Devoir maison n°2

Corrigé

⋆ ⋆ ⋆

1 Castors affairés

Question 1 Le nombre de programmes contenant n caractères est 128^n (qui est bien fini). Malheureusement, on ne peut pas se contenter de tous les exécuter et ne garder que celui qui s'exécute sans erreur, termine son calcul en temps fini et renvoie un entier, le plus grand possible, car certains de ces programmes peuvent avoir une exécution qui ne termine pas.

Question 2 La fonction C est correctement définie, car chaque l'exécution de chaque programme peut soit terminer en temps fini, soit ne pas terminer (il n'y a pas d'entre deux, les programmes contenant des erreurs terminent en temps fini). De plus, le programme contenant n répétitions du caractère `1` est correct, termine en temps fini et renvoie un entier. Ainsi, l'ensemble des entiers renvoyés par des programmes de taille n qui terminent est un ensemble fini, non vide, inclus dans $\mathbb{Z}$, il possède donc un maximum.

Question 3 On peut rajouter une espace à la fin d'un programme sans changer le déroulé de son exécution (l'espace sera ignorée). On en déduit que pour $n \in \mathbb{N}^*$, $C(n+1) \geqslant C(n)$: si on dispose d'un castor affairé de taille n, alors il existe un programme de taille $n+1$ qui renvoie le même entier, qui est donc inférieur ou égal à l'entier renvoyé par un castor affairé de taille $n+1$.

Par ailleurs, la dernière expression d'un castor affairé de taille n peut être complétée par `+1`. Un tel programme modifié terminera toujours son exécution en temps fini et renverra un entier strictement plus grand (on suppose une représentation des entiers sur une mémoire infinie). Sa taille sera exactement $n+2$. On en déduit bien que $C(n+2) > C(n)$.

Question 4 Les seuls programmes à un seul caractère qui renvoient un entier sont des programmes constitués d'un des 10 chiffres. Ainsi, $C(1) = 9$. Cela ne contredit pas la non-calculabilité, car cela ne donne pas de méthode générale permettant de calculer $C(n)$ pour tout entier n.

Question 5 Notons $m = \lfloor \log_{10} n \rfloor$, et $n = (c_m c_{m-1} \ldots c_1 c_0)_{10}$ l'écriture décimale de n. f étant calculable, il existe un programme OCaml commençant par `let f x =` permettant de définir une fonction calculant $f(x)$, dont l'exécution termine toujours. Notons k_0 la taille de ce programme. En le complétant par `f`$c_m c_{m-1} \ldots c_1 c_0$, on obtient un programme de taille $k_0 + 2 + m + 1$, dont l'exécution termine toujours et qui calcule $f(n)$. En posant $k = k_0 + 3$, on obtient un programme de taille $m + k$ qui renvoie un entier, donc cet entier est inférieur ou égal à l'entier renvoyé par un castor affairé de taille $m + k$, soit $f(n) \leqslant C(m+k)$.

Question 6 On a $\dfrac{n}{\lfloor \log_{10} n \rfloor + k + 2} \underset{n \to +\infty}{\longrightarrow} +\infty$, donc pour n_0 assez grand, $n_0 \geqslant \lfloor \log_{10} n_0 \rfloor + k + 2$. Pour un tel n_0, par la question 3, on a :

$$f(n_0) \leqslant C(\lfloor \log_{10} n_0 \rfloor + k) < C(\lfloor \log_{10} n_0 \rfloor + k + 2) \leqslant C(n_0)$$

Soit $f(n_0) < C(n_0)$.

Question 7 La question précédente montre que $f \neq C$. Comme ce résultat est valable pour toute fonction calculable, on en déduit que C n'est égale à aucune fonction calculable, donc n'est pas calculable elle-même.

Question 8 Supposons que le problème `Arrêt` est décidable. Alors le programme suivant pourrait calculer $C(n)$, pour tout n :

- on énumère tous les programmes OCaml de taille n ;

– on décide, à l'aide d'un programme qui résout `Arrêt`, lesquels ont une exécution en temps fini ;
– on exécute les programmes qui terminent à l'aide d'une machine universelle, en gardant en mémoire les valeurs renvoyées, si ce sont des entiers ;
– on renvoie le maximum de tous ces entiers.

