4 2, x_2 0, x_5 3, x_7 0, x_6 1, x_3 2.
III. D. 2)

```
let colore aretes = let n = Array.length aretes in let c = Array.make n -1;
   for i = 0 to n-1 do
      c.(i) <- couleur_disponible aretes c i
   done;
   c;;
```

III. D. 3)
a) On choisit la couleur c_i pour x_i. x_i a donc au moins c_i voisins. Par hypothèse l'ensemble de ces voisins auquel on ajoute x_i déjà coloriés forment une clique. On a donc $\chi(G) \geq \omega(G) \geq 1 + c_i$.
b) Cet algorithme utilise un nombre de couleurs égal à $\max\limits_{0 \leq i \leq n-1}(c_i)+1$. De plus, d'après la question précédente, $\chi(G) \geq \max\limits_{0 \leq i \leq n-1}(c_i)+1$.
Reste à montrer que l'algorithme fournit bien une coloration. Supposons que deux voisins I_i et I_j aient même couleur avec $i < j$. Alors au moment du coloriage de I_j, I_i était déjà colorié donc l'algorithme fait en sorte de choisir, pour I_j, une couleur disponible, en particulier une couleur différente de celle de I_i. L'hypothèse est donc absurde. On a donc bien une coloration qui utilise $\max\limits_{0 \leq i \leq n-1}(c_i) + 1$ couleurs et qui est donc optimale en vertu de l'inégalité $\chi(G) \geq \max\limits_{0 \leq i \leq n-1}(c_i) + 1$.
IV. A. 1) Supposons, par l'absurde, qu'un intervalle est inclus dans l'autre. Quitte à réindexer, on peut supposer qu'il s'agit de I_0 et I_1. Quitte à changer l'ordre de parcours du graphe, on peut supposer que $I_0 \subset I_1$. Alors $I_0 \cap I_3 \subset I_1 \cap I_3$ donc, comme $I_0 \cap I_3 \neq \emptyset$, $I_1 \cap I_3 \neq \emptyset$ ce qui est absurde.
IV. A. 2) D'après la question précédente (et l'hypothèse de bornes distinctes), on $\min I_1 < \min I_2 < \max I_1 < \max I_2$ ou $\min I_2 < \min I_1 < \max I_2 < \max I_1$. Supposons être dans le deuxième cas. Alors $\max(\min I_2, \min I_0) < \min I_1 < \min(\max I_2, \max I_0)$ donc $\max(\min I_2, \min I_0) < \min(\max I_2, \max I_0)$ donc $I_2 \cap I_0 \neq \emptyset$, ce qui est absurde. On a donc $\min I_1 < \min I_2 < \max I_1 < \max I_2$.

Par récurrence, on obtient le même résultat pour I_2, I_3.
IV. A. 3) On obtient encore le même résultat pour I_3, I_0. On a donc, en particulier, $\min I_0 < \min I_1$, $\min I_1 < \min I_2$, $\min I_2 < \min I_3$ et $\min I_3 < \min I_0$ ce qui implique l'absrudité $\min I_0 < \min I_0$. Un tel graphe non cordal ne peut donc exister.
IV. B. Soit un graphe d'intervalle $(I_0, \cdots, I_{n-1})$. Supposons qu'il possède un cycle non cordal de longueur $p \geq 4$. Considérons le sous-graphe constitué par ce cycle que l'on réindexe en $(J_0, \cdots, J_p)$. Posons $J'_i = J_i$ pour $0 \leq i \leq 2$ et $J_3 = \bigcup_{i=1}^{p} J_i$. Ce graphe vérifie exactement les hypothèses précédentes, à savoir qu'il s'agit d'un graphe d'intervalle cordal de 4 sommets. Ceci étant absurde, un tel graphe d'intervalle de $n \geq 4$ sommets non cordal ne peut exister.
IV. C. On représente le graphe dont les arêtes correspondent au fait d'avoir été en même temps dans la bibliothèque. Il s'agit d'un graphe d'intervalles (les intervalles de temps passé dans la bibliothèque). Il y a deux cycles de longueurs quatre sans corde dans ce graphe : celui entre Albert, Didier, Bernard et Isabelle ; celui entre Albert, Didier, Edouard et Charlotte. Ces deux cycles sont absurdes d'après la question précédente. Il y a un menteur dans chacun de ces deux groupes. Comme il n'y a qu'un menteur, celui-ci est dans l'intersection des deux groupes : cela peut être Albert ou Didier. En enlevant le témoignage du menteur, on doit enlever toutes les absurdités (puisque les autres ont dit la vérité). En enlevant le témoignage d'Albert, on a toujours le cycle sans corde (Albert, Didier, Isabelle, Bernard). Il y a toujours un menteur en enlevant ce témoignage. Ce ne peut donc être que Didier (et on voit qu'à l'inverse, en enlevant le témoignage de Didier, les deux cycles n'existent plus). Ce malotru de Didier est donc le responsable de cette satanée question !
IV. D. 1)

```
let simplicial (aretes,sg) k =
   let rec aux l = match l with
      | [] -> []
      | y::q -> if sg.(y) then y::aux q else aux q
   in est_clique aretes (k::(aux aretes.(k))) ;;
```

