I. Structures de données linéaires

- Besoin de stocker/organiser plus d'information / d'éléments dans une variable

1) Tableau

- Éléments stockés dans des espaces mémoire contigus ;
- $\oplus$ Accès indicé aux éléments contenus ($t[i]$) ;
- $\ominus$ Taille fixe.

2) Liste chainée

- Éléments non contigus, chaque "case" renvoie vers la suivante ;
- $\oplus$ Pas besoin de définir la taille à la création, ajout d'éléments aisé ;
- $\ominus$ Accès séquentiel, pas indicé.

II. Recherche dans une structure non triée

1) Présence d'une valeur dans la structure

- Recherche séquentielle, parcours linéaire de la structure ;
- Fin du parcours si la valeur recherchée est trouvée ;
- Si la fin est atteinte alors la valeur n'est pas présente.
- complexité : $O(n)$

```
Entrées : v : la valeur recherchée
tab : le tableau dans lequel on cherche
Sorties : vrai si v est présent dans tab, faux sinon
début
    n ← longueur de tab
    pour i de 0 à n-1 faire
        si tab[i] == n alors
            retourner vrai
        fin
    fin
    retourner faux
fin
```

2) Recherche d'un minimum/maximum local

- Choix d'une valeur référence (premier élément) ;
- Chaque élément est comparé à la valeur référence ;
- Si il est plus petit (plus grand) on met à jour la valeur référence.
- complexité : $O(n)$

```
Entrées : tab : le tableau dans lequel on cherche
Sorties : La valeur minimale présente dans le tableau
début
    n ← longueur de tab
    min ← tab[0]
    pour i de 1 à n-1 faire
        si tab[i] < min alors
            min ← tab[i]
        fin
    fin
    retourner min
fin
```

III. Recherche dans une structure triée

1) Présence d'une valeur dans la structure

- Recherche dichotomique ;
- Besoin d'un accès indicé (impossible avec une liste chaînée).
- complexité : $O(logn)$

```
search(tab, v, debut, fin)
Entrées : v : la valeur recherchée
tab : le tableau dans lequel on cherche
debut : indice de début de la recherche (par défaut 0)
fin : indice de fin (par défaut longueur de tab)
Sorties : vrai si v est présent dans tab, faux sinon
début
    milieu ← (debut + fin) ÷ 2
    si fin - debut ≤ 1 alors
        retourner faux
    sinon
        si tab[milieu] = v alors
            retourner vrai
        sinon
            si t[milieu] > v alors
                retourner search(tab, v, debut, milieu)
            sinon
                retourner search(tab, v, milieu, fin)
            fin
        fin
    fin
fin
```

2) Recherche d'un minimum/maximum local

- Prendre le premier / dernier élément de la structure
- complexité : $O(1)$