Corrigé du DS4

Ordonnancement

Partie A – Ateliers de fabrication

A.1

```
# let rec chemin_aux n = function
        | [] -> 0
        | [a] -> fabrique n a
        | a :: q as l -> fabrique n a + transport a (hd q) + chemin_aux (n - 1) q;;
chemin_aux : int -> int list -> int = <fun>

# let chemin l = chemin_aux (list_length l) l;;
chemin : int list -> int = <fun>
```

A.2 On effectue la première tâche dans le premier atelier, les deux autres dans le second, ce qui donne un coût de 8.

A.3 Il y a p^n chemins possibles.

A.4 $t_{min}(k,i)$ est le minimum entre $t_{min}(k-1,i)+t_{k,i}$ et le minimum de $t_{min}(k-1,j)+c_{j,i}+t_{k,i}$, j dans $[\![1,n]\!]\setminus\{i\}$ (d'ailleurs, il n'est pas nécessaire de retirer i, et on peut même supposer que, par convention, $c_{i,i}=0$).

A.5

a

```
# let temps_minimal tab k = let p = vect_length tab -1 in
        let res = make_vect (p + 1) 0 in
                for i = 1 to p do
                        let temp = ref (tab.(i) + fabrique k i) in
                                for j = 1 to p do
                                temp := min !temp (tab.(j) + transport i j + fabrique k i)
                                done;
                res.(i) <- !temp
        done;
res;;
temps_minimal : int vect -> int -> int vect = <fun>
```

b

```
# let rec temps_tab n p =
        if n = 0 then
                make_vect (p + 1) 0
        else
                temps_minimal (temps_tab (n - 1) p) n;;
temps_tab : int -> int -> int vect = <fun>

# let temps n p = let tab = temps_tab n p in let res = ref tab.(1) in
        for i = 2 to p do res := min !res tab.(i) done;
        !res;;
temps : int -> int -> int = <fun>
```

c temps_minimal est de complexité p^2, et est exécuté de l'ordre de n fois par temps_tab, qui est donc de complexité np^2 (on a négligé la dernière étape, consistant à déterminer le minimum d'un tableau de taille p, de complexité p).

Partie B – Théorie des matroïdes

B.1 Supposons disposer de deux indépendants maximaux A et B de cardinaux distincts, par exemple tels que $|A| < |B|$: par échange, il existe $x \in B \setminus A$ tel que $A \cup \{x\}$ soit indépendant, ce qui contredit la maximalité de A.

Tous les indépendants maximaux d'un matroïde (S, I) ont donc le même cardinal.

B.2 Bien sûr, il existe un indépendant I de poids maximal (car toute partie non vide et majorée (ou finie) de $\mathbb{N}$ admet un plus grand élément) : s'il ne comprend pas s_k, on peut, par application itérée de la propriété d'échange, adjoindre $|I| - 1$ éléments de I à $\{s_k\}$, formant un ensemble J de même cardinal que I. Notons s_j l'élément de $I \setminus J$. On a $w(s_k) \geqslant w(s_j)$ (par définition de s_k et par hérédité), et donc

$$w(J) = w(I) + w(s_k) - w(s_j) \geqslant w(I).$$

B.3 Déjà, l'algorithme renvoie par construction un indépendant.

Montrons par récurrence sur $|S|$ (que l'on appellera cardinal du matroïde) qu'il est de poids maximal. L'amorçage pour $|S| = 1$ est évident.

Fixons $n \in \mathbb{N}^*$, supposons le résultat prouvé pour tout matroïde (S', I') de cardinal n. Soit (S, I) un matroïde de cardinal $n+1$. On sait (cf. B.2) que l'un des indépendants de poids maximal comprend s_k. Soit maintenant $S' = I \setminus \{s_k\}$ et I' l'ensemble des éléments de I comprenant s_k, ôtés de s_k. (S', I') est un matroïde, dont l'algorithme glouton fournit un indépendant maximal par hypothèse de récurrence, l'hérédité est prouvée.

Partie C – Retards et pénalités

C.1 Dans l'ordonnancement f, certaines tâches sont effectuées à l'heure, formant un ensemble A, et les autres (en retard) forment un ensemble B. Si l'on propose l'ordonnancement obtenu en concaténant A et B, les tâches de A restent à l'heure, et sont effectuées avant les tâches de B. Si en outre, dans A, on ordonne en un ensemble A' les tâches dans l'ordre des dates limites croissantes, alors chacune des tâches de A' est effectuée à l'heure : c'est évident pour celles effectuées plus tôt que dans A, et, pour les autres, on peut le voir en appliquant le tri bulle.

On a donc bien trouvé un ordonnancement vérifiant l'ordre canonique et dont la pénalité totale est inférieure ou égale à P.

C.2 On obtient successivement

i	1	2	3	4	5	6	7
A	$[T_1]$	$[T_2, T_1]$	$[T_2, T_1, T_3]$	$[T_2, T_4, T_1, T_3]$	$[T_2, T_4, T_1, T_3]$	$[T_2, T_4, T_1, T_3]$	$[T_2, T_4, T_1, T_3, T_7]$
B	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$[T_5]$	$[T_5, T_6]$	$[T_5, T_6]$

C.3 1. $\Rightarrow$ 2. : (par contraposition) Supposons qu'il existe $t \in [\![1, n]\!]$ tel que $N_t(A) > t$: au moins $t+1$ tâches doivent être exécutées avant t, A ne peut pas être indépendant.

2. $\Rightarrow$ 3. : Supposons 2., et ordonnons les tâches dans l'ordre des dates limites croissantes. Pour tout i, il y a au plus i tâches dont la date limite est inférieure ou égale à i, donc toutes les tâches sont effectuées à l'heure.

3. $\Rightarrow$ 1. est évident.

C.4 On peut supposer A et B donnés dans l'ordre canonique. Soit t_0 maximal tel que $N_{t_0}(B) \leqslant N_{t_0}(A)$. Puisque $N_n(B) = |B| > |A| = N_n(A)$, on a $t_0 < n$, et, pour tout t tel que $t_0 < t \leqslant n$, $N_t(B) > N_t(A)$. On a donc $N_{t_0+1}(B) > N_{t_0+1}(A)$. Ainsi, B comprend plus de tâches de date limite $t_0 + 1$ que A : soit T l'une d'entre elles, n'appartenant pas à A, et soit A' obtenu en ajoutant T à A en position i dans l'ordre canonique. Il est clair que les tâches de A' avant T sont à l'heure (car A est indépendant), de même pour T (qui remplace un élément de A de date limite plus tardive). Les tâches suivant T dans A' sont également à l'heure, car B est indépendant et, pour tout t après i,

$$N_t(A') = N_t(A) + 1 \leqslant N_t(B) \leqslant t.$$

Ainsi, d'après l'équivalence entre 1. et 2. à la question précédente, A' est indépendant.

C.5 Chercher un ordonnancement de pénalité minimale, c'est chercher un indépendant de poids maximal (le poids correspondant ici à la somme des pénalités).

Il est clair que le couple (S, I), où $I = [\![1, n]\!]$ et S est l'ensemble des ensembles de tâches indépendants, est un matroïde (l'hérédité est évidente, et l'échange a été démontré à la question précédente). L'algorithme glouton donne donc un indépendant de poids maximal. L'algorithme ordonnance_tâches applique cette stratégie gloutonne, tout en imposant l'ordre canonique. D'après la question B.3, cet algorithme donne bien un ordonnancement de pénalité minimale.