La fonction auxiliaire est en $O(n)$ où n est la longueur de la liste.
La fonction "aux2" de "est clique" est en $O(n_1 n_2)$ où n_1, n_2 sont les tailles de $l1$, $l2$. La fonction "aux1" est en $O(n^3)$ donc "est clique" est en $O(n^3)$.
Ainsi, la fonction "simplicial" est en $O(d(x)^3)$ où $d(x)$ est le nombre de voisins de x. Cette complexité est aussi en $O(n^3)$.
IV. D. 2)

```
let trouver_simplicial (aretes, sg) = let k = ref 0 and n=Array.length sg in
   while !k< n && not(simplicial (aretes,sg) !k) do
      k:=!k+1
   done;
   if !k=n then faiwith "pas de sommet simplicial"
   else !k;;
```

La complexité de cette fonction est en $\sum_{x \in H} O(d(x)^3)$ où H est le sous-graphe considéré. Cela donne une complexité en $O(n^4)$.
IV. D. 3)

```
let ordre_parfait aretes = let n=Array.length aretes in let sg = Array.make n false in
   let rec miroir l acc = match l with
      | [] -> []
      t::q -> miroir q (t::acc)
   and aux k acc = if k=n then miroir acc []
   else let i = trouver_simplicial (aretes,sg) k in sg.(i) <- true; aux (k+1) (i::acc)
   in aux 0 [];;
```

Cette fonction utilise dans une boucle de taille n la fonction "trouver simplicial" ; elle a donc une complexité en $O(n^5)$.
IV. E. 1) En posant $C' = C \backslash \{x\}$ et en considérant $S \backslash C'$, C' n'est pas une coupure par minimalité. Ainsi, il existe un chemin $\mathcal{C}$ entre a et b évitant les sommets de C' mais il n'en existe pas dans en évitant les sommets de C. Ainsi, le chemin $\mathcal{C}$ passe par x ; il est du type $(s_0 = a, \cdots, s_{k-1}, s_k = x, s_{k+1}, \cdots, s_p = b)$. Comme tous les sommets s_j, $j \neq k$ sont différents de x, ils ne sont pas dans C ; ainsi, les sommets $s_0, \cdots, s_{k-1}$ sont dans G_1 et ceux de $s_{k+1}, \cdots, s_p$ dans G_2. x est voisin de $s_{k-1} \in G_1$ et de $s_{k+1} \in G_2$.
De même, y est voisin d'un élément de G_1 et d'un élément de G_2.
IV. E. 2) Soit a_1 un voisin de x dans G_1, a_p un voisin de y dans G_1. Comme G_1 est une composante connexe de H, il existe un chemin $(a_1, \cdots, a_p)$ dans G_1. Ainsi, $(x, a_1, \cdots, a_p, y)$ est un chemin de x à y tel que tous les sommets hors x et y sont dans G_1. Par symétrie des rôles de G_1 et G_2, ceci vaut aussi pour G_2.
IV. E. 3) En considérant le sous-graphe dont les sommets sont ceux de P_1 et de P_2, on obtient un cycle $(x, a_1, \cdots, a_p, y, b_1, \cdots, b_q, x)$. Ce cycle est de taille au moins 4 (un élément de G_1, un de G_2, x et y). D'après le paragraphe précédent, il possède une corde.
Comme les a_i et les b_j sont dans deux composantes connexes distinctes de H, il n'y a aucune arête entre un a_i et b_j ; en outre, par minimalité des chemins P_1 et P_2, il ne peut exister une arête entre les a_i (sinon on pourrait, grâce à cette corde, réduire la longueur de P_1), ni une arête entre a_i, $i > 1$, et x ; de même, il n'existe pas d'arête entre les b_j ni entre un b_j, $j > 1$, et y. La seule corde possible est entre x et y. x et y sont donc reliés.
IV. E. 4) On a montré que pour tout x et y dans C, il y a un chemin entre x et y. Ceci signifie que C est une clique.
IV. F. 1) Si G est complet, tous les sommets sont reliés par une arête. Les voisins d'un sommet forment l'ensemble des sommets du graphe. Tous ces voisins sont reliés par des arêtes donc le sommet considéré est simplicial.
IV. F. 2) Si G possède un sommet, le graphe est complet donc le sommet est simplicial.
Si G possède deux sommets, alors, s'il y a une arête entre eux, ce graphe est complet ; s'il n'y a pas d'arête entre eux, chacun n'a pas de voisin donc est simplicial.
Si G possède trois sommets, alors, s'il n'y a aucune arête, chaque sommet n'a pas de voisin donc est simplicial. S'il y a une arête, les deux sommets reliés forment une clique d'après le cas précédent et le sommet isolé également d'après le premier cas ; il y a bien deux sommets simpliciaux non voisins. S'il y a deux arêtes, il y a par exemple une arête de x_0 à x_1 et une de x_1 à x_2. x_0 (et x_2) a pour seul

