TP 11

Programmation: Marche aléatoire, tri par insertion et nombres premiers circulaires

Les trois sections sont indépendantes.

1 Marche aléatoire sur $\mathbb{Z}$

Maple permet de générer un entier de manière aléatoire avec la fonction `rand`. La commande `rand();` fournit aléatoirement un entier positif compris entre 0 et 10^{12}.

Il est possible de générer un entier entre deux bornes *min* et *max* par `rand(`*min*`..`*max*`)`. Dans ce cas, Maple créé une procédure qu'il faut donc nommer. On la lancera ultérieurement pour obtenir un entier aléatoire. C'est ce que nous faisons dans l'exemple ci-dessous :

```
> a:=rand(1..4);
```

a:=**proc**()(**proc**() **option** *builtin*; 391 **endproc**)(6,4,2)+1 **endproc**

```
> a();
```

2

Dans ce paragraphe, on considère la marche aléatoire d'un individu sur l'ensemble $\mathbb{Z}$:

- à l'instant 0, l'individu est à l'abscisse 0.
- à tout instant n, il se déplace d'une unité à gauche ou à droite avec la même probabilité (égale à 1/2).

Pour un entier $n \in \mathbb{N}$, on s'intéresse à l'événement A_n : "l'individu est revenu à la case départ à l'instant n".

1. L'événement A_n est-il possible si n est impair?
2. Calculer la probabilité p_n que l'événement A_n soit vrai.

Exercice 1

3. Écrire une procédure `marche` qui, étant donné un entier $n \in \mathbb{N}$, donne l'abscisse où se situe l'individu à l'instant n.
4. Écrire une procédure `casedepart` (utilisant `marche`) qui, étant donné un entier $n \in \mathbb{N}$, retourne 0 si l'événement A_n n'est pas réalisé et 1 s'il est réalisé.
5. Écrire une procédure `frequence` (utilisant `casedepart`) qui, étant donnés deux entiers n et $N \in \mathbb{N}$, donne la fréquence que l'évènement A_n soit réalisé lors de N trajets.
6. Calculer `frequence(4,N)` pour différentes valeurs de N. Comparer avec la valeur de la probabilité p_4 que l'événement A_4 soit vrai.

2 Tri par insertion

On cherche donc à trier une liste L dans l'ordre croissant, c'est-à-dire tel que $L[1] \leq L[2] \leq \ldots \leq L[n]$, en notant n la longueur de L.

On a vu lors du TP7 un exemple de tri de liste, **le tri par sélection** : pour une liste donnée, on sélectionne l'élément maximal pour le placer dans une nouvelle liste; puis, on recommence l'opération jusqu'à ce que la liste initiale soit vide.

Voici un autre exemple de tri itératif, **le tri par insertion**, basé sur le principe suivant :

- On initialise une liste avec le premier élément $L[1]$.
- On insère correctement le deuxième élément $L[2]$ en fonction de $L[1]$, de sorte que $\{L[1], L[2]\}$ soit ordonné.
- On itère alors le procédé afin d'insérer au fur et à mesure les autres éléments de la liste: $L[3], L[4], ..., L[n]$.

Exercice 2

1. Écrire une procédure `inser` qui, étant donné une liste triée L et un entier naturel k, insère k dans L de tel sorte que la liste obtenue soit triée. Par exemple,

```
> inser([1,3,6,6,9],4);
```

[1,3,4,6,6,9]

2. Écrire alors une procédure `triinser`, utilisant `inser`, prenant en entrée une liste `L` et retournant cette liste triée. Par exemple,

```
> triinser([2,3,6,10,12,4,9,6,16]);
```

[2,3,4,6,6,9,10,12,16]

3 Nombres premiers circulaires

On appelle **nombre premier circulaire** un nombre qui est premier, ainsi que tous les nombres obtenus par permutations circulaires de ses chiffres (c'est à dire les chiffres qui le constituent sont décalés vers la droite).

Par exemple, on vérifie que 197 est un nombre premier circulaire (car 197, 719 et 971 sont premiers).

L'objectif est de créer une procédure qui, étant donné un entier N, donne les nombres premiers circulaires inférieurs à N.

Exercice 3

1. Écrire une procédure `liste` qui, étant donné un nombre $n = a_k 10^k + \ldots + a_1 10^1 + a_0$, construit une liste $[a_k, \ldots, a_1, a_0]$ constituée des chiffres de n mis dans l'ordre croissant. Par exemple,

```
> liste(197);
```

[1,9,7]

2. Écrire une procédure `nombre` qui, étant donnée une liste $L = [a_k, \ldots, a_1, a_0]$, construit le nombre $n = a_k 10^k + \ldots + a_1 10^1 + a_0$ constitué des chiffres contenus dans L. Par exemple,

```
> nombre([1,9,7]);
```

197

3. Écrire une procédure `permutation` qui, étant donnée une liste L, donne une liste construite à partir L par une permutation circulaire de ces éléments. Par exemple,

```
> permutation([1,9,7]);
```

[7,1,9]

4. Écrire une procédure `circulaire`, utilisant les procédures `liste`, `nombre` et `permutation`, qui cherche les nombres premiers circulaires inférieurs à N, pour un entier N donné.