INF301 : rapport d'examen session 1 2022-2023

Les questions 1 et 3, 4, 5 étaient simples et ont été réussies la plupart du temps.

À la question 2, nombreux·ses ont répondu "erreur mémoire", alors que la case 6 fait bien partie du tableau ! (c'est ce qui permet d'ailleur d'ajouter des éléments à la séquence...).
Cependant on ne maitrise ni ne connait a priori pas le contenu de ces cases. La meilleure réponse était donc "manque de données dans l'énoncé".

Question 6, c'est le pendant de la question 2. Par contre cette fois on connait la valeur située dans la case 5 du tableau : c'est 20. Effectivement, la suppression dans une séquence ne va ici modifier que S.longueur (aucun décalage à faire puisque 20 est le dernier élément), donc S.tab[5] sera encore à 20.

Question 7 : beaucoup de bonnes réponses, mais attention cependant à l'arbre qui contient un nœud ayant 3 fils alors que la définition stipulait bien que les tas sont des arbres _binaires_.

Question 8 : la fonction vérifie_nœud est incorrecte : elle ne peut jamais renvoyer "Vrai" !
La question est donc absurde puisqu'il est impossible d'appeler cette fonction pour que verifie_tas fonctionne. (Points partiels donnés à la réponse "Aucune réponse correcte")

Question 9 : la plus difficile de l'examen. Le plus simple était de se rendre compte que pour supprimer la valeur à la racine, on peut la remplacer par la plus grande valeur entre le fils gauche et le fils droit, et ensuite supprimer recursivement la valeur choisie dans le sous-arbre.

Une autre solution consistait à essayer de "fusionner" les deux sous-arbres en "descendant" mais il est plus facile de faire des erreurs algorithmiques, et le plus gros inconvénient est que cela va fortement déséquilibrer le tas.

Les erreurs les plus courantes étaient les suivantes :
- modifications locales qui ne sont donc plus visible après le retour de la fonction (par exemple pour "vider" l'arbre : "A ← Nil", alors que A est un argument donc une variable locale)

- incohérence entre valeurs de retour : exemple A.gauche ← f (A.gauche) mais f retourne un entier

- ordre de vérification de Nil : exemple comparer A.gauche.valeur et A.droit.valeur, et _ensuite_ vérifier si A.gauche et A.droit sont bien ≠ Nil ?!?

Question 10 : il était difficile de répondre correctement à cette question à partir d'un algorithme complétement faux.
Pour un algorithme correct, on observe que les appels récursifs, quand il y en a, ne se font à chaque fois que dans une branche de l'arbre. Ainsi, dans le pire cas, on sélectionne à chaque fois la branche qui mène à la plus grande profondeur. Le pire cas est ainsi la hauteur totale du tas notée h. Comme chaque appel est en O(1), la complexité totale est O(h).
Si le tas est bien équilibré, alors cette hauteur est en O(log n) avec n le nombre de valeurs du tas.

Question 11 : était assez simple si on utilisait bien la question 10 ne se mélangeait pas les pinceaux.
Le principal problème était cependant de bien récupérer un tas modifié à chaque appel à 'extrait_max' (sans quoi le tas est modifié localement, et donc jamais vidé dans la boucle principal).
Il fallait être attentif également à l'ordre des éléments : bien ajouter _en tete_ pour que chaque maximum (qui est en fait plus petit que ceux qu'on a extraits avant...) se mette avant les autres.
Il était inefficace de passer par un tableau intermédiaire avant de recréer une LC. Il ne fallait pas refaire d'algorithme de tri.

Recommandations pour la question 9 :
Bien spécifier les valeurs de retour. Ici, on peut utiliser un couple (entier, arbre) qui permettra de renvoyer à la fois la plus grande valeur du Tas, et de renvoyer le tas mis à jour (sans cette valeur). On peut utiliser une fonction auxiliaire pour simplifier l'algorithme et réduire le risque d'erreurs. (Note : l'algorithme présenté ici ne libère pas la mémoire pour ne pas complexifier un code déjà difficile à comprendre pour beaucoup.)

extrait_max doit s'utiliser ainsi : (val_max, T) ← extrait_max(T)

```
extrait_max (T):
    si T = Nil alors erreur "arbre vide"
    max = T.valeur
    T ← supprime_max (T)
    retourner (max, T)


supprime_max (T):
    // on est sûr que T ≠ Nil
    si T.gauche = Nil et T.droit = Nil
        retourner Nil
    si T.gauche = Nil :
        retourner T.droit
    si T.droit = Nil :
        retourner T.gauche
    si T.gauche.valeur > T.droit.valeur
        T.valeur ← T.gauche.valeur
        T.gauche ← supprime_max (T.gauche)
    sinon
        T.valeur ← T.droit.valeur
        T.droit ← supprime_max (T.droit)
    retourner T
```

Recommandations pour la question 11:
Fonctionner encore une fois en retournant le tas modifié comme valeur de retour.

```
Tri_par_tas (T)
    S ← nouvelle séquence
    tant que T non vide:
        (x, T) ← extrait_max(T)
        ajoute_tete (x, S)
    retourner S
```

Exécution d'algorithmes :

Question 13 : une analyse simple montre qu'à chaque itération, on s'écarte un peu plus vers la gauche, sans jamais tomber sur un gris. On sort donc de la grille.

Question 14 : pas de difficulté majeure. Même si on oubliait d'exécuter les "gauche" et "droite" en retour d'appel de fonction 'f', on trouvait la bonne solution.

Question 15 : x est une variable locale et chaque appel de fonction a sa propre "copie".
Donc au retour des appels la valeur locale n'a pas été changée par les appels.
On arrive ainsi en 3 appels récursifs g(1), encore g(1), puis g(2) sur A5, puis au retour des appels on refait 2 fois gauche (dans le deuxième appel où x a été modifié), puis 1 fois à gauche dans l'appel initial (x non modifié) pour arriver en A2.

Question 16 : ici la difficulté et de ne pas oublier que l'exécution continue au retour des appels récursifs, en particulier que la condition "si <hachures>" est effectuée quel que soit le résutat du "si <blanc>" (car ce n'est pas un "sinon").
On a donc trois appels récursifs à h() qui s'empilent et arrivent en A5, puis au retour dans la pile des appels les trois conditions "si <hachures>" sont vraies (en A5, B6 puis C7) et font arriver en D8.

Note : certain·es étudiant·es ont permuté les réponses aux questions 14, 15, 16, en répondant à la place dans le mauvais ordre 15, 16 puis 14. J'ai pu détecter une partie de ces erreurs et corriger manuellement mais pas toutes (en particulier quand il y avait une mauvaise réponse parmis ces questions).
Si c'est votre cas merci de me contacter pour voir s'il est possible de corriger votre note.