Défi c0d1ngUP
Les cuisines Dentrassis

c0d1ngUP team

Le nouveau défi que nous vous proposons provient de l'édition 2025 de c0d1ngUP et s'intitule *Les cuisines Dentrassis*. Le thème de la 11e édition du challenge de programmation était H2G2, la série de romans de Douglas Adams. Par ailleurs, nous fournissons dans cet article des pistes de résolution pour le défi proposé dans le numéro 25 : *Fabrique de spores*.

Nouveau défi — Les cuisines Dentrassis

> *Les Dentrassis forment une tribu de gourmands indisciplinés, un gros tas de mecs sympa que les Vogons avaient depuis peu choisi d'employer aux cuisines sur leurs flottes au long cours, à la condition expresse qu'ils se tiennent strictement à carreau.*
>
> *Le guide galactique (H2G2 1)*, Douglas Adams[1]

Plutôt que de se bousculer dans les cuisines, les Dentrassis opèrent un roulement, et chaque jour, un seul Dentrassis prépare des plats.

Il utilise un seul four, et tente d'optimiser son temps pour finir au plus vite sa besogne. Pour chaque plat, il connaît à l'avance le temps de préparation, et le temps de cuisson. Chaque plat doit être d'abord préparé puis cuit. Le Dentrassis de corvée ne peut préparer qu'un plat à la fois, mais il peut tout à fait préparer quelque chose pendant qu'un autre plat cuit. Par ailleurs, lorsqu'il commence à préparer un plat, il ne s'interrompt pas (sauf éventuellement pour mettre quelque chose au four) et le termine avant de commencer à en préparer un autre, et lorsqu'il met un plat au four, il ne le ressort pas tant qu'il n'a pas fini de cuire. Enfin, il ne peut cuire qu'un plat à la fois.

1. *Le guide galactique (H2G2 1)*, Douglas Adams, 1979. Traduction de Jean Bonnefoy. Folio SF, ISBN 2-07-041568-6.

DOI:10.48556/SIF.1024.26.145

Supposons qu'il doive préparer les trois plats suivants :
– `M` : milkshake de brume tachyonique et de fruits cosmiques ; prép. 15 min, cuis. 5 min ;
– `I` : infusion de feuilles d'anémone lunaire ; prép. 25 min, cuis. 15 min ,
– `S` : shot de plasma stellaire épicé ; prép. 20 min, cuis 20 min.

Il peut commencer par préparer le milkshake, puis le mettre au four pendant qu'il prépare l'infusion. Au moment de mettre l'infusion au four (oui… c'est une infusion qui va au four, c'est pour les Vogons), le milkshake sera cuit. Il prépare alors le shot de plasma, et enfin le met au four (figure 1).

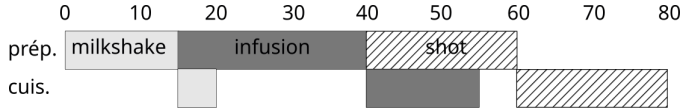


Fig. 1. Premier planning de préparation des 3 plats

En procédant ainsi, au bout de 80 minutes, tout est prêt. Mais il peut s'organiser différemment (figure 2).

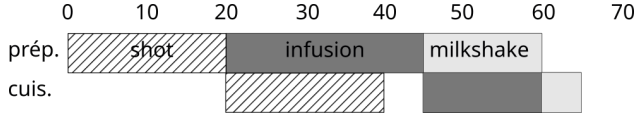


Fig. 2. Deuxième planning de préparation des 3 plats

Et dans ce cas, il sera libre au bout de 65 minutes. Nous donnons aux figures 3 et 4 des plannings qui ne sont pas convenables (ils ne respectent pas les contraintes), si l'on s'en tient aux règles (énoncées plus haut) que s'imposent les Dentrassis.

