Lycée Jean Moulin **Draguignan**
NSI 1re Année .....

Analyse des algorithmes

Recherche du maximum dans un tableau

Voici un algorithme qui prend en entrée un tableau de n valeurs et qui retourne le maximum

Données : T un tableau de n valeurs

Résultat : max le maximum du tableau
max← T[0]
n← taille du tableau
Pour i allant de 1 à n-1 **faire**
  Si T[i]>max **alors**
    max←T[i]
retourner S

? Question 1:

Que pouvez vous dire sur la terminaison de cet algorithme ?

Aucun problème de terminaison car cet algorithme ne comporte aucune boucle while (une boucle for se termine toujours).

? Question 2:

Examiner la correction de cet algorithme en considérant l'invariant de boucle : max est le maximum des valeurs de la partie du tableau traité

Avant la boucle on a traité uniquement la 1ère valeur du tableau, et max=T[0] est donc bien le maximum puisque c'est la seule valeur.
Supposons que la propriété soit vraie avant une itération, alors après cette itération on a traité une valeur de plus du tableau. Si cette valeur n'est pas plus grande que max alors max ne change pas et c'est toujours le maximum du tableau traité; sinon max devient cette nouvelle valeur qui est bien le nouveau maximum du tableau traité. La propriété reste donc vraie, c'est un invariant de boucle et l'algorithme est correct.

? Question 3:

Montrer que la complexité de cet algorithme est linéaire ($\Theta(n)$)

La boucle for se répète n-1 fois (de i = 1 à i=n-1) où n est la taille du tableau fourni en entrée.
Si le tableau est k fois plus grand, le nombre d'itérations sera donc aussi multiplié par k (ou presque).
La complexité de cet algorithme est donc bien linéaire.

Recherche dans un tableau

Voici un algorithme de recherche de valeur dans un tableau

Données : T un tableau de n valeurs
x un élément (pas forcément dans T)

Résultat : -1 si $x \notin$ T, l'indice de la première occurrence de x dans T

```
i ← 0
n ← taille(T)
Tant que  i < n et T[i]≠ x faire
  i=i+1

Si i=n alors
  i←-1
retourner i
```

? Question 4:

Que pouvez vous dire sur la terminaison de cet algorithme ?

Dans le pire des cas, c'est-à-dire si x n'est pas dans T, le variant de boucle i prend les valeurs i=(0,1,2, ... , n) car il part de 0 et est incrémenté de 1 à chaque itération.
La condition de sortie de boucle (i>=n) sera donc forcément remplie en un nombre fini d'itérations et l'algorithme se termine.

? Question 5:

Examiner la correction de cet algorithme en considérant l'invariant de boucle : x n'apparaît pas dans la partie du tableau traité

Avant la première itération aucune partie du tableau n'est traité et la propriété est donc vraie.
Si la propriété est vraie avant une itération, elle le sera encore à l'itération suivante car sinon on sort de la boucle.
La propriété est donc bien un invariant de boucle qui reste vraie à la fin de la boucle.
Si i=n, tout le tableau a été traité sans trouver x et on renvoie -1.
Sinon, on est sorti au moment ou T[i]=x et on renvoie i qui est bien l'indice de la première occurence de x dans T.

? Question 6:

Montrer que la complexité de cet algorithme est linéaire ($\Theta(n)$)

Dans le pire des cas, la boucle tant que parcourt le tableau de taille n en entier en n itérations (n= 0 à n-1 par pas de 1).
Donc si le tableau fourni en entrée est k fois plus grand, il y aura k fois plus d'itérations.
La complexité de cet algorithme est donc bien linéaire.