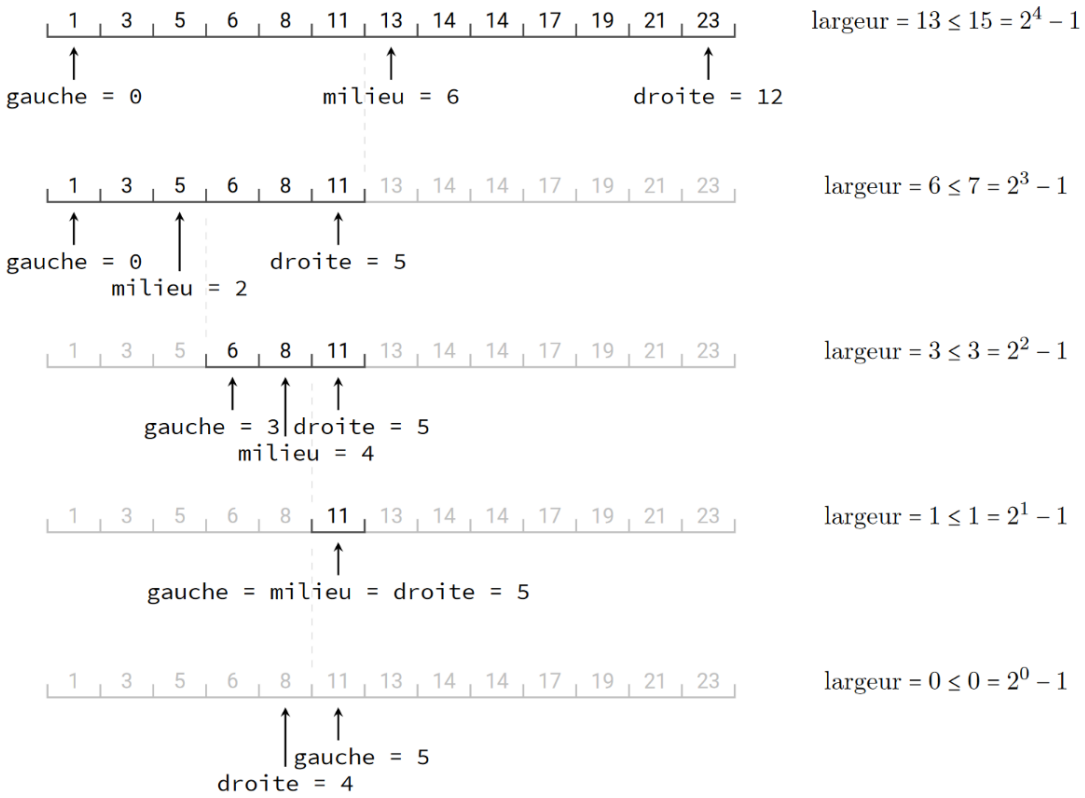
Recherche dichotomique

L'idée centrale de cette approche repose sur l'idée de réduire de moitié l'espace de recherche à chaque étape : on regarde la valeur du milieu et si ce n'est pas celle recherchée, on sait qu'il faut continuer de chercher dans la première moitié ou dans la seconde. Plus précisément, en tenant compte du caractère trié du tableau, il est possible d'améliorer l'efficacité d'une telle recherche de façon conséquente en procédant ainsi :

1. on détermine l'élément `m` au milieu du tableau ;
2. si c'est la valeur recherchée, on s'arrête avec un succès ;
3. sinon, deux cas sont possibles :
   (a) si `m` est plus grand que la valeur recherchée, comme la tableau est trié, cela signifie qu'il suffit de continuer à chercher dans la première moitié du tableau ;
   (b) sinon, il suffit de chercher dans la moitié droite.
4. on répète cela jusque avoir trouvé la valeur recherchée, ou bien avoir réduit l'intervalle de recherche à un intervalle vide, ce qui signifie que la valeur recherchée n'est pas présente.

On recherche la valeur `10` dans le tableau `[3, 3, 5, 6, 8, 11, 13, 14, 14, 17, 19, 21, 23]`.



Une implémentation en Python est la suivante :

```python
def recherche_dichotomique(tab, val):
    gauche = 0
    droite = len(tab) - 1
    while gauche <= droite:
        milieu = (gauche + droite) // 2
        if tab[milieu] == val:
            # on a trouvé val dans le tableau,
            # à la position milieu
            return milieu
        elif tab[milieu] > val:
            # on cherche entre gauche et milieu - 1
            droite = milieu - 1
        else: # on a tab[milieu] < val
            # on cherche entre milieu + 1 et droite
            gauche = milieu + 1
    # on est sorti de la boucle sans trouver val
    return -1
```