TD 15 : Algorithmes dichotomiques

La technique `diviser pour régner` est un paradigme de programmation basé sur une approche récursive : il consiste à diviser le problème en plusieurs sous-problèmes de plus petite taille, jusqu'à parvenir à un problème tellement simple que sa solution est immédiate.
Cette méthode n'est efficace que si la taille des sous-problèmes suit une loi géométrique (par exemple, on divise par 2 la taille lors de chaque appel). Le calcul récursif du terme u_n d'une suite à partir de u_{n-1} n'est pas du type `diviser pour régner`.

On s'intéresse dans ce TD à la recherche dichotomique et à l'exponentiation rapide.

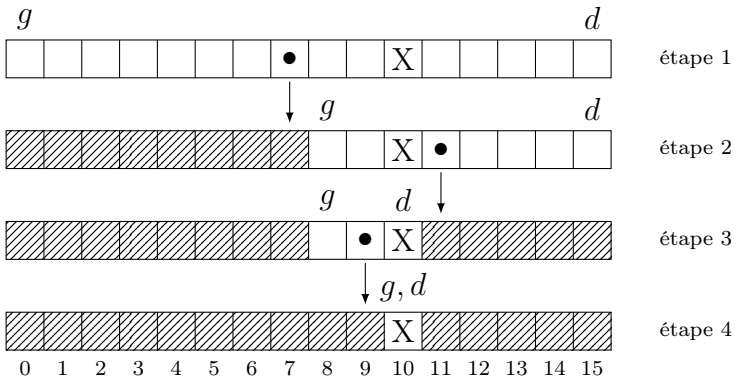
1 Recherche dichotomique

Le premier élément incontournable de la recherche dichotomique est qu'elle opère sur un tableau **trié** ! Cette contrainte est primordiale, c'est grâce à elle qu'on gagne en efficacité par rapport à un tri quadratique comme les tris insertion ou sélection.

Algorithme : `recherche_dichotomique (T, x)`
Entrées :
- `T` : tableau **trié** de `n` valeurs
- `x` : valeur recherchée dans le tableau `T`

Sortie :
- soit indice `i` tel que `T[i]=x`
- soit valeur spéciale « `Non Trouvé` »

Procédure :
1. Fixer `g=0` et `d=n-1`
2. Tant que `g` $\leq$ `d`, faire la chose suivante :
   A. Fixer `m` à `partie_entière( (g+d)/2 )`
   B. Si `T[m]=x`, alors retourner `m`
   C. Sinon, si `x < T[m]`, alors fixer `d=m-1`
   D. Sinon, fixer `g=m+1`
3. Retourner la valeur de `Non-Trouvé`



Exercice 1 (Exemple)

1. Lire l'algorithme proposé de recherche dichotomique.
2. Appliquer cet algorithme sur l'exemple proposé. Le tableau est représenté sous forme d'une série de case, le tableau est trié de manière croissante vers la droite. La valeur recherchée est représentée par un `X` dans le tableau. La zone de recherche dans le tableau est délimitée par un indice `g` (gauche) et un indice `d` (droite). Pour chaque étape, déterminer les valeurs de g, d et du milieu m.
3. Comment qualifier la suite des valeurs prises par g ? et par d ?
4. Pour quelle raison, à l'étape 2C dans l'algorithme, a-t-on le droit de fixer `d` à `m-1` ?
5. Pour quelle raison, à l'étape 2D dans l'algorithme, a-t-on le droit de fixer `g` à `m+1` ?

Exercice 2 (Recherche dichotomique : cas général)

1. Programmer en python l'algorithme proposé.
2. Proposer une version récursive de cet algorithme.
3. Pour quelle raison, à l'étape 2C dans l'algorithme, a-t-on le droit de fixer `d` à `m-1` ? Que peut-on en déduire pour la suite des valeurs prises par `d` ?
4. Pour quelle raison, à l'étape 2D dans l'algorithme, a-t-on le droit de fixer `g` à `m+1` ? Que peut-on en déduire pour la suite des valeurs prises par `g` ?
5. En déduire une preuve de terminaison de l'algorithme proposé.
6. Prouver la correction partielle de l'algorithme à l'aide de l'invariant de boucle suivant : « Au début de chaque itération de la boucle à l'étape 2, si `x` est présent dans le tableau `T`, alors il est présent dans le sous-tableau `T[g..d]` ».

2 Exponentiation rapide

On se propose de calculer x^n en n'utilisant que l'opération `*`. On s'interdit d'utiliser la fonction mathématique `pow` ou `x**n`.

Exercice 3 (Calcul naïf de x^n)

1. Ecrire la fonction `puissance(x,n)` qui calcule x^n de manière récursive.
2. Combien de multiplication faut-il pour calculer x^n ? Estimer l'ordre de grandeur asymptotique quand $n \to \infty$.

Exercice 4 (Exponentiation rapide de x^n)

On peut remarque que $x^{2k} = (x^k)^2$ et que $x^{2k+1} = x(x^k)^2$

1. Ecrire la fonction `puissance_rapide(x,n)` qui calcule x^n de manière récursive en tenant compte des informations fournies.
2. Combien de multiplication faut-il pour calculer x^n ? Estimer l'ordre de grandeur asymptotique quand $n \to \infty$.