Corrigé CCP 2014 Partie III : Tri par sélection

1. Détaillons l'appel `tri exemple` où `exemple = [3;1;4;2]`:

```
appel tri [3;1;4;2] : renvoie [1;2;3;4]

   appel selection [3;1;4;2] : renvoie (1, [3;4;2])
      appel aux 3 [1;4;2] : renvoie (1, [3;4;2])
         appel aux 1 [4;2] : renvoie (1, [4;2])
            appel aux 1 [2] : renvoie (1, [2])
               appel aux 1 [] : renvoie (1, [])

   appel tri [3;4;2] : renvoie [2;3;4]

      appel selection [3;4;2] : renvoie (2, [4;3])
         appel aux 3 [4;2] : renvoie (2, [4;3])
            appel aux 3 [2] : renvoie (2, [3])
               appel aux 2 [] : renvoie (2, [])

      appel tri [4;3] : renvoie [3;4]

        appel selection [4;3] : renvoie (3, [4])
           appel aux 4 [3] : renvoie (3, [4])
              appel aux 3 [] : renvoie (3, [])

        appel tri [4] : renvoie [4]

           appel selection [4] : renvoie (4, [])
              appel aux 4 [] : renvoie (4, [])

             appel tri [] : renvoie []
```

2. Montrons d'abord que la fonction `aux` vérifie les propriétés suivantes : si `v` $= [v_n, ..., v_1]$ et `aux c v = (m, r)` avec `r` $= [r_p, ..., r_1]$:

   (d) $\forall i, 1 \leqslant i \leqslant n$, $\delta(v_i, m) + card(v_i, r) = card(v_i, v) + \delta(v_i, c)$
   et $card(m, r) + 1 = card(m, v) + \delta(m, c)$

   (e) $n = p$

   (f) $m \leqslant c$ et $\forall i, 1 \leqslant i \leqslant p$, $m \leqslant r_i$

   On montre ces propriétés par récurrence sur $n = len(v)$:

   <u>Initialisation:</u> Si $n = 0$, v est la liste vide et `aux c v = (c,v)` donc le résultat est vérifié.

   <u>Hérédité:</u> On suppose le résultat vrai pour un certain entier n, on démontre qu'il reste vrai si $len(v) = n + 1$. On a donc `v` $= [v_{n+1}, ..., v_1]$ et on considère l'appel `aux c v`:

- Si $c < v_{n+1}$: on appelle `(m,r') = aux c v'` avec `v'` $= [v_n, ..., v_1]$, et on renvoie `(m, r)` avec `r` $= v_{n+1}$ ::`r'`$=[r_{p+1}, ..., r_1]$. Vérifions les propriétés :

$$(e) \text{ Par HR on a } n = p \text{ donc on a aussi } n+1 = p+1.$$

$$\begin{gathered}(f) \text{ Par HR on a } m \leqslant c \text{ et } \forall i, 1 \leqslant i \leqslant p, m \leqslant r_i. \\ \text{De plus } c < v_{n+1} \text{ donc on a aussi } m < v_{n+1} = r_{p+1}.\end{gathered}$$

$$\begin{aligned}(d) \text{ Pour } 1 \leqslant i \leqslant n : \delta(v_i, m) + card(v_i, r) &= \delta(v_i, m) + \delta(v_i, v_{n+1}) + card(v_i, r') \\ &= \delta(v_i, v_{n+1}) + card(v_i, v') + \delta(v_i, c) \text{ par HR} \\ &= card(v_i, v) + \delta(v_i, c). \\ \text{De plus, comme } c \neq v_{n+1}, m \neq v_{n+1} : \quad & \\ \delta(v_{n+1}, m) + card(v_{n+1}, r) &= card(v_{n+1}, r) \\ &= 1 + card(v_{n+1}, r') \\ &= 1 + card(v_{n+1}, v') + \delta(v_{n+1}, c) \text{ par HR} \\ &= card(v_{n+1}, v) + \delta(v_{n+1}, c). \\ \text{Enfin, } card(m, r) + 1 &= card(m, r') + 1 \text{ car } m \neq v_{n+1} \\ &= card(m, v') + \delta(m, c) \text{ par HR} \\ &= card(m, v) + \delta(m, c).\end{aligned}$$

- Si $c \geqslant v_{n+1}$: on montre de même les trois propriétés.

Montrons maintenant les trois propriétés demandées pour la fonction `selection`: si `s` $= [s_n, ..., s_1]$ (avec $n \geqslant 1$) et `(m, r) = selection s` avec `r` $= [r_p, ..., r_1]$, on note que `selection s` = `aux` s_n `s'` où `s'` $= [s_{n-1}, ..., s_1]$ est de taille $n-1$. Alors:

$$\begin{aligned}(a)\ \forall i,\ 1 \leqslant i \leqslant n-1,\ \delta(s_i, m) + card(s_i, r) &= card(s_i, s') + \delta(s_i, s_n) \text{ par } (d) \\ &= card(s_i, s)\end{aligned}$$

$$\begin{aligned}\text{et } \delta(s_n, m) + card(s_n, r) &= card(s_n, s') + 1 \text{ par } (d) \\ &= card(s_n, s)\end{aligned}$$

$$(b)\ n-1 = p \text{ par } (e) \text{ donc } n = p+1$$

$$(c)\ \forall i, 1 \leqslant i \leqslant p\ ,\ m \leqslant r_i \text{ par } (f)$$

3. On montre les trois propriétés par récurrence sur $n = len(s)$:

   <u>Initialisation:</u> Si $n = 0$, `s` est la liste vide donc `r` aussi, le résultat est vérifié.

   <u>Hérédite:</u> On suppose le résultat vrai pour un certain entier n, on montre qu'il reste vrai si `s` est de longueur $n+1$. On appelle `(m,s') = selection s` puis `r' = trier s'` et on renvoie `r = m::r'`.

   $$(a) \text{ On a } len(s') = len(s) - 1 \text{ par la question précédente, } len(r') = len(s') \text{ par HR,}$$
   $$\text{donc } len(s) = len(s') + 1 = len(r') + 1 = len(r).$$

   $$\begin{aligned}(b)\ \forall i, 1 \leqslant i \leqslant n+1 : card(s_i, s) &= \delta(s_i, m) + card(s_i, s') \text{ d'après la question précédente} \\ &= \delta(s_i, m) + card(s_i, r') \text{ par HR} \\ &= card(s_i, r).\end{aligned}$$

   (c) Par HR, `r'` est triée, et m est inférieur ou égal à tous les éléments de `s'` par la question précédente, donc aussi de `r'` par (b), donc `r` est triée.

4. La fonction `aux` se termine car on l'appelle successivement sur des listes dont la taille décroît strictement (diminue de 1 à chaque appel), le cas d'arrêt étant la liste vide.

   La fonction `selection` appelle la fonction `aux` donc elle se termine également.

   Enfin, la fonction `tri` appelle `selection` qui se termine puis fait un appel récursif sur une liste dont la taille décroît strictement (on lui a retiré un élément), le cas d'arrêt étant la liste vide. Donc `tri` se termine.

5. Il n'y a pas de meilleur ou de pire cas, la fonction a toujours un coût en $O(n^2)$. En effet:

   - La complexité de la fonction `aux` vérifie la relation de récurrence $c_n = c_{n-1} + O(1)$ donc est en $O(n)$.

   - Donc la fonction `selection` a également un coût linéaire.

   - La fonction `tri` a une complexité vérifiant la relation de récurrence $c_n = c_{n-1} + O(n)$ (le $O(n)$ étant le coût de `selection`), donc on a bien un coût en $O(n^2)$.