Programmation efficace – Licence 3 Informatique
Tenir l'équilibre !

Problème

Vous avez m boutons (numérotés de 1 à m) devant vous, chacun correspond à une note de musique. Vous devez choisir un de ces boutons, appuyer dessus (et la note retentit), et recommencer... mais pas n'importe comment ! En effet, pour chaque note i vous avez une certaine fréquence $0 < f_i \leq 1$ à respecter : le « taux de note i » parmi toutes les notes jouées depuis le début doit être proche de f_i. Plus formellement, si on note s_i le nombre de fois où on a choisi la note i (depuis le début), alors on doit avoir $n \cdot f_i - 1 < s_i < n \cdot f_i + 1$ où n est le nombre total de notes depuis le début (c'est-à-dire $n = \sum_{i=1..m} s_i$). Il se peut que l'on soit bloqué à un certain moment (c'est-à-dire qu'appuyer sur n'importe quel bouton entraine la violation d'une des contrainte) mais il est parfois possible, en choisissant habilement les boutons, de pouvoir jouer à l'infini.

Le programme demandé devra donner la taille (c'-à-d. le nombre de notes) d'une suite **maximale** de notes prolongeant **une séquence initiale donnée** (c'est-à-dire une séquence finie de valeurs entre 1 et m décrivant les notes qui ont été jouées jusque là). Cette taille sera un entier ou « infini ».

Formats

Une instance du problème est donnée dans un fichier contenant trois lignes. La première ligne contient l'entier m (le nombre de notes) et un entier k indiquant la longueur de la séquence initiale.
La seconde ligne contient m **entiers** a_1, ..., a_m utilisés pour définir les fréquences avec : $f_i = \frac{a_i}{a_1+...+a_m}$ pour tout $1 \leq i \leq m$.
La troisième ligne décrit la séquence initiale avec k entiers $b_1, \ldots, b_k$ où chaque b_i est un nombre entre 1 et m : b_j indique quelle note a été jouée à la j-ème itération.

Par exemple, pour l'entrée ci-dessous, le résultat attendu est 1 (on ne peut compléter la séquence que par une note et ensuite il y a blocage) :

```
6 5
2 1 6 3 5 3
1 2 5 3 5
```

Et pour l'exemple ci-dessous, on doit obtenir « infini » :

```
6 4
2 1 6 3 5 3
1 2 5 3
```