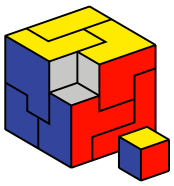
Prologin 2019



Concours national d'informatique

Épreuve écrite d'algorithmique
Paris

3 février 2018

Hache-Age

1 Préambule

Bienvenue à **Prologin**. Ce sujet est l'épreuve écrite d'algorithmique et constitue la première des trois parties de votre épreuve régionale. Sa durée est de 3 heures. Par la suite, vous passerez un entretien (20 minutes) et une épreuve de programmation sur machine (4 heures).

Conseils

— Lisez bien tout le sujet avant de commencer.
— **Soignez la présentation** de votre copie.
— N'hésitez pas à poser des questions.
— Si vous avez fini en avance, relisez bien, ou préparez votre présentation pour l'entretien.
— N'oubliez pas de passer une bonne journée.

Remarques

— Le barème est donné à titre indicatif uniquement.
— Indiquez lisiblement vos nom et prénom, la ville où vous passez l'épreuve et la date en haut de votre copie.
— Tous les langages sont autorisés, veuillez néanmoins préciser celui que vous utilisez.
— Ce sont des humains qui lisent vos copies : laissez une marge, aérez votre code, ajoutez des commentaires (**seulement** lorsqu'ils sont nécessaires) et évitez au maximum les fautes d'orthographe, sinon ça va barder.
— Le barème récompense les algorithmes les plus efficaces : écrivez des fonctions qui trouvent la solution le plus rapidement possible.
— Si vous trouvez le sujet trop simple, relisez-le, réfléchissez bien, puis dites-le-nous, nous pouvons ajouter des questions plus difficiles.

2 Hache-Age

Cette année, Joseph Marchand, un nain de la mine de la Moria, organise le concours annuel de la meilleure hache. Voulant faire de son mieux, il demande à l'organisateur de l'année précédente quel était le plus gros problème qu'il avait rencontré. Ce dernier lui rapporte que le concours avait pris du retard car les nains ne savaient pas où leurs haches étaient rangées et les candidats ont perdu trop de temps à les chercher au moment de la compétition.
Pour remédier à ce problème, Joseph décide de prendre du temps pour organiser les casiers de manière à retrouver la hache d'un participant le plus rapidement possible.
Sa première intuition est de tout simplement de mettre une étiquette avec le nom du nain propriétaire de la hache sur chacune d'entre elles.
Afin que le comité de décision du peuple de la Moria approuve son idée, Joseph doit expliquer les changements qu'il veut apporter au concours sur papier [1].

Tout au long du sujet, nous supposons que chaque participant peut présenter une unique hache.

3 Une hache dans une une botte de foin

Joseph Marchand veut d'abord représenter cette situation algorithmiquement par une liste. Chaque élément de la liste contient un couple contenant le nom du propriétaire (une chaîne de caractères) et le numéro du casier (un entier) contenant la hache du propriétaire. Il veut maintenant trouver une méthode efficace pour savoir où chaque nain a rangé sa hache.

Question 1 (1 point)

> Écrire une fonction permettant de chercher le numéro de casier correspondant à la hache d'un nain à partir de son nom.

La complexité d'un algorithme correspond au nombre d'actions élémentaires nécessaires pour exécuter un algorithme en fonction de la taille de l'entrée.

Question 2 (2 points)

> Donner la complexité (en fonction de A, la taille de la liste) de votre algorithme pour la Question 1, c'est-à-dire le nombre d'opérations effectuées par l'algorithme en fonction de la taille de l'entrée.

Joseph souhaite présenter la meilleure méthode possible au comité afin de marquer l'histoire du concours. En se remémorant l'année précédente il se rend compte que beaucoup de candidats s'inscrivent sans finalement se présenter au concours. Lorsque l'on recherche les haches de ces participants, l'équipe des organisateurs parcourt toutes les haches avant de se rendre compte à la fin que celle qu'ils cherchaient n'était pas présente. Après longue réflexion, Joseph pense qu'il existe une manière plus efficace de rechercher la hache d'un nain si la liste des haches est triée.

