Listes et vecteurs

Fonctions élémentaires de traitement de listes

1) Écrire une fonction `longueur : 'a list -> int` qui calcule la longueur d'une liste.

2) Écrire une fonction récursive, puis une fonction itérative, qui concatène deux listes.

3) Écrire une fonction itérative, puis une fonction récursive, qui inverse une liste. Calculer les temps de calcul de vos deux fonctions.

4) Écrire une fonction `recherche : 'a -> 'a list -> bool` qui teste si un élément est présent dans une liste. Écrire une fonction `rang : 'a -> 'a list -> int` qui renvoie 0 si l'élément n'est pas dans la liste et son premier rang d'apparition sinon.

5) Écrire une fonction `insere_croissant: 'a -> 'a list -> int list` qui insère un élément dans une liste triée dans l'ordre croissant (les éléments pourront être de type `int`, `char` ou `strings`). Utiliser cette fonction pour écrire une fonction `trie_liste: 'a list -> 'a list` qui trie une liste dans l'ordre croissant. Quel est le temps de calcul dans le pire des cas ?

6) Écrire une fonction `map: ('a-> 'b) -> 'a list -> 'b list` telle que `map f [n1;n2;...;nk]` renvoie la liste `[f(n1);f(n2);...;f(nk)]`.

7) Écrire une fonction `crible: ('a-> bool) -> 'a list -> 'a list` telle que `crible f [n1;n2;...;nk]` renvoie la liste des éléments `ni` tels que `f(ni)` prenne la valeur `true`.

Fonctions élémentaires de traitement de vecteurs

1) Écrire une fonction itérative, puis une fonction récursive, qui calcule la somme des éléments d'un vecteur d'entiers.

2) Écrire une fonction itérative, puis une fonction récursive, qui calcule le rang d'apparition d'un élément dans un vecteur. En cas d'échec de la recherche, on renverra la valeur -1 ; en cas de réussite, on stoppera le calcul dès que l'élément aura été trouvé.

3) Écrire une fonction qui applique le tri bulle à un vecteur d'entiers.

4) Écrire une fonction qui, appliquée à deux vecteurs v_1 et v_2 triés dans l'ordre croissant, renvoie le vecteur trié obtenu en réunissant v_1 et v_2. Par exemple, pour $v_1 = [|1; 3; 5; 8|]$ et $v_2 = [|1; 2; 4; 10|]$, la fonction devra renvoyer $[|1; 1; 2; 3; 4; 5; 8; 10|]$.

5) Le *tri rapide* d'un vecteur $[|x_0; x_1; \ldots; x_{n-1}|]$ se fait récursivement de la façon suivante :

- on commence par placer x_0 "au bon endroit" dans le tableau, de sorte que les éléments situés avant (resp. après) x_0 soient tous inférieurs (resp. supérieurs) à x_0 ; par exemple, après cette première étape, la liste $[|17; 35; 5; 2; 67; 12; 367|]$ devient, par exemple, $[|5; 2; 12; \mathbf{17}; 35; 67; 367|]$ (ou toute autre vecteur

obtenu en permutant 2,5,12 ou 35,67,367).

- on trie récursivement le début et la fin du tableau, le cas d'arrêt étant obtenu quand il reste 0 ou 1 élément à trier.

Écrire une fonction

`mettre_en_place: 'a vect -> int -> int -> int`

tel que l'appel `mettre_en_place v i j`, où v est un vecteur de longueur n (d'entiers ou de mots), i et j deux indices tels que $0 \leq i < j < n$, met en place l'élément v_i entre les indices i et j et renvoie la position finale de l'élément v_i. Ainsi, avec $v = [|0; 1; \mathbf{17}; \mathbf{35}; \mathbf{5}; \mathbf{2}; \mathbf{67}; \mathbf{12}; \mathbf{367}; 523; 715|]$, $i = 2$ et $j = 8$, le vecteur v sera modifié (par exemple) en $[|0; 1; \mathbf{5}; \mathbf{2}; \mathbf{12}; \mathbf{17}; \mathbf{35}; \mathbf{67}; \mathbf{367}; 523; 715|]$ et la fonction renverra l'entier 5.

Écrire une fonction qui applique le tri rapide à un vecteur (d'entiers ou de mots).

Une variante : la méthode est rapide à condition que l'élément que l'on met en place se situe le plus près possible du milieu du tableau ; écrire une fonction de tri qui utilise l'analyse suivante :

- si la partie du tableau est de taille 2, on échange, si besoin est, les deux éléments ;
- si la taille est au moins égale à 3, on met en place l'élément médian parmi les 3 premiers éléments du tableau, puis on trie récursivement le début et la fin du tableau.

Ainsi, le vecteur $[|51; 1; 22; 33; 7; 12; 37; 44; 11; 25|]$ est transformé (par exemple) en $[|1; 7; 12; 11; \mathbf{22}; 51; 33; 37; 44; 25|]$, la valeur 22 étant bien placée ($1 < 22 < 51$) ; il reste ensuite à trier récursivement les parties $[|1; 7; 12; 11|]$ et $[|51; 33; 37; 44; 25|]$.