Ainsi, le problème du castor affairé serait calculable. Par la question précédente, on en déduit par l'absurde que `Arrêt` n'est pas décidable.

2 Graphes d'intervalles

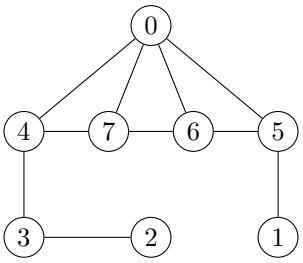
2.1 Représentations des ensembles de tâches

Question 9 Pour $\mathcal{T}_1$, $(1, 0, 1, 2, 2, 1, 0)$ est un ordonnancement optimal. Il utilise 3 machines. Il n'est pas possible d'utiliser strictement moins de machines, car pendant l'intervalle $[3, 4]$, trois tâches sont exécutées simultanément.

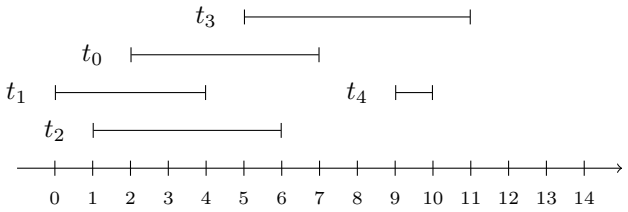
Pour $\mathcal{T}_2$, $(1, 0, 0, 1, 0, 2, 0, 2)$ est un ordonnancement optimal. Il utilise 3 machines. Il n'est pas possible d'utiliser strictement moins de machines, car pendant l'intervalle $[6, 7]$, trois tâches sont exécutées simultanément.

2.2 Lien avec les graphes

Question 10 On obtient le graphe suivant :

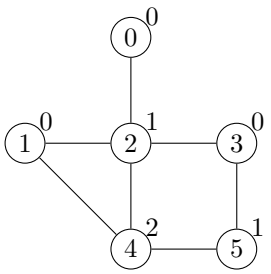


Question 11 Le schéma suivant correspond à une réalisation de ce graphe :



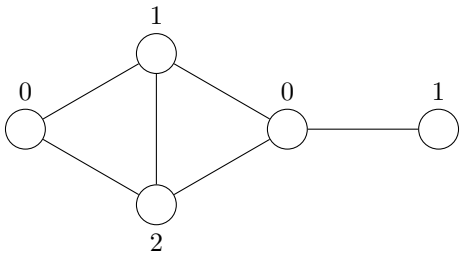
2.3 Coloration de graphes

Question 12 On a $\chi(G_0) = 3$. En voici une coloration. Il n'est pas possible d'utiliser moins de couleurs, car les sommets 1, 2 et 4 sont deux à deux adjacents (donc doivent avoir des couleurs distinctes).

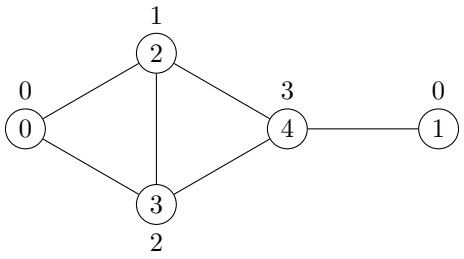


Question 13 On obtient la coloration donnée à la question précédente. Elle est donc optimale.

Question 14 On commence par remarquer que $\chi(G_2) = 3$, car il existe une clique (un sous-graphe complet) de taille 3, et que la coloration suivante est une 3-coloration :



Or, si on indexe les sommets de la manière suivante, l'algorithme glouton renverra la coloration suivante, qui est une 4-coloration :



2.4 Coloration de graphes d'intervalles

Question 15 Soit $r = (t_0, \ldots, t_{n-1})$ la réalisation de G. Alors :

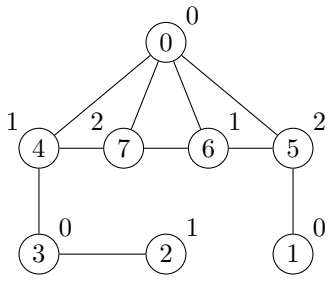
$$\begin{aligned} \textit{ordo} \text{ est cohérent pour } r &\Leftrightarrow \text{pour tout } \{i,j\} \subseteq S, \textit{ordo}(i) = \textit{ordo}(j) \text{ implique } t_i \cap t_j = \emptyset \\ &\Leftrightarrow \text{pour tout } \{i,j\} \subseteq S, \textit{ordo}(i) = \textit{ordo}(j) \text{ implique } \{i,j\} \notin A \\ &\Leftrightarrow \text{pour tout } \{i,j\} \subseteq S, \{i,j\} \in A \text{ implique } \textit{ordo}(i) \neq \textit{ordo}(j) \\ &\Leftrightarrow \textit{ordo} \text{ est une coloration de } G \end{aligned}$$

