TD n°1

Arbre Binaire de Recherche

1 Arbres binaires de recherche

Exercice 1 Combien y a-t-il d'abres binaires de recherche dont les éléments sont $\{3, 5, 8, 12\}$?

Exercice 2 On considère tous les nombres compris entre 1 et 1000. Donnez deux ordres d'insertion de ces nombres dans un ABR :
– Un qui va donner un arbre completement déséquilibré, c'est-à-dire de hauteur maximale possible ;
– Un qui va donner un arbre équilibré, c'est-à-dire le moins haut possible.

Exercice 3
1. Calculer le nombre maximal de nœuds d'un arbre binaire de profondeur p.
2. En déduire qu'un arbre binaire à n nœuds a une profondeur d'au moins $\lceil \log_2(n+1) \rceil$.
3. Montrez par récurrence sur p que la profondeur **moyenne** d'un nœud de l'arbre binaire **complet** de profondeur p est comprise entre $p-1$ et p. (NB : la profondeur de la racine est 1).

Exercice 4 Un arbre binaire de profondeur p est *équilibré* si tout nœud ayant moins de deux descendants est à la profondeur p ou $p-1$.

Écrire un algorithme qui dit si un arbre binaire donné est équilibré ou non.

2 Analyse de complexité

Exercice 5 Que fait le programme `devinette` ? Quelle sont les complexités des deux programmes dans le pire, le meilleur, des cas ? En moyenne ?

```
devinette (T[]: tableau, n : entier) {
   pour i=n à 2 par pas de -1
      Pour j=1 à i-1
         Si T[j]>T[j+1]
            échanger T[j] et T[j+1]
         FinSi
      FinPour
   FinPour
}
```

```
boucle (n : entier) {
   pour i=1 à n
      Pour j=1 à i
         Pour k=1 à j
            instruction en O(1)
         FinPour
      FinPour
   FinPour
}
```

Exercice 6 Que calculent les deux fonctions suivantes ? Quelle est leur complexité dans le pire, le meilleur, des cas ? En moyenne ?

```
inconnue1 (a : entier) {
   p <- 0;
   Tant que (a modulo 2 = 0) Faire {
      a <- a div 2;
      p <- p + 1;
   }
   Si (a = 1) Alors
      retourner p;
   Sinon
      retourner -1;
}
```

```
inconnue2 (a, b : entiers) {
   p <- 1;
   r <- 0;
   Tant que (a > 0) Faire {
      r <- r + (a modulo 10) * p;
      a <- a div 10;
      p <- p * b;
   }
   retourner r;
}
```

Exercice 7 Soit n un entier strictement positif et soit T un tableau de n nombres distincts. On appelle *maximum provisoire* de T tout indice i dans $[1, n]$ tel que, pour tout j dans $[1, i-1]$, $T[i]$ est plus grand que $T[j]$.

On veut calculer le nombre moyen de maxima provisoires, sur tous les tableaux T qui sont des permutations de $[1, n]$ (*i.e.*, des bijections de $[1, n]$ dans $[1, n]$). C'est le nombre moyen d'affectations à faire dans l'algorithme classique de recherche du maximum.

On appelle $P_{n,k}$ le nombre de permutations de $[1, n]$ qui ont k maxima provisoires. Montrer que l'on a les relations :
- $P_{1,1} = 1$, et $P_{1,k} = 0$ pour tout $k \neq 1$;
- $P_{n,k} = P_{n-1,k-1} + (n-1)P_{n-1,k}$.

Soit S_n le nombre moyen de maxima provisoires d'une permutation de $[1, n]$. Montrer que :

$$S_n = H_n = \sum_{k=1}^{n} \frac{1}{k}.$$

Exprimer S_n en Θ d'une fonction en n.