Question 3 (1 point)

> Écrire (par vous même, sans utiliser une fonction de tri de votre langage) une fonction permettant de trier la liste en fonction des noms de propriétaires. On trie les noms par ordre lexicographique, c'est a dire comparaison de la première, lettre puis la deuxième, etc ... comme dans le dictionnaire. Vous pourrez supposer que cet ordre est déjà implémenté et simplement comparer les noms entre eux en utilisant les symboles de comparaison habituels comme "$<$" ou "$>$".

Question 4 (1 point)

> Donnez la complexité de la fonction écrite à la question précédente.

Maintenant que la liste est triée, Joseph veut améliorer les performances de sa recherche.

1. Il faut savoir que les nains sont très procéduriers

Question 5 (3 points)

> Réécrire l'algorithme de la question 2 afin de tirer avantage des modifications de notre liste.

Question 6 (2 points)

> Donner la complexité (en fonction de A, la taille de la liste) de votre algorithme pour la Question 5

4 Hachage de hache

Joseph continue ses recherches et souhaite marquer l'histoire en étant celui qui aura révolutionné l'organisation du concours. Les rumeurs disent que cette année, on attend un nombre record de nains pour le concours de la meilleure hache, il est donc plus que temps de trouver une solution efficace au problème de rangement des haches. Au fil de ses recherches il découvre quelque chose qui correspond exactement à ce qu'il cherche : les tables de hachage.
Une table de hachage est une structure de données qui permet d'associer une clef à une donnée. Une table de hachage est en fait un tableau ordonné mais qui est ordonné d'une manière peu intuitive mais très efficace pour certaines opérations. Dans notre cas, une table de hachage permet d'associer au nom d'un nain le numéro de sa hache.
Dans cette partie nous utiliserons la fonction de hachage suivante : le condensat. La fonction de hachage associe au nom d'un nain la somme des indexes de chaque caractère de son nom dans l'alphabet (les majuscules sont traduites en minuscule, l'index de 'a' est 1 et l'index de 'z' est 26). Par exemple la valeur de cette fonction pour le prénom "Bor" est $35 = 2 + 15 + 18$.
On va maintenant ranger les couples (nom, numéro de casier) dans un tableau de taille 100. Le couple associé au nom d'un nain est stocké dans ce tableau à l'indice donné par sa fonction de hachage modulo [2] la taille du tableau (donc modulo 100).

Question 7 (1 point)

> Écrire une fonction qui permet de trouver le numéro d'une hache dans une table de hachage à partir du nom d'un nain.

Joseph veut maintenant utiliser sa table de hachage avec le nain **guthorrme** et avec le nain **formatique** qui ont tous les deux le même condensat, ce qui pose problème.

Question 8 (1 point)

> Imaginer une structure de données qui nous permettrait de les sauvegarder correctement tous les deux.

2. un modulo, représenté par l'opérateur '%', est le reste de la division euclidienne, par exemple : $27\%12 = 3$ car $27 = 12 \times 2 + 3$

5 Combien de haches à hacher ?

Joseph Marchand rencontre un problème dans son plan pour devenir le génie révolutionnant le concours, il ne peut pas garantir que sa solution sera efficace. En effet, si trop de participants se présentent, ils vont être très nombreux à avoir le même index et on retombera alors dans les problèmes de la première section... Dans la section précédente, Joseph avait décidé de calculer tous les index modulo 100, il ne pouvait donc y avoir que 100 valeurs possible pour l'index de 0 à 99. Dans la pratique quand le nombre de personnes dépasse environs $\frac{2}{3}$ du nombre d'index possible, le tableau devient un peu trop plein et inefficace. Quand on dépasse cette limite, il faut doubler l'espace que l'on autorise : si on avait une limite de 100 indexes possibles, on passe cette limite à 200. Il faut alors recalculer toutes les indexes, non plus modulo 100 mais modulo 200.