Question 16 Si on choisit comme ordre de traitement l'ordre correspondant à l'ordre croissant des dates de début d'intervalles, alors l'algorithme glouton de coloration renverra une coloration optimale.

En effet, dans l'ordonnancement associé (qui est cohérent d'après la question précédente), le nombre de machines utilisées à chaque instant correspond au nombre d'intervalles qui s'y recoupent. L'ordonnancement n'utilisera donc jamais plus de $\chi(G)$ machines (le nombre maximum d'intervalles qui se recoupent à un instant donné). On en déduit que la coloration est bien optimale.

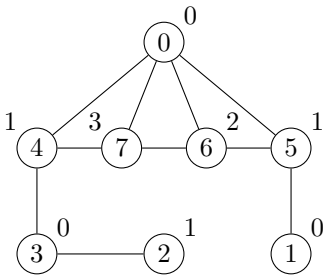
Question 17 On obtient l'ordre $\sigma = (0, 4, 7, 6, 5, 3, 1, 2)$.

Question 18 En appliquant l'algorithme glouton de coloration, on obtient la coloration suivante :



Il s'agit d'une 3-coloration.

L'ordre obtenu avec le parcours en largeur classique aurait été $(0, 4, 5, 6, 7, 3, 1, 2)$. La coloration obtenue avec l'algorithme glouton aurait alors été :



Il s'agit d'une 4-coloration, qui n'est donc pas optimale.

Question 19 On dit qu'un sommet est *simplicial* (pour l'ordre σ) si tous les sommets qui le précèdent forment une clique. Cela revient à dire qu'un ordre est simplicial si tous les sommets sont simpliciaux pour cet ordre. Pour simplifier la rédaction de la preuve, on notera $s \prec t$ si le sommet s précède t dans σ.

Soit σ l'ordre obtenu par l'algorithme LexBFS appliqué à un graphe d'intervalle $G = (S, A)$ d'ordre n. Dans l'hypothèse où il existe des sommets non simpliciaux, on va montrer qu'on va pouvoir construire une suite infiniment décroissante pour $\prec$.

En effet, soit s_0 le sommet non simplicial minimal pour $\prec$, et soient $s_2 \prec s_1$ deux voisins de s_0, non adjacents, et minimaux pour $\prec$. Comme $s_1 \prec s_0$ mais $\{s_2, s_0\} \in A$, selon le principe du parcours en largeur lexicographique, il existe $s_3 \prec s_2$ tel que $\{s_3, s_1\} \in A$ mais $\{s_3, s_0\} \notin A$ (sinon s_0 serait traité avant s_1 car le traitement de s_2 lui donnerait une étiquette plus grande que celle de s_1). Sachant qu'un graphe d'intervalle ne contient pas de cycle de taille $\geqslant 4$ sans corde, on en déduit que $\{s_3, s_2\} \notin A$ (sinon le cycle $(s_3, s_1, s_0, s_2, s_3)$ serait sans corde). De plus, on peut choisir s_3 vérifiant ces propriétés et minimal pour $\prec$. On va montrer qu'on peut continuer la suite de cette manière.

Pour $n \in \mathbb{N}$, supposons $s_0, s_1, \ldots, s_{n+2}$ construits tels que :

- $\forall i \in [\![1, n+2]\!]$, s_i est simplicial ;
- $\forall (i, j) \in [\![0, n+2]\!]^2, i > j \Rightarrow s_i \prec s_j$;
- $\forall (i, j) \in [\![0, n+2]\!]^2, \{s_i, s_j\} \in A \Leftrightarrow |j - i| = 2$ ou $\{i, j\} = \{0, 1\}$;
- $\forall i \in [\![1, n+2]\!]$, s_i est minimal pour $\prec$ parmi les sommets vérifiant ces propriétés.

