Résolution du jeu "Le Compte est bon"

Arthur Barjot

January 2022

Contents

1 Introduction, règles du jeu

J'ai produit un programme de résolution du jeu "Le Compte est bon", issu de l'émission "Des Chiffres et des Lettres" ceci en m'appuyant sur les cours que j'ai pu suivre en MP* option informatique au lycée Blaise Pascal.

Le jeu se joue avec des "plaques" (des entiers naturels) et un résultat (aussi un entier naturel, le principe est de combiner nos plaques (en les utilisant qu'une seule fois maximum chacune) avec des opérations arithmétiques, en faisait attention que chaque résultat intermédiaire soit un entier naturel (problème avec la soustraction et la division). Plus précisément, on fournit au joueur (ici au programme):

-un résultat : r, un entier avec r appartenant à [100,999]

-une liste d'entiers (les plaques) : l contenant 6 nombres, dont des éléments de [1,10], qui peuvent chacun intervenir 2 fois maximum ; et des éléments de {25;50;75;100}, qui peuvent chacun intervenir 1 fois maximum, exemple : [1;5;100;7;1;25]

2 Résolution du jeu

2.1 Solution optimale

J'ai produit un programme capable de trouver une solution dite "optimale" à ce jeu, c'est à dire :

-Lors ce que le jeu a une solution (on peut arriver au résultat exacte), on parle de "compte bon", solution utilisera le moins de plaques possible pour atteindre cette solution et donc le moins d'opérations possible. Il renverra également un booléen "true", signifiant que le programme à trouvé le compte bon.

Exemple de résolution lorsqu'il y a compte bon :

```
solution_chiffre_dcdl [301;9;8;1;9;5;1];;
```

Le programme renvoie en 1.1 secondes :

```
sol = S (301, "(((1+1)*((9+8)*9))-5)", true)
```

–Lors ce qu'il n'y a pas de solution exacte, le programme renverra la solution la plus proche du résultat qu'il trouvera et celle utilisant le moins de plaque, il renverra également le résultat auquel il arrive et un booléen "false", signifiant que le compte n'est pas bon.

Exemple de meilleure approche lorsqu'il n'y a pas compte bon :

```
solution_chiffre_dcdl [857;2;50;3;1;1;7];;
```

Le programme renvoie en 0.9 secondes :

```
sol = S (856, "(7+1)*(((50+3)*2)+1)", false)
```

2.2 Solution rapide

J'ai également produit un programme "rapide", ou du moins, beaucoup plus rapide que le précédent. Celui-ci à pour but lors de comptes compliqués, dépassant les règles du jeu (ceci est expliqué dans la partie suivante) de trouver une solution sans trop attendre. En fait le point négatif est qu'il ne renvoie pas une solution optimale, il renvoie la première solution qu'il trouve, celle-ci peut ne pas être visuellement très belle (trop longue alors qu'un humain trouverai rapidement une solution bien plus courte). De plus si le compte n'est pas bon elle ne renvoie pas de résultat approché.

Exemple de résolution rapide :

```
resol_rapide [801;10;50;3;1;1;75];;
```

Le programme renvoie en 0.3 secondes :

```
sol = S (801, "(((1+3)*75)+((10*50)+1))", true)
```

3 Extension de la résolution

Mon programme résout aussi ce jeu avec moins de contraintes :

-On peut lui fournir une listes l d'entiers, de taille quelconque, et d'entiers quelconques avec un nombre d'occurrence de chacun quelconque.

-On peut lui fournir un résultat quelconque avec pour seule condition que ce résultat soit différent de 0.

Attention tout de même, si l'on augmente la taille de l, les temps de calcul sont décuplés, je vous conseille alors d'utiliser le programme "resol_rapide" qui pourra résoudre dans un temps encore acceptable si l n'est pas trop grande, mais il ne renverra pas la solution optimale, seulement la première solution qu'il trouvera.

Exemple de la puissance, l'utilité que peut avoir ce programme, avec cette entrée :

```
resol_rapide [91372; 172;96;268;64;4;22;78];;
```

Le programme renvoie en moins de 20 secondes :

```
sol = S (91372, "((((22*96)-((172+64)*4))*78)+268)", true)
```

Ainsi on voit que mon programme résout dans un temps raisonnable un problème qui aurait était très dur pour un être humain.