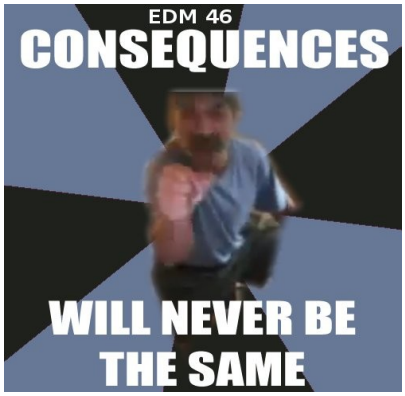
TD d'algorithmique avancée n°4

1 CHERCHONS...



Rebecca Black, interprète d'une formidable chanson, va bientôt subir une attaque de la part de 4chan... Sa réputation virtuelle va prendre un mauvais coup, et sa maison de disque risque d'apprendre ce qu'est une attaque DDOS.

Heureusement, vous faîtes partie de la police d'Internet (créée par Mr Slaughter) et vous allez empêcher cela. Vous possédez un tableau **tab** représentant tout Internet (ah oui, quand même), et où sont représentés des entiers : chaque entier est une adresse IP d'un ordinateur sur Internet.

Ce tableau est trié, et vous avez la charge de devoir retrouver un ordinateur repéré pour piratage le plus rapidement possible afin de le désactiver.

1. Ecrire une fonction qui recherche, dans **tab**, une adresse IP (un entier) donné en paramètre. Combien de cases, dans le pire des cas, devez-vous parcourir ? Etait-ce assez rapide ?

2. Et si **tab** n'était pas trié, cela changerait-il quelque chose au nombre de cases à parcourir dans le pire des cas ?

3. Reprenons la question 1 : pouvez-vous faire mieux ? En quoi votre méthode est meilleure ?

4. Ecrivez une fonction qui recherche dans **tab** le plus grand élément.

2 ET TRIONS...

On imagine que l'on est responsable maintenant de trier le tableau de toutes les adresses IP d'Internet. Bon courage...

1. Ecrivez une fonction qui parcourt le tableau et, dès que deux éléments côte à côte ne sont pas dans le bon ordre, les inverse et repart au début du tableau. Combien de cases, dans le pire des cas, devez-vous parcourir avant que la tableau ne soit trié ?

2. Pouvez-vous améliorer cet algorithme ? Avez-vous vraiment amélioré l'algorithme ?

3. Voyez-vous un autre moyen de faire, en recherchant dans le tableau le plus grand élément ? Votre méthode est-elle meilleure ?

4. Voyez-vous un moyen plus rapide ?

3 4CHAN

4chan est un forum assez populaire sur Internet. Sa particularité est de garantir l'anonymat complet de ses contributeurs : aucune adresse IP n'est conservée en mémoire, et les sujets les moins populaires sont remplacés par les plus populaires (pas de stockage à long terme des messages).

Nous allons tenter de reproduire le comportement de ce forum avec des listes. Dans un premier temps, nous définirons une liste **SujetDeDiscussion**, qui permettra de stocker tous les messages relatifs à un sujet de conversation précis. Puis, nous devrons trouver un moyen de stocker au maximum 150 **SujetDeDiscussion**, et prévoir les fonctions permettant d'ajouter un **SujetDeDiscussion** tout en en supprimant un.

1. Définissez la structure **ListeMessage** qui contient un champ *Auteur* (de type String), un champ *Contenu* (de type String), et ?... Puis, définissez la structure **SujetDeDiscussion**, qui contient un champ *DateDeDerniereContribution* (de type entier), un champ *Sujet* (de type String), et un champ *Messages* (de type **ListeMessage***)* qui pointera vers le premier message du sujet de discussion.

2. Faîtes une fonction **AjouterMessage**, qui prend en paramètre un **SujetDeDiscussion**, un nom d'auteur et un message, et qui ajoute un message au sujet de discussion (vous devrez utiliser la fonction fictive Instant(), qui renvoie un entier avec la date et l'heure du jour (à la milliseconde près).

3. Vous devez trouver maintenant un moyen de stocker au plus 150 messages de discussion dans une structure. Avant de vous lancer, réfléchissez au fait que vous devrez faire une fonction **AjouterSujetDeDiscussion**, qui prendra en paramètre un sujet, un auteur et un message, qui recherchera dans la structure choisie le sujet de discussion avec la date de contribution la plus ancienne, et le remplacera par le nouveau sujet de discussion.

4. Imaginons que chaque **SujetDeDiscussion** possède un nouveau champ *NumeroIndentiteUnique*, qui l'identifie de façon unique par rapport aux autres **SujetDeDiscussion**. De quoi auriez vous besoin pour faire une fonction qui ajoute rapidement un message à un sujet de discussion qu'on ne possède pas (c'est à dire que l'on ne possède pas la structure **SujetDeDiscussion** à proprement dit, mais on peut posséder autre chose, comme par exemple le *NumeroIndentiteUnique* du sujet de discussion auquel on souhaite ajouter un message.

Attention : prenez en compte le fait que pleins de gens sont connectés en même temps sur votre site et y font des opérations.