Les hypothèses sont résumées par le schéma ci-dessous, les sommets étant placés en respectant l'ordre donné par LexBFS :

Toujours selon le principe de LexBFS, il existe un sommet $s_{n+3} \in S$ tel que $s_{n+3} \prec s_{n+2}$, $\{s_{n+3}, s_{n+1}\} \in A$ et $\{s_{n+3}, s_n\} \notin A$ (sinon s_n serait traité avant s_{n+1}). Choisissons s_{n+3} minimal pour $\prec$ parmi les sommets vérifiant ces propriétés. Montrons qu'alors, $\forall i \in [\![0, n]\!], \{s_{n+3}, s_i\} \notin A$. On ne traite ici que le cas où n est pair, mais le cas où n est impair est symétrique. La preuve est la suivante :

- si $n - 2 > 0$ et $\{s_{n+3}, s_{n-2}\} \in A$, alors par hypothèse, s_{n-2} est simplicial, donc tous ses voisins sont adjacents. Comme s_n est adjacent à s_{n-2}, cela impliquerait $\{s_n, s_{n+3}\} \in A$, ce qui est absurde. Par une récurrence rapide, on montre de même que $\{s_{n+3}, s_{n-2k}\} \notin A$ pour $k \in [\![0, \frac{n}{2} - 1]\!]$;
- d'après ci-dessus, $\{s_{n+3}, s_2\} \notin A$. Si on suppose $\{s_{n+3}, s_0\} \in A$, alors sachant que $s_{n+3} \prec s_1$, cela contredirait la minimalité de s_1. Par l'absurde, on conclut que $\{s_{n+3}, s_0\} \notin A$;
- si on suppose $\{s_{n+3}, s_1\} \in A$, alors s_{n+3} est adjacent à s_1, mais pas à s_0. Sachant que $s_{n+3} \prec s_3$, cela contredirait la minimalité de s_3. Par une récurrence rapide, on montre de même que $\{s_{n+3}, s_{n+1-2k}\} \notin A$ pour $k \in [\![0, \frac{n}{2}]\!]$.

On fait la preuve dans ce sens pour qu'au moment de contredire la minimalité d'un sommet, toutes les hypothèses d'adjacence et de non-adjacence soient bien les mêmes.

Finalement, $\{s_{n+3}, s_{n+2}\} \notin A$, car sinon le graphe admet un cycle sans corde (on vient de montrer qu'aucune corde ne peut exister).

On a bien prouvé l'existence d'un nouveau sommet à rajouter à la suite, et donc par récurrence l'existence d'une suite infiniment strictement décroissante. C'est absurde car il existe un nombre fini de sommets. On conclut par l'absurde que tous les sommets de σ sont simpliciaux et donc que σ est simplicial.

Question 20 Soit σ l'ordre donné par LexBFS dans $G = (S, A)$ un graphe d'intervalle. Il existe une clique de taille $\omega(G)$ dans G. Notons $s_{\max}$ le sommet de cette clique maximal pour σ. Montrons par récurrence que l'application de l'algorithme glouton de coloration renvoie une $\omega(G)$-coloration lorsqu'il parcourt les sommets selon σ, c'est-à-dire que la couleur maximale utilisée est $\omega(G) - 1$:

- le premier sommet de σ reçoit la couleur $0 \leqslant \omega(G) - 1$ (si le graphe est non vide) ;
- supposons le résultat vrai au cours de l'exécution de l'algorithme. Soit s le prochain sommet à colorer. Alors les voisins de s déjà colorés forment une clique (c'est le résultat de la question précédente). s a donc au plus $\omega(G) - 1$ voisins déjà colorés. La plus petite couleur absente parmi les couleurs des voisins de s est donc au plus $\omega(G) - 1$, ce qui assure la conservation de la propriété. De plus, le sommet $s_{\max}$ sera bien coloré avec la couleur $\omega(G) - 1$ (car il a exactement $\omega(G) - 1$ voisins colorés, tous adjacents).

Ceci achève la récurrence. Enfin, comme $\chi(G) \geqslant \omega(G)$, l'existence d'une $\omega(G)$-coloration montre qu'en fait $\chi(G) = \omega(G)$ et donc que la coloration renvoyée est optimale.

⋆ ⋆ ⋆