## 6.3 L'inversion d'une liste : un argument de plus

La liste inverse d'une liste $x_1, \ldots, x_n$ est la liste $x_n, \ldots, x_1$. On peut programmer une fonction qui inverse une liste très facilement en utilisant la récursivité, puisque, si la liste `l` est non vide, son inverse est obtenue en ajoutant `l.hd` à la fin de l'inverse de `l.tl`.

```
static List reverse (final List x) {
 if (x == null) return null;
 return add(reverse(x.tl), x.hd);}
```

où la fonction `add` est définie par

```
static List add (final List x, final int y) {
 if (x == null) return new List(y,null);
 return new List(x.hd,add(x.tl,y));}
```

Pour ajouter un élément à la fin d'une liste de `n` éléments, la fonction `add` demande un nombre d'opérations linéaire en `n`. De ce fait, pour inverser une liste de `n` éléments la fonction `reverse` demande un nombre d'opérations proportionnel à `1 + 2 + ... + n`, qui est équivalent à $n^2$, à une constante près.

On appelle *complexité en temps*, ou plus simplement *complexité*, le nombre d'opérations que demande un programme pour s'exécuter, en fonction de la taille des données. Dans le cas de la fonction `reverse` ce nombre d'opérations ne dépend que de la taille `n` de la liste à inverser. Dans d'autres cas, le nombre d'opérations que demande un programme pour s'exécuter n'est pas uniquement fonction de la taille des données. On distingue alors deux notions de complexité : la *complexité dans le pire cas* qui est la fonction qui à `n` associe le nombre maximal d'opérations que demande le programme pour s'exécuter sur une donnée de taille `n` et la *complexité en moyenne* qui est la fonction qui à `n` associe la moyenne arithmétique du nombre d'opérations que demande le programme pour s'exécuter sur toutes les données de taille `n`.

La complexité de la fonction `reverse` est asymptotiquement équivalente à $n^2$, à une constante près, on dit que cette fonction est *quadratique* en la taille de la liste.

Cependant, nous savons qu'il est possible de retourner un paquet de cartes en temps linéaire