Fiche : Algorithmes de recherche

On a souvent besoin de rechercher un élément ou sa position dans un tableau. La plupart des bons algorithmes de recherche séquentielle ou linéaire (parcours de l'ensemble du début jusqu'à la fin) fonctionnent grâce à une organisation astucieuse des données. Par exemple, on sait intuitivement que pour retrouver une carte dans un jeu, il est très utile que le jeu soit trié.

1 Recherche séquentielle

La méthode de recherche la plus simple est la recherche séquentielle qui consiste à examiner les éléments l'un après l'autre. Elle s'effectue en temps linéaire. Elle ne nécessite pas d'avoir une structure de données triée. La fonction suivante recherche de cette façon si `tab` contient la valeur `element`.

[ ]:
```
def recherche_seq(element, tab):
    n = len(tab)
    i = 0
    while i < n:
        if tab[i] == element:
            return i
        i += 1
    return -1
```

La complexité est linéaire dans le pire des cas.

2 Dichotomie

Si le tableau est trié, on peut utiliser cela pour faire une recherche plus rapide : la recherche dichotomique.

Idée générale

Soit `tab` un tableau trié et `i` un indice.

- Si `element < tab[i]` alors tous les éléments "à droite" de `tab[i]` (d'indice supérieur à `i`) sont supérieurs à `element`, et il faut continuer la recherche dans la moitié gauche du tableau,
- si `element > tab[i]` alors tous les éléments "à gauche" de `tab[i]` (d'indice inférieur à `i`) sont inférieurs à `element`, et il faut continuer la recherche dans la moitié droite du tableau.

À chaque itération, on "coupe le tableau" en deux.

[ ]:
```
def recherche_dt(element, tab):
    debut = 0
    fin = len(tab) - 1
```

```
    milieu = (debut + fin) // 2
    while debut <= fin:
        if element == tab[milieu]:  # on a trouvé l'élément
            return milieu
        elif element < tab[milieu]:
            fin = milieu - 1  # on prend la moitié gauche
        else:
            debut = milieu + 1  # on prend la moitié droite
        milieu = (debut + fin) // 2
    return -1
```

- La complexité est logarithmique dans le pire des cas : $\log_2(n)$.
- La recherche dichotomique est plus rapide que la recherche séquentielle en général (complexité en moyenne et dans le pire des cas) mais dans certains cas, la recherche séquentielle peut être plus rapide : si les tableaux sont de petite taille, ou si la valeur à chercher se trouve en début de tableau.

3 Courbes de complexité

[ ]:
```
%matplotlib inline
figure(figsize=(5, 3))
plot(tailles, temps_seq, label="séquentiel")
plot(tailles, temps_dt, label="dichotomie")
xlabel("taille")
ylabel("temps (ms)")
legend()
```

[ ]:
```
<matplotlib.legend.Legend at 0x7f08823e7c10>
```