

Applications de parcours séquentiel d'un tableau

Les algorithmes de parcours séquentiel permettent de parcourir les éléments d'un tableau à l'aide d'une boucle.

Ils font partie des algorithmes fondamentaux en informatique, leur parfaite maîtrise est donc indispensable car ils sont à la base de beaucoup d'autres algorithmes qui doivent passer en revue tous les éléments d'une collection (tableau, tuple, liste, dictionnaires, etc.).

On détaille dans les trois algorithmes au programme :

- l'algorithme de **recherche d'un élément** (ou d'une occurrence)
- l'algorithme de **recherche d'un extremum** (maximum ou minimum)
- l'algorithme de **calcul d'une moyenne**

Spécification de l'algorithme

La **spécification** d'un algorithme précise de manière non ambiguë ce que doit faire un algorithme. En particulier, on y indique : le nom des données manipulées (entrées), le nom des données renvoyées (sorties), le rôle de l'algorithme ainsi que les hypothèses sur les entrées (précondition) et les sorties (postcondition). Un algorithme sera considéré comme *correct* s'il respecte cette spécification.

Algorithme de recherche d'un élément

On veut écrire un algorithme qui cherche si un élément est présent ou non dans un tableau.



Voici la spécification de l'algorithme de recherche d'un élément :

- **Entrée(s)** : un tableau **T** de taille n et une valeur **v**
- **Sortie(s)** : un booléen **trouvee**
- **Rôle** : chercher si la valeur **v** est présente dans **T**
- **Précondition(s)** : **v** est du même type que les éléments de **T**
- **Postcondition(s)** : **trouvee** vaut Vrai si **v** appartient à **T**, et Faux sinon

Algorithme

Dans cette activité, on va utiliser une *boucle for*

Voici l'algorithme :

```
trouvee ← Faux
pour i de 0 à n-1 faire :
    si T[i] = v alors :
        trouvee ← Vrai
    fin si
fin pour
```

Renvoyer trouvee

Coût de l'algorithme

Le **coût d'un algorithme** est le nombre d'opérations élémentaires (arithmétiques ou logiques) ainsi que d'affectations nécessaires à son exécution **dans le pire cas**.

Le coût d'un algorithme dépend toujours de la taille des données d'entrée (ici la taille n du tableau).

Dans tous les cas, notre algorithme va effectuer n tours de boucle, donc dans le pire cas aussi. Ainsi, la comparaison `T[i] = v` sera effectuée n fois.

Si tous les éléments du tableau sont égaux à `v` alors l'affectation `trouvee ← Vrai` est effectuée à chaque tour de boucle, donc n fois.

On dénombre ainsi :

Comparaisons	Affectations	Opérations arithmétiques
n	$n + 1$	0

Le coût de l'algorithme (dans le pire cas) est donc $n + (n + 