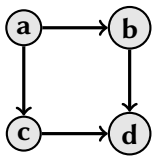
Outils logiques et algorithmiques – TD 5 – Correction

Exercice 1

1. Etats successifs de la pile : [], [A], [A,B], [A,B,C], [A,B,C,D], [A,B,C,D,G], [A,B,C,D,G,F], [A,B,C,D,G], [A,B,C,D,G,H], [A,B,C,D,G], [A,B,C,D], [A,B,C], [A,B], [A,B,E], [A,B], [A], [].
2. Après exécution de `visiter s` l'état de la pile `en_cours` est le même qu'avant l'appel (pendant l'exécution, des sommets ont été d'abord ajoutés puis retirés). Le sommet retiré à la fin est exactement `s`.
3. Lorsqu'un nouveau sommet `s'` est ajouté au sommet de la pile, ledit sommet est un successeur du sommet alors présent au sommet de la pile.
4. Dans un DAG comme dessiné ci-dessous le sommet d est considéré deux fois comme le sommet courant, et est donc déjà marqué lors de la deuxième fois. Le graphe est pourtant acyclique.



   Correction du critère : on a un cycle lorsque le sommet rendant le test vrai est dans la pile `en_cours` (peut se montrer en utilisant la question 3).

Exercice 2

1. L'ensemble `à_explorer` peut grandir ou rétrécir. Le nombre de sommets marqués peut rester stable ou augmenter.
2. On prend un ordre lexicographique avec en première composante le nombre de sommets non marqués, et en deuxième composante le nombre de sommets dans l'ensemble `à_explorer`. En effet, lors de l'exploration d'un sommet dont certains voisins ne sont pas encore marqués on marque ces voisins (et le nombre de sommets non marqués diminue), et lorsque tous les voisins sont marqués le nombre de sommets non marqués reste stable mais le nombre de sommets dans `à_explorer` diminue (on a retiré `s` au début de la boucle, et on n'a ajouté aucun voisin).

Exercice 3

1. On cherche le premier entier pour lequel les valeurs de f, g et h diffèrent (plus précisément, le premier entier pour chaque paire de fonctions).

   Les trois fonctions sont égales sur les arguments 0, 1 et 2. On a en revanche $f(3) = h(3) = 3$ et $g(3) = 0$. Donc $g(3) < f(3)$ et $g(3) < h(3)$ et on en déduit $g < f$ et $g < h$ (avec $l = 3$). Les fonctions f et h restent égales jusqu'à 5. Puis on a $f(6) = 6$ et $h(6) = 0$, d'où $h < f$ (avec $l = 6$). L'ordre complet est donc $g < h < f$.
2. Pour montrer que la relation est transitive on suppose $f < g$ et $g < h$ pour trois fonctions f, g et h arbitraires (il ne s'agit **pas** des trois fonctions de la question 1 !) et on essaie de justifier que $f < h$.

   Si $f < g$ alors il existe un $l \in \mathbb{N}$ tel que $f(l) < g(l) \wedge (\forall k < l, f(k) = g(k))$, et de même si $g < h$ il existe un $m \in \mathbb{N}$ tel que $g(m) < h(m) \wedge (\forall k < m, g(k) = h(k))$. (Attention, rien ne dit que ce l et ce m sont égaux, il faut donc bien garder deux noms distincts)

   On cherche alors un p tel que $f(p) < h(p)$ et $\forall k < p, f(k) = h(k)$.

   Prenons p le minimum de l et de m. De nos deux hypothèses $\forall k < l, f(k) = g(k)$ et $\forall k < m, g(k) = h(k)$ on a donc $\forall k < p, f(k) = g(k) = h(k)$.

   Selon que $l > m$, $l = m$ ou $l < m$ on a en outre $f(p) = g(p) < h(p)$ ou $f(p) < g(p) < h(p)$ ou $f(p) < g(p) = h(p)$, c'est-à-dire dans tous les cas $f(p) < h(p)$.

3. L'ordre est total, c'est-à-dire que parmi n'importe quelles deux fonctions distinctes on aura toujours une plus petite et une plus grande. En effet, si $f \neq g$ alors regardons le plus petit l tel que $f(l) \neq g(l)$. Par définition, on a $\forall k < l, f(k) = g(k)$. En outre, on a soit $f(l) < g(l)$, dont on déduit $f < g$, soit $f(l) > g(l)$, dont on déduit $f > g$.
4. Il faut trouver un l permettant de comparer f_{n+1} et f_n. On prend $l = n$. Par définition des fonctions f_n on a bien $\forall k < n, f_{n+1}(k) = 0 = f_n(k)$, et $f_{n+1}(n) = 0 < 1 = f_n(n)$.
5. L'ordre n'est pas bien fondé, car la suite des fonctions f_n forme une suite infinie strictement décroissante.

Exercice 4

1. Avec la définition de l'ordre bien fondé comme "toute partie non vide admet un élément minimal". Soit P une partie non vide de $A \times B$. L'ensemble $A' = \{a \in A \mid \exists b \in B, (a, b) \in P\}$ est une partie non vide de A. Comme $<_A$ est un ordre bien fondé, A' admet un élément minimal, qu'on note a_0. L'ensemble $B' = \{b \in B \mid (a_0, b) \in P\}$ est une partie non vide de B. Comme $<_B$ est un ordre bien fondé, B' admet un élément minimal, qu'on note b_0. Alors (a_0, b_0) est un élément minimal de P.

   Justification de la minimalité de (a_0, b_0) *:* soit $(a, b) \in P$ tel que $(a, b) < (a_0, b_0)$. Par inversion, soit $a < a_0$ soit $a = a_0 \wedge b < b_0$.

   — Supposons $a < a_0$. Comme $(a, b) \in P$ on déduit $a \in A'$. Contradiction avec la minimalité de a_0.

   — Supposons $a = a_0$ et $b < b_0$. Comme $(a_0, b) \in P$ on déduit $b \in B'$. Contradiction avec la minimalité de b_0.

   Donc (a, b) est un élément minimal de P.
2. On vérifie que tous les appels récursifs potentiellement déclenchés par $\mathsf{ack}(m, n)$ sont sur des paramètres m' et n' tels que $(m', n') < (m, n)$.

   — Si $m = 0$, aucun appel.

   — Si $m > 0$ et $n = 0$, appel $\mathsf{ack}(m - 1, 1)$, avec $(m - 1, 1) < (m, n)$ car $m - 1 < m$.

   — Si $m > 0$ et $n > 0$, deux appels $\mathsf{ack}(m, n-1)$ et $\mathsf{ack}(m-1, x)$ (avec x le résultat de $\mathsf{ack}(m, n-1)$). On a bien d'abord $(m, n - 1) < (m, n)$, avec $m = m$ et $n - 1 < n$. En outre, quel que soit x on peut affirmer que $(m - 1, x) < (m, n)$, car $m - 1 < m$.

   On a donc systématiquement décroissance stricte pour $<$ à chaque appel récursif. Or cet ordre est bien fondé : la fonction ack termine bien pour tous paramètres positifs.