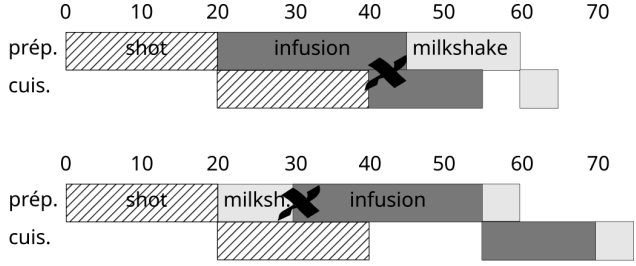


Fig. 3. Deux plannings qui ne respectent pas les règles Dentrassis.

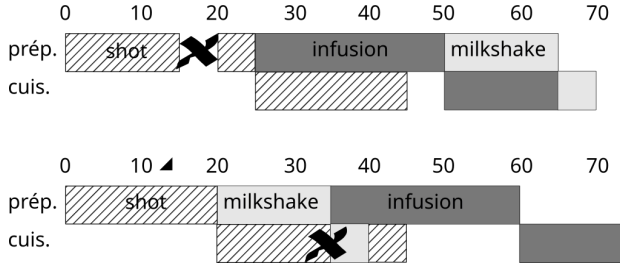


Fig. 4. Deux plannings qui ne les respectent guère plus.

Tous les temps de préparation, ainsi que les temps de cuisson, sont des multiples de 5 minutes, et il est donc possible de décrire un plan de travail en donnant, pour chaque tranche de 5 minutes, deux caractères indiquant quel plat est en cours de préparation et quel plat est en cours de cuisson. Pour la seconde solution, qui ne dure que 65 minutes, on a :

– durant les 5 premières minutes, préparation du shot, et pas de cuisson : `"S."` (en seconde position, un `"."` signifie qu'il n'y a pas de cuisson pendant cette tranche de 5 minutes et, en première position, qu'il n'y a pas de préparation) ;
– idem durant les minutes 5 à 10, 10 à 15 et 15 à 20 : `"S.S.S."` ;
– durant les minutes 20 à 25, l'infusion est en cours de préparation et le shot en cours de cuisson : `"IS"` ;
– …

Toute la séquence est finalement décrite par cette chaîne :

```
S.S.S.S.ISISISISI.MIMIMI.M
```

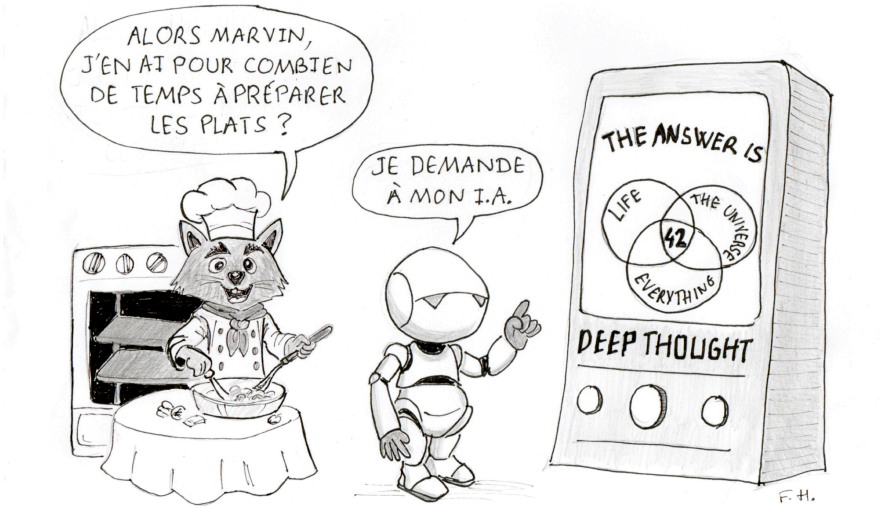
La première séquence, qui dure 80 minutes, serait décrite par :

```
M.M.M.IMI.I.I.I.SISISIS..S.S.S.S
```

Il y a plusieurs séquences différentes qui permettent de terminer le travail en 65 minutes, par exemple : `S.S.S.S.I.ISISISISMIMIMI.M`, mais aucune séquence ne peut libérer le Dentrassis avant 65 minutes.