Question 9 (2 points)

> Écrire une fonction *double_size*(*axes*) qui à partir de la liste *axes* au format que vous avez décrit à la question 8, renvoie une liste de taille doublée comme expliqué ci-dessus.
> Vous pouvez modifier le format de votre liste si vous le précisez (notamment pour sauvegarder les noms des nains, dont vous aurez besoin, en plus de sauvegarder leur hache),

Question 10 (2 points)

> Estimer la complexité pour doubler la taille d'une table de hachage en fonction de N, sa taille d'origine.

Question 11 (3 points)

> En imaginant qu'on ajoute les nains à la table au fur et à mesure qu'ils arrivent et qu'on ne double la taille de la liste que quand on dépasse les 2/3 de remplissage de celle-ci (ce qui est assez rare), estimer la complexité moyenne de l'ajout d'un nain. C'est à dire lorsque n est grand, le coût pour ajouter n nains, divisé par n.

Seulement Joseph a un autre problème avec cette méthode, si le nombre de participants augmente beaucoup, la taille de leurs noms de famille ne change pas pour autant. Quelqu'un ayant un nom de 10 lettres pourra avoir un index d'au plus $10 \times 26 = 260$ s'il s'appel "zzzzzzzzzz". Dans ce cas, calculer modulo 1000 n'est pas plus utile que de calculer modulo 500 pour éviter que tous les nains participants n'aient tous le même index, on ne peut plus continuer d'agrandir notre tableau pour mieux y répartir les participants. Il faut alors changer de fonction de hachage, une nouvelle méthode pour calculer un nombre à partir d'un nom, plus grand que ceux obtenus avec la méthode précédente.

Question 12 (2 points)

> Proposer une nouvelle fonction de hachage qui résoudrait ce problème, expliquer en quoi elle semble pertinente ou au contraire moins bien.

Question 13 (1 point)

> Implémenter le calcul de l'index d'un nain en utilisant votre méthode.

6 Un triumvirat de nains

Maintenant que chaque nain sait où sa hache est rangée, Joseph peut commencer à réfléchir à la grande nouveauté de cette édition du concours de la meilleure hache, **le défilé de hache par équipes** [3]. Mais pour garder du suspense jusqu'à la fin du défilé, M. Marchand veut que les toutes les équipes aient le même niveau. Pour cela une équipe de nains corruptibles [5] a évalué minutieusement la hache de chaque nain afin de lui attribuer une note (un entier relatif). Après avoir jeté un dé, il décide que les équipes seront composées de 3 nains.
Pour composer les équipes, notre nain préféré [6] a rassemblé les notes de chaque nains au sein d'une seul liste. Il veut maintenant lister toutes les équipes possibles de 3 nains.

Question 14 (1 point)

> Écrire une fonction qui, donnée la liste des notes de chaque nains, affiche tous les triplets de notes possibles (pour chaque équipe de 3 nains possible, les notes de ces trois nains).

Question 15 (1 point)

> Écrire une fonction qui pour un entier k et une liste de notes, renvoie un équipe de 3 nains dont la somme des notes est égale à k (une telle équipe existera toujours [7]).

Question 16 (1 point)

> Estimer la complexité de la fonction écrite pour répondre à la question 15.

7 Bonus

Question bonus 17 (1 point)

> Combien faut-il de nains pour creuser en 2 jours un tunnel de 28 mètres dans du granit [8] ?

Question bonus 18 (1 point)

> Que signifie 'Gimli' le nom du célèbre nain en Khuzdul (la langue des nains) ?

Question bonus 19 (0 point)

> Dessiner un ou plusieurs nains [9].

Le sujet comporte 5 pages (sans compter la page de garde), 19 questions, et 2 questions bonus. Les questions normales sont notées sur 25 points, et les questions bonus rapportent au total 2 points, plus 1 point de présentation.

3. Rien à voir avec Flaubert [4] rassurez-vous
5. En effet un nain pourra assez facilement oublier une petite imperfection sur la hache d'un candidat contre une gemme de plus
6. Joseph bien sûr
7. les nains corruptibles font quand même bien leur travail
8. Des nains standards avec un équipement standard dans du granit standard.
9. Ça ne rapporte pas de points mais tous le monde aime les dessins !