voisin x_1. Les deux sommets x_0 et x_1 forment une clique donc x_0 est simplicial ; de même les deux sommets x_2 et x_1 forment une clique donc x_2 est simplicial ; x_0 et x_2 sont donc simpliciaux et non voisins. Supposons que x_0, x_1, x_2 forment un cycle. Alors ce graphe est complet donc tout sommet est simplicial.

IV. F. 3) a) Considérons un cycle $a_0, \cdots, a_p$ un cycle de longueur au moins 4 dans H_1. C'est aussi un cycle dans G ; il admet donc une corde. Cette corde relie deux sommets de H_1 donc c'est une corde de H_1.

b) Tout sommet de H_1 est dans ce cas simplicial. S_1 contient donc un sommet simplicial s_0. Les voisins de s_0 dans G qui ne sont pas dans C sont dans la composante connexe de s_0 dans $S \backslash C$; ils sont donc dans S_1 ; les autres sont dans C ; ce sont donc tous des sommets de $S_1 \cup C$. Les voisins de s_0 dans G sont les mêmes que ceux dans H_1. s_0 est aussi simplicial dans G.

c) H_1 a strictement moins d'éléments que G. D'après $\mathcal{P}(H_1)$, H_1 n'étant pas complet, H_1 possède au moins deux sommets simpliciaux non voisins. Ces sommets sont dans $S_1 \cup C$. S'ils sont tous les deux dans C, ils sont voisins d'après la partie IV. E. Il y en a donc au moins un dans S_1.

Comme dans la question précédente, les voisins dans G de ce sommet sont dans C ou dans la composante connexe de s_0 dans $S \backslash C$ donc dans S_1. Tous ses voisins (dans G) sont donc dans $S_1 \cup C$. Ce sommet est donc aussi simplicial dans G.

d) Si G est complet, alors tous les sommets sont simpliciaux. Si G n'est pas complet, d'après ce qui été fait précédemment, en considérant C une coupure minimale, il existe s_0 simplicial dans S_1. De même, il existe s'_0 simplicial dans S_2. Comme s_0 et s'_0 ne sont pas dans la même composante connexe de $S \backslash C$, ils ne sont pas voisins.

Ceci démontre $\mathcal{P}(G)$. Par récurrence sur $\mathrm{Card}(G)$, ceci démontre la propriété énoncée.

IV. G. On considère un graphe cordal G. Un tel graphe possède un sommet simplicial x_n. On considère alors le sous-graphe induit par $G \backslash \{x_n\}$. Il est encore cordal. On peut choisir un sommet simplicial x_{n-1}.

Supposons avoir construit une suite $(x_{i+1}, \cdots, x_n)$ telle que, pour tout $j \geq i+1$, x_j est simplicial dans $S \backslash \{x_{j+1}, \cdots, x_n\}$. Alors $S \backslash \{x_{i+1}, \cdots, x_n\}$ est cordal donc possède un sommet simplicial x_i.

Par récurrence, on peut donc construire une énumération $(x_0, \cdots, x_n)$ tel que x_i est simplicial dans $S \backslash \{x_{i+1}, \cdots, x_{n+1}\}$ pour tout i.

Alors, pour tout $0 \leq i \leq n$, le graphe induit par $\{x_0, \cdots, x_i\}$ est celui induit par $S \backslash \{x_{i+1}, \cdots, x_n\}$. x_i étant simplicial dans ce sous-graphe, ses voisins d'indice inférieurs forment, par définition, une clique. Cette énumération est donc bien un ordre d'élimination parfait.