Pour les remercier de vous avoir pris en stop, vous décidez d'optimiser la prochaine journée pour les Dentrassis. La liste des plats à préparer est disponible en entrée, avec les temps de préparation et de cuisson. Proposez une organisation dans le format utilisé précédemment qui permettra au Dentrassis d'être débarrassé de sa tâche au plus tôt. Attention, chaque couple de lettres du planning vaut pour 5 minutes. Par exemple, le planning `A.BABA.B` signifie :

– minutes 0 à 5 : A en préparation, et four vide ;
– minutes 5 à 10 : B en préparation, A au four ;
– minutes 10 à 15 : B en préparation, A au four ;
– minutes 15 à 20 : aucune préparation, B au four.



Les données d'entrée sont accessibles en téléchargement à cette URL[2] et un extrait est reporté ci-dessous :

```
A : Caviar de poissons de Sandwich IV : prep. 35, cuis. 35
a : Carpaccio de tentacule de poulpe galactique : prep. 5, cui...
B : Beignets de crevettes de la faille de Zorgon : prep. 45, c...
b : Mousseline d'escargots spatiaux de Lamuella : prep. 35, cu...
C : Sushi de kraken cosmique marin : prep. 40, cuis. 30
    ......................
O : Poêlée de bulbes de lunes mordorées : prep. 45, cuis. 40
o : Ratatouille de feuilles d'arbre fractal : prep. 10, cuis. 10
```

Notons que les LLM sont particulièrement doués pour trouver des noms de plats qui auraient pu figurer dans la saga de Douglas Adams…

Vous pouvez nous faire parvenir votre méthode, vos astuces de résolution, vos remarques, accompagnées de la solution au défi, à l'adresse suivante : codingup.team+1024@gmail.com. Dans le prochain bulletin, nous proposerons une synthèse de notre solution et de celles qui nous seront parvenues à temps.

2. `https://codingup.socinfo.fr/1024/26/pb/input_data.txt`.

Solution du défi précédent : Fabrique de spores

Ce problème consistait en un automate cellulaire en 3 dimensions, dont les règles de transition étaient complètement décrites. L'objet du défi consistait à calculer combien de cellules (appelées spores dans le scénario) de chaque type formaient l'automate après la 17e étape de calcul. L'énoncé complet peut être retrouvé dans le numéro précédent de 1024[3].

Commençons par évaluer une solution naïve à ce problème : peut-on représenter l'ensemble des spores jusqu'à la 17e étape. Comme indiqué dans l'énoncé, au départ, on a 8 spores qui forment un cube. À la première itération, 19 nouvelles spores sont déjà créées, donc 27 au total, puis 125 à l'étape suivante, etc. Le nombre de spores après l'étape n est donné par $(2^n + 1)^3$, soit une progression exponentielle[4]. À la 17e étape, on aura près de 2.2×10^{15} spores, qui est une quantité rédhibitoire si l'on envisageait de stocker toutes les configurations, sans parler du temps de calcul nécessaire pour le faire.

Comme alternative, on peut remarquer que l'énoncé ne nous demande « que » de compter le nombre de spores, pas de savoir où elles sont placées. Ainsi, une idée pourrait être, plutôt que de conserver la position de chaque spore, de ne mémoriser que le nombre de cubes de chaque type (il y en a 4^8, soit 65 536 types de cubes). Cette information est suffisante pour déduire le nombre de cubes de chaque type à l'étape suivante, ainsi que le nombre de spores de chaque sorte, qui est l'information qui nous intéresse. Stocker en mémoire une table du nombre de cubes de chaque type à chaque itération devient beaucoup plus commode. Alors, le problème peut se décomposer en trois tâches :

- tout d'abord, créer une table de transition qui, pour chaque type de cube, va donner le type et le nombre de cubes qui vont en découler à l'étape suivante ; c'est cette structure qui va consommer la plus grande quantité de mémoire dans notre algorithme, mais sa taille va rester fixe tout au long des calculs ;
- ensuite, initialiser une nouvelle table contenant le nombre de cubes de chaque type à une itération donnée ; à la première étape, on a un seul cube, celui de départ ;
- finalement, à chaque itération, utiliser la table de transition pour recalculer le nombre de cubes de chaque type.

