Permutations (méthode lexicographique)

Objectif : écrire toutes les permutations d'une liste.
L'algorithme utilise un sous-algorithme qui « renverse » la liste, c'est-à-dire qui l'écrit dans l'ordre décroissant : $L[n], \dots, L[1]$.

Exemple : On part de 123. On doit retrouver toutes les permutations dans l'ordre lexicographique (= celui du dictionnaire) : 123, 132, 213, 231, 312, 321.

Données :

La liste $L := [1, 2, \dots, n]$.

Principe de l'algorithme

- 1) On détermine l'indice maximum j tel que $L[j] < L[j+1]$ (de sorte qu'à partir de $j+1$, les valeurs décroissent).
- 2) On détermine l'indice maximum k tel que $L[j] < L[k]$ (k est donc élément de $[j+1, n]$).
- 3) On échange $L[j]$ et $L[k]$.
- 4) On renverse $L[j+1], \dots, L[n]$

L'algorithme s'arrête lorsque $j=0$.

Valeur de sortie

La séquence des permutations de L , dans l'ordre lexicographique.

Implémentation MAPLE

```
> permutations:=proc(L::list)
local j,k,n,R,S; R:=L;n:=nops(R);
S:=R;
do # boucle externe
  j:=n-1;
  while(R[j]>=R[j+1]) do
    j:=j-1 ;
    if j=0 then return(S) end if; # terminaison
  end do;
  k:=n;
  while(R[j]>=R[k]) do k:=k-1 od;
  R[j],R[k]:=R[k],R[j];
  R:=[op( R[1..j] ) , op( renverser(R[j+1..n]) )];
  S:=S,R # valeur de sortie
end do; # répétition
end proc;
```

```
> permutations([1,2,3]);
      [1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1],
      [3, 1, 2], [3, 2, 1]
```

Remarque

1. En appliquant cet algorithme à la liste des indices, on obtient des permutations de toute liste.

```
> L:=[Rifi,Fifi,Loulou]:seq([seq(L[i],i=S)],S=permutations(P));
      [Rifi, Fifi, Loulou], [Rifi, Loulou, Fifi],
      [Fifi, Rifi, Loulou],
      [Fifi, Loulou, Rifi],
      [Loulou, Rifi, Fifi],
      [Loulou, Fifi, Rifi]
```

2. MAPLE ne permet pas d'affecter le paramètre formel (L) de la procédure. On effectue donc une copie $R := L$ qu'on peut ensuite modifier.

3. Si la liste de départ n'est pas injective, l'algorithme ne fournit que les permutations « distinctes » :

```
> permutations([1,1,2]);
      [1, 1, 2], [1, 2, 1], [2, 1, 1]
```

Algorithme de renversement

Objectif : ré-écrire une liste dans l'ordre décroissant (des indices) : $[L[n], \dots, L[1]]$.

Exemple : On part de $[1,2,3]$. On obtient $[3,2,1]$.

Données : La liste $L := [1,2, \dots, n]$.

Principe de l'algorithme

On échange le premier et le dernier, puis le second et l'avant-dernier, etc.

Valeur de sortie La liste « renversée ».

Implémentation MAPLE

```
> renverser:=proc(L::list) local j,k,n,R;
R:=L;n:=nops(R); j:=1;k:=n;
while(j<k) do
R[j],R[k]:=R[k],R[j];
j:=j+1;k:=k-1 end do ;
R; end proc;
```

```
> renverser([1,2,3,4]);
      [4, 3, 2, 1]
```