Licence Info
Algorithmique Avancée
TD 10 : décembre 2006
Les algorithmes Fusion-Recherche

1. Quel est le temps de calcul (à $O(\ )$ près dans le pire cas) d'une suite d'opérations `fusion` suivie d'une suite d'opérations `recherche` pour les trois algorithmes

   (a) liste avec mise à jour du plus petit ensemble

   (b) arbre avec le plus grand ensemble comme racine (fusion équilibrée) mais sans mise à jour des pointeurs

   (c) (l'algorithme de Tarjan) arbre avec le plus grand ensemble comme racine et avec mise à jour des pointeurs (compression des chemins)?

2. Pour l'algorithme de Tarjan, quel est le nombre de mises à jour de pointeurs pour la suite d'opérations (si les deux ensembles sont de la même taille, on prendra le deuxième comme racine):

```
fusion(1,2)
fusion(3,4)
fusion(1,3)
chercher(1)
fusion(7,8)
fusion(5,6)
fusion(6,8)
chercher(5)
fusion(1,5)
chercher(1)
fusion(9,10)
fusion(11,12)
fusion(9,11)
chercher(9)
fusion(15,16)
fusion(13,14)
fusion(14,16)
chercher(13)
fusion(10,14)
chercher(9)
fusion(1,13)
chercher(2)
```

3. Dans l'arbre donné, les chiffres indiquent la taille de chaque sous-arbre;

```
          12
         /  \
        /    \
       /      \
      5        6
     / \      / \
    /   2    /   \
   2   /    3     2
  /   1    / \     \
 /        /   \     \
1        1     1     1*
```

   (a) Expliquez pourquoi cet arbre ne peut pas avoir produit par une suite de fusions avec compression des chemins lors de chaque fusion.
   (b) Dessinez l'arbre produit par la compression des chemins si on fait une recherche sur l'elément marqué *,
   (c) Notez que dans cet arbre, pour le sous-arbre de taille 3 il n'est plus vrai que la taille du père est au moins 2 fois la taille d'un fils,
   (d) Expliquez comment l'analyse du temps de calcul peut prendre en compte ce type de possibilité et préserver la conclusion que le temps amorti des opérations est $O(log^*n)$.