# Algorithmes dichotomiques

**Objectif :** L'objectif de ce tp est de rendre l'élève capable :

- d'écrire en langage **Python** des algorithmes dichotomiques
- de tracer l'évolution du nombre d'itérations nécessaire pour résoudre un problème en fonction de la taille du problème initial et le comparer au nombre d'itérations théorique

## 1 Devinettes

**Q - 1** : *Écrire une fonction* `devinette(n, i)` *qui prend en arguments deux entiers naturels* `n` *et* `i` *tels que* $n \geq i$ *et qui retourne le nombre d'itérations effectuées pour retrouver* `i` *à l'aide d'une méthode par dichotomie (cf Algo 1) dans l'intervalle* $[\![0;n]\!]$.

**Q - 2** : *Générer 1000 appels de la fonction* `devinette` *avec* `n` *tiré au hasard dans l'intervalle* $[\![10;[\![100$ *et* `i` *tiré au hasard dans l'intervalle* $[\![0;n]\!]$.

**Q - 3** : *Tracer pour chaque valeur de* `n` *tirée, le nombre d'itérations utilisées pour déterminer* `i`.

**Q - 4** : *Tracer sur le même graphe la fonction* $\ln_2$ *pour chaque entier compris entre 10 et 100.*

## 2 Recherche dichotomique dans un tableau trié

**Q - 5** : *Programmer la fonction* `recherche_dicho(T, x)` *présentée par l'algorithme Algo 1.*

**Algorithm 1** Recherche dichotomique dans un tableau trié

**entrée:** T un tableau de $n$ valeurs et $x$ un élément (pas forcément du tableau)
**résultat:** -1 si $x \notin$ T, l'indice d'un $x$ dans le tableau T sinon

```
1:  recherche_dicho(T, x)
2:  n← taille(T )
3:  min,max,mil← 0,n-1,(min+max)//2
4:  tant que min<max et T[mil]≠x faire
5:      si T[mil]<x alors
6:          min← mil+1
7:      sinon
8:          max ← mil-1
9:      fin si
10:     mil← (min+max)//2
11: fin tant que
12: si T[mil]≠x alors
13:     rep ← -1
14: sinon
15:     rep ← mil
16: fin si
17: renvoi: rep
```

**Q - 6** : *Générer 1000 tableaux d'entiers de taille comprise entre 10 et 100 éléments et de valeurs comprises entre 0 et 100, trier les tableaux (avec* `.sort()`*) et pour chaque tableau, tirer au sort un nombre compris entre 0 et 100.*

**Q - 7** : *Déterminer pour chaque tableau, le nombre d'itérations nécessaires pour déterminer l'indice de la valeur tirée au sort si elle appartient ou sa non-appartenance.*

**Q - 8** : *Tracer, pour chaque tableau, le nombre d'itérations effectuées en fonction de la taille du tableau.*

**Q - 9** : *Tracer sur le même graphe la fonction* $\ln_2$ *pour chaque entier compris entre 10 et 100.*

## 3 Exponentiation

**Q - 10** : *Écrire trois fonctions permettant de calculer* $x^n$ *:*

- *avec un algorithme naïf itératif*
- *avec une méthode itérative dichotomique*
- *avec un méthode récursive dichotomique*

**Algorithm 2** Exponentiation naive

```
    entrée: x un nombre réel, n un entier naturel
    résultat: x^n
    puiss_naive(x,n)
 1: p←1
 2: pour i entre 1 et n faire
 3:     p ←p.x
 4: fin pour
 5: renvoi: p
```

**Algorithm 3** Exponentiation rapide itérative

```
    entrée: x un nombre réel et n un entier naturel
    résultat: un nombre réel r = x^n
    puiss(x, n)
 1: si n==0 alors
 2:     renvoi: 1
 3: fin si
 4: r← 1
 5: tant que n > 0 faire
 6:     si n modulo 2==1 alors
 7:         r←r.x
 8:     fin si
 9:     x←x.x
10:     n←n//2
11: fin tant que
12: renvoi: r
```

**Algorithm 4** Exponentiation rapide récursive

```
    entrée: n un entier positif et x un nombre réel
    résultat: un nombre réel r = x^n
    puis_rec(x, n)
 1: si n==0 alors
 2:     renvoi: 1
 3: sinon
 4:     si n modulo 2==0 alors
 5:         renvoi: puis_rec(x.x,n/2)
 6:     sinon
 7:         renvoi: x.puis_rec(x.x,(n-1)/2)
 8:     fin si
 9: fin si
```