Le gars zombie colore

1 le background

Devant le succès inattendu de son deuxième film « On zombie road again », le dieu du calembour foireux décide de réaliser un troisième opus pour atteindre une clientèle fortunée, qui serait à même de lui financer un quatrième film. Le pitch : un golden-boy golfeur particulièrement doué vient d'envoyer sa balle sur le green d'un seul coup. Il remonte le fairway pour l'atteindre. Malheureusement pour lui, les produits bio utilisés par le jardinier ont fait muter au hasard certains insectes présents dans le gazon, en multipliant leur taille par $2^{\frac{97}{5}}$, et en les rendant avides de cervelle humaine. Deuxième problème, ils flairent le golfeur et se mettent à remonter le fairway en sens inverse. Pourquoi ? Parce que sinon, ils sortent du golf et comme ils s'y sentent bien, ils préfèrent y rester. Le golden-boy pourra-t-il survivre à son voyage vers le green en se cachant dans les bunkers le long du parcours ? Suspens insoutenable.

2 l'énoncé

Pour tester différent scénarios, le dieu fait appel à vous avec une rémunération proportionnelle aux recettes du film. Les différentes situations sont décrites par une chaîne de caractères de longueur N (variable selon les situations, mais $N \leq 10000$) composée de points, de caractères C pour les cachettes, d'un unique caractère H représentant l'humain, et de 0 à $N-1$ caractères Z représentant les insectes zombies. Les ., C, H et Z peuvent se trouver n'importe où dans cette chaîne mais **jamais** au même endroit. Par exemple : `..H.CZC..Z..` ou encore `Z.ZH.Z.C`. Pour simuler le déplacement des intervenants, on utilise un pas de temps plutôt qu'un déplacement en continu. A chaque pas de temps, on suppose que les Z se déplacent de 1 caractère vers la gauche, H de 1 caractère vers la droite ou 0 s'il se cache. L'objectif de H est d'atteindre la fin de chaîne qui représente l'arrivée, **le plus vite possible**. La simulation doit déterminer si H va :

- croiser un zombie sans être caché, et donc mourir,
- croiser des zombies en étant chaque fois caché, **ET** en sortant des cachettes le plus tôt possible,
- atteindre la fin de chaîne sans jamais avoir besoin de se cacher.

Important : les déplacements étant concurrents, on considère que l'humain peut se cacher s'il atteint une cachette exactement en même temps que le zombie. Il peut également atteindre la cachette avant et s'y cacher pendant plusieurs tours, c'est-à-dire le temps que le zombie atteigne lui-même la cachette. En revanche, si le zombie est juste

après l'humain et que ce dernier bouge, alors il meurt forcément. Ces situations sont illustrées par les exemples suivants.

— `H.C.Z` : après 2 déplacements, H et Z arrivent au même moment sur C. H peut donc se cacher pendant tout un tour et survivre.
— `HC.Z` : après 1 déplacement, H arrive sur C et Z juste après. Le tour suivant, si H sort de la cachette, il rencontre Z et meurt. H doit donc se cacher pendant deux tours pour survivre.
— `H.CZ` : après 1 déplacement, Z arrive sur C, et H juste avant. Lors du déplacement suivant, H pourrait arriver sur C. Mais comme Z avance aussi, H n'a pas le temps de se cacher et meurt.

Pour tester les situations, votre programme doit lire sur l'entrée standard :

1. un entier M représentant le nombre de situations à tester
2. M chaînes de caractères au format décrit ci-dessus (avec ., C, H, Z), représentant chacune une situation.

Ensuite, votre programme doit déterminer pour chaque situation, le sort de l'humain et écrire sur la sortie standard :

— `dead` si l'humain meurt,
— `X` s'il peut atteindre l'arrivée. `X` est le nombre **minimal** de fois où il doit se cacher afin de survivre et atteindre l'arrivée, **en minimisant le temps passé dans chaque cachette**.

Cela implique qu'il faut déterminer `X` en considérant que si l'humain est caché dans une cachette, il doit en sortir dès qu'il lui est possible d'atteindre une autre cachette sans être tué.

Le tableau 1 donne un exemple d'entrée et la sortie associée.

entrée	sortie
`4`	`0`
`...C..Z..Z..H..C..`	`dead`
`H........C.Z.Z.`	`1`
`....H.C.ZCC..Z..`	`2`
`....HC.ZCC.....Z.....`	

Table 1 – Exemple d'entrée et de sortie associée

Commentaires :

— situation 1 : aucun zombie n'est devant H. Il atteint le green sans se cacher.
— situation 2 : H n'a pas le temps d'arriver à la cachette. Il meurt.
— situation 3 : H arrive à la cachette 1 en même temps que le premier Z. Il peut donc se cacher. En revanche, il ne peut pas en sortir tout de suite, car il lui faudrait 3 mouvements pour atteindre la cachette 2. Or, à ce moment là, le deuxième Z n'aurait besoin que de 2 mouvements pour atteindre la cachette 2. H est donc obliger de rester dans la cachette 1 en attendant que le deuxième zombie passe. Il ne se cache au total qu'une seule fois.

— situation 4 : H doit d'abord se cacher dans la cachette 1 et attendre que le premier Z arrive sur la cachette. H peut alors en sortir car il a le temps d'atteindre la cachette 3, et ce en même temps que le deuxième Z. Il se cache au total deux fois.

3 les ressources

Pour vous aider dans la réalisation du programme, vous trouverez sur
`http://cours-info.iut-bm.univ-fcomte.fr`
un article dans la section `hackathon` portant le même titre que l'exercice. Il contient un lien permettant de télécharger un canevas de code, ainsi que le fichier d'entrée donné ci-dessus.

Bien entendu, vous êtes libres d'utiliser ou non ce canevas, mais c'est un gain de temps que de s'en servir comme base.