Devoir surveillé

Partitions d'entiers

d'après un sujet de l'E.N.S.

10 décembre 1996 – 2 heures

Une partition d'un entier positif n est une suite $a = (n_1, \ldots, n_s)$ d'entiers tels que $n_1 \geq \ldots \geq n_s > 0$ et $n_1 + \cdots + n_s = n$. Chaque n_i $(1 \leq i \leq s)$ est une *part* de a, et s est le *nombre de parts* de la partition.

Les partitions d'un entier n sont ordonnées comme suit : si $a = (n_1, \ldots, n_s)$ et $a' = (n'_1, \ldots, n'_{s'})$ sont deux partitions de n, on pose $a < a'$ si et seulement s'il existe un entier $i \leq s, s'$ tel que $n_j = n'_j$ pour $j = 1 \ldots i-1$ et $n_i < n'_i$ (ordre lexicographique).

1 Questions préliminaires

- Ecrire une fonction `affiche_list : int liste -> unit` qui affiche une liste d'entiers, séparés deux à deux par un espace, et suivie d'un retour à la ligne, *en respectant l'ordre des éléments dans la liste.*
- Ecrire une fonction `affiche_rev : int list -> unit` affichant les éléments d'une liste d'entiers de la même manière, *mais dans l'ordre inverse.* La liste ne sera parcourue qu'une seule fois (et donc en particulier on n'utilisera pas la fonction précédente combinée avec la fonction `rev` !).

2 Enumérations des partitions

- Ecrire une fonction `partitions_m : int -> int -> unit` qui prend en argument deux entiers positifs n et m et qui affiche en ordre décroissant toutes les partitions de n dont toutes les parts sont inférieures ou égales à m (on pourra supposer $n \leq 20$).
- Exemple numérique : $n = 9, m = 4$.
- Ecrire une fonction `partitions : int -> unit` qui prend en argument un entiers positif n et qui affiche en ordre décroissant toutes les partitions de n.
- Exemple numérique : $n = 6$.

Dénombrement. On note $p(n)$ le nombre de partitions de $n > 0$ et on pose $p(0) = 1$. Pour $m > 0$ et $n > 0$, on note $p_m(n)$ le nombre de partitions de n en parts inférieures ou égales à m, et on pose $p_m(0) = 1$.

- Démontrer que

$$p_m(n) = \begin{cases} p_{m-1}(n) & \text{si } m > n \\ p_{m-1}(n) + p_m(n-m) & \text{si } n \geq m > 1 \end{cases}$$

- Ecrire une fonction `p_n : int -> int` qui prend en argument n et qui calcule $p(n)$.
- Exemple numérique : $n = 60$.
- Démontrer que $p_m(n)$ est égal au nombre de partitions de n en au plus m parts.

Enumérations particulières

- ☐ Ecrire une fonction `parts_distinctes : int -> unit` qui affiche en ordre décroissant toutes les partitions de n en parts distinctes.
- ☐ Exemple numérique : $n = 11$.
- ☐ Ecrire une fonction `parts_distinctes_impaires : int -> unit` qui affiche en ordre décroissant toutes les partitions de n en parts distinctes et toutes impaires.
- ☐ Exemple numérique : $n = 30$.

3 Application : comment rendre la monnaie

On considère un appareil qui doit rendre la monnaie. Supposons, pour fixer les idées, que cet appareil contient des pièces de 10, 5, 2 et 1 francs. Si on considère dans un premier temps que cet appareil possède une infinité de pièces de chaque sorte, alors rendre une monnaie de n francs, c'est trouver une partition de n dont les parts sont dans l'ensemble $\{10, 5, 2, 1\}$.

- ☐ Ecrire une fonction `parts_dans_l : int -> int liste -> unit` qui affiche en ordre décroissant toutes les partitions de n dont les parts appartiennent à une liste l donnée en argument. On supposera que la liste l est triée par ordre décroissant. **Attention :** on écrira une solution qui ne teste que des parts appartenant à la liste l.
- ☐ Exemple numérique : donner toutes les façons de rendre 11 francs avec l'appareil ci-dessus.

On ne suppose plus que l'appareil possède une infinité de pièces de chaque sorte mais au contraire que ces pièces sont des ressources épuisables. Pour cela, si les pièces de valeur v disponibles sont en nombre r_v, alors une telle ressource est représentée par le couple (v, r_v) et la totalité des ressources de l'appareil est représentée par la liste de ces tels couples. Ainsi, si on dispose de 2 pièces de 10 francs, 3 de 5 francs, 1 de 2 francs et 3 de 1 franc, alors ces ressources sont représentées par la liste

```
[ (10,2) ; (5,3) ; (2,1) ; (1,3) ]
```

- ☐ Ecrire une fonction `monnaie : int -> (int*int) liste -> unit` qui affiche en ordre décroissant toutes les façons de rendre n francs en disposant des ressources représentée par une liste l donnée en argument. On supposera que la liste l est triée par ordre décroissant des premières projections de ses éléments.
- ☐ Exemple numérique : donner toutes les façons de rendre 17 francs avec 2 pièces de 10 francs, 3 de 5 francs, 1 de 2 francs et 3 de 1 franc.