TD sur les files

Exercices sur les files

On dispose d'une structure de file, par exemple implémentée à l'aide du module Caml `queue`.

1. Ecrire une fonction renvoyant le dernier élément d'une file.
2. Ecrire une fonction transformant une file en deux files en envoyant un élément sur deux dans chaque file. Exemple : $\langle 1;2;3;4;5\rangle \to \langle 1;3;5\rangle, \langle 2;4\rangle$.
3. Ecrire une fonction prenant en argument une file $\langle a_1; a_2; \ldots; a_n\rangle$ et une fonction f et retournant la file $\langle f(a_1); f(a_2); \ldots; f(a_n)\rangle$.
4. Ecrire une fonction créant une file à partir des éléments d'une liste.
5. Ecrire une fonction renvoyant la liste des éléments d'une file sans modifier celle-ci.
6. **Problème de Josephus**
   Un vieil homme qui a beaucoup d'enfants veut choisir son héritier. Pour cela, il les dispose en cercle en les numérotant de 1 à n. Il fixe un entier k une fois pour toutes. Il en élimine un tous les k en recommençant le processus après chaque éliminé.

   Par exemple, pour $n = 6$ et $k = 4$, on élimine successivement 4,2,1,3,6 et 5 est le vainqueur.

   Où doit se placer l'informaticien de la famille pour être l'héritier ? Ecrire un programme utilisant une file pour résoudre ce problème.

Permutation aléatoire d'une liste

1. Ecrire une fonction `shuffle` prenant une liste en argument et renvoyant une permutation aléatoire de cette liste.

   Mode opératoire suggéré : on pourra écrire une fonction `extrait : int -> 'a list -> int -> 'a * 'a list` telle que `extrait i l n` renvoie le `i`-ème élément de la liste `l` (de longueur `n`) et `l` privée de cet élément et une fonction `choix : 'a list -> 'a list -> int -> 'a list` telle que `choix source but n` rajoute à la liste `but` les éléments de la liste `source` dans un ordre aléatoire. On utilisera la fonction `Random.int` du module `Random`.
2. En déduire une fonction générant aléatoirement une permutation des entiers de 0 à $n-1$.

Mariages stables

On se propose d'implémenter l'algorithme de Gale-Shapley permettant de résoudre le problème des mariages stables.
On se donne deux ensembles H et F de même cardinal n. Chaque élément de H établit une liste de préférence parmi les éléments de F et de même avec chaque élément de F parmi ceux de H.
Un mariage est dit stable lorsqu'il n'existe pas d'éléments x dans H et y dans F tels que x préfère y à l'élément de F avec lequel il est associé et y préfère x à l'élément de H avec lequel il est associé.

1. Justifier sur un exemple simple que le problème des mariages stables n'admet pas de solution unique.

   Le but de l'algorithme est de calculer un mariage stable pour les éléments de H et F à partir des listes de préférence.

L'algorithme débute sans couple marié. On initialise une file `celib` contenant au départ tous les entiers de 0 à $n-1$, et représentant les éléments de H célibataires. Tant qu'elle est non vide, on note i son premier élément et j l'élément de F que préfère i parmi tous ceux qui n'ont pas encore été envisagés. Si j est encore célibataire, on marie i à j. Sinon, on note k le mari provisoire de j. Si j préfère k à i, on remet i dans la file, sinon, on marie i à j et on remet k dans la file.

Dans le programme, on devra manipuler les 3 tableaux suivants :

- un tableau `suiv` de taille n initialisé à la valeur 0 tel que `suiv[i]` indique le premier indice à partir duquel l'élément i de H doit chercher son conjoint dans sa liste de préférence.

- un tableau `mari` initialisé à la valeur -1 tel que `mari[j]` désigne le conjoint de $j \in F$.

- un tableau 2D `rang` tel que `rang[j][i]` est l'ordre de préférence que donne j à i.

2. Ecrire le morceau de code qui crée le tableau `rang` à partir des tableaux `H` et `F`.

3. Ecrire la fonction `gale_shapley` qui prend les tableaux `H` et `F` de type `int array array` en argument et qui renvoie le tableau `mari` de type `int array`.

4. Justifier que la complexité de l'algorithme est en $\mathcal{O}(n^2)$.

5. Tester cette fonction sur des tableaux aléatoires représentant des permutations des entiers de 0 à $n-1$.

Il est à noter que l'algorithme d'affectation des étudiants dans les Grandes Ecoles après les résultats des concours s'apparente à l'algorithme de Gale-Shapley.