Évidemment, dans cette représentation, chaque spore va être comptée plusieurs fois. En effet, une spore située au milieu de notre structure sera comptée 8 fois (elle est un sommet de 8 cubes différents). Si elle est située sur un bord, elle ne sera comptée que 4 fois, 2 fois sur une arête, et finalement une seule fois sur un sommet de la structure (voir la figure 5). Ainsi, la position d'une spore est importante après tout.

3. https://doi.org/10.48556/SIF.1024.25.203.

4. OEIS Foundation Inc. (2025), Entry A343318 in The On-Line Encyclopedia of Integer Sequences, https://oeis.org/A343318 : *The number of vertices when starting with a cube (n=0) and iterating by dividing every cube into 8 equal cubes.*

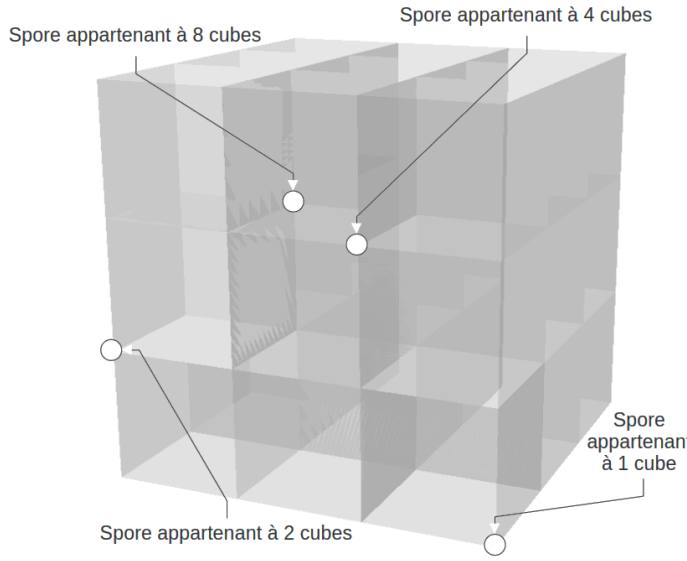


Fig. 5. Nombre de cubes auxquels appartient une spore.

Bien entendu, il est possible de résoudre cette épreuve en maintenant non seulement la table de transition des cubes, mais également des carrés en surface, des segments sur les arêtes, ainsi que des sommets. Mais cela complique le code et multiplie le risque de bugs dans le décompte de tous ces éléments.

Une autre manière de résoudre ce problème est d'ajouter un nouveau type de spores : ajoutées tout autour du cube de départ, elles ont pour propriété essentielle que toute spore en contact avec elles sera dans l'impossibilité de se multiplier. Ainsi, les spores situées sur les bords et les sommets de notre structure sont maintenant bien comptabilisées 8 fois, comme les autres, mais elles ne se multiplieront pas. Cela simplifie grandement le calcul final, au détriment de notre table de transition, car, avec maintenant 5 types de spores au lieu de 4, la table prend de l'embonpoint et comporte 5^8, soit 390 625 entrées.

Et effectivement, une implémentation telle que celle-ci aura la propriété de consommer la plupart des ressources (mémoire et processeur) sur la construction de la table de transition. Le calcul des différentes itérations est comparativement assez peu coûteux. Cela dit, le temps de traitement reste très raisonnable : à titre d'exemple, un code Python implémentant cet algorithme donne une solution en environ 2 secondes sur un ordinateur portable

moderne, 75 % du temps de calcul étant dédiés à la construction de la table de transition initiale.

Cette solution offre tout de même des possibilités d'optimisation. Par exemple, notre table de transition contient tous les cubes possibles, mais sa taille pourrait être réduite afin de tenir compte des symétries et rotations d'un même cube. Toutefois, si cela améliore l'empreinte mémoire, c'est probablement une optimisation dont on peut (et voudra !) se passer dans le contexte d'une compétition en temps limité.

c0d1ngUP team :
Emmanuel Laizé, université de Poitiers (UP),
Freddy Légé (UP), Laurent Signac (UP),
Benoit Tremblais (UP), Loïc Restoux, Caroline Tartary.