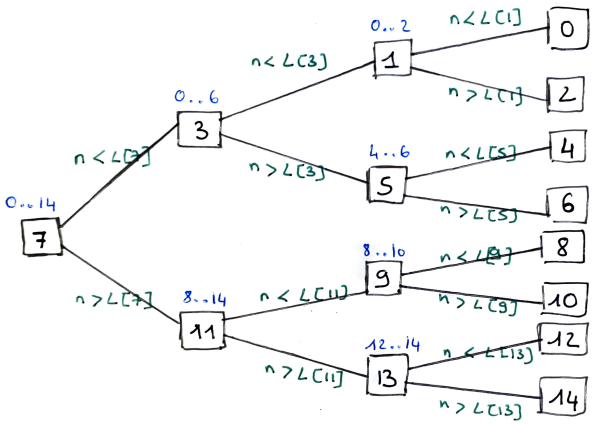
CORRIGÉ - TP4 : ALGORITHMES DICHOTOMIQUES

1 Recherche dichotomique

1. On a l'arbre de décision suivant, j'indique en vert la condition à vérifier pour que la branche soit celle choisie à chaque étape, en en bleu les indices de la partie de la liste concernée.



2.

```
def recherche_dichotomique(L,x):
    g = 0
    d = len(L) - 1
    while g <= d:
        m = (g+d)//2
        if x == L[m]:
            return True
        elif x < L[m]:
            d = m-1
        else:
            g = m+1
    return False
```

3.

4.

```
def indice_dicho(L,x):
    g = 0
    d = len(L) - 1
    while g <= d:
        m = (g+d)//2
        if x == L[m]:
            return m
        elif x < L[m]:
            d = m-1
        else:
            g = m+1
    return None
```

Pour cette fonction, on propose par exemple le jeu de tests suivant :

```
L = [2,4,5,7,11]

assert indice_dicho(L,4) == 1

assert indice_dicho(L,6) == None

assert indice_dicho(L,2) == 0

assert indice_dicho(L,11) == 4

assert indice_dicho(L,0) == None

assert indice_dicho(L,12) == None

assert indice_dicho([],12) == None
```

5. Dans le cas d'une liste de longueur $2^n - 1$, au maximum, n cases sont lues.

2 Exponentiation rapide

1.

```
def puissance(x,k):
    assert k >= 0
    r = 1
    for _ in range(k):
        r = r*x
    return r
```

2. On remarque que si $k < 0$, cela revient à evaluer $\frac{1}{x}$ à la puissance k, on a donc :

```
def puissance(x,k):
    if k<0:
        assert x != 0
        x = 1/x
        k = -k
    r = 1
    for _ in range(k):
        r = r*x
    return r
```

3. Pour calculer x^{16}, on peut calculer $x^2 = x \times x$ (1 multiplication), puis $x^4 = x^2 \times x^2$ (1 multiplication), puis $x^8 = x^4 \times x^4$ (1 mulitplication) et enfin $x^{16} = x^8 \times x^8$ (1 multiplication). On a donc en tout réalisé 4 multiplications.
4. On remarque que $21 = 2 \times 10 + 1, 10 = 2 \times 5, 5 = 2 \times 2 + 1$ et $2 = 1 + 1$. On calcule donc $x^2 = x \times x$ (1 multiplication), puis $x^5 = x^2 \times x^2 \times x$ (2 multiplications), puis $x^{10} = x^5 \times x^5$ (1 multiplication) et enfin $x^{21} = x^{10} \times x^{10} \times x$ (2 multiplications).On a donc en tout réalisé 6 multiplications.
5. (a) Si initialement $r = 1$, on doit initialiser y à x et n à k.

   (b) · Si n est pair :

$$\begin{array}{rcl} y & \leftarrow & y^2 \\ r & \leftarrow & r \\ n & \leftarrow & \left\lfloor \frac{n}{2} \right\rfloor \end{array}$$

   · Si n est impair :

$$\begin{array}{rcl} y & \leftarrow & y^2 \\ r & \leftarrow & yr \\ n & \leftarrow & \left\lfloor \frac{n}{2} \right\rfloor \end{array}$$

   (c) On déduit le programme suivant :

```
def puissance_rapide(x,k):
    if k<0:
        assert x != 0
        x = 1/x
        k = -k
    y = x
    n = k
    r = 1
    while n > 0:
        if n%2 == 1:
            r = r * y
        y = y*y
        n = n//2
    return r
```

   (d) On modifie la fonction précédente :

```
def nombre_mult(x,k):
    if k<0:
        assert x != 0
        x = 1/x
        k = -k
    y = x
    n = k
    r = 1
    c = 0
    while n > 0:
        if n%2 == 1:
            r = r * y
            c = c+1
        y = y*y
        c = c+1
        n = n//2
    return c
```

Pour trouver le plus petit k tel que le calcul de x^k par la fonction `puissance_rapide` nécessite au moins 40 multiplications, on utilise le petit code suivant :

```
i = 0
while nombre_mult(1,i)<40:
    i = i+1
print(i)
```

et Python nous renvoie :

```
>>> (executing file "TP4 - dichotomie.py")
1048575
```

C'est beaucoup !

3 Calcul de frontière par dichotomie

1.

```
def frontiere(L,x):
    g = 0
    d = len(L) - 1
    while g<d:
        m = (g+d)//2
        if x <= L[m]:
            d = m
        else:
            g = m+1
    if g>d or x <= L[g]:
        return g
    else:
        return g+1
```

2. On propose le jeu de tests suivant :

```
if frontiere([1,3,4,6],4) != 2:
    print("Lorsque x est un element de la liste le resultat \
    n'est pas correct'")

if frontiere([1,3,4,6], 2) != 2:
    print("Lorsque x n'est pas un element de la liste le resultat\
    n'est pas correct'")

if frontiere([1,3,4,6], 7) !=4 :
    print("Erreur lorsque x est plus grand que tous les elements\
     de la liste")
```

```
if frontiere([1,3,4,6], 0) !=0 :
    print("Erreur lorsque x est plus petit que tous les elements\
     de la liste")

if frontiere([1,3,4,6, 8],4) != 2:
    print("Lorsque x est un element de la liste le resultat\
     n'est pas correct, la liste n'a pas une longueur puissance de 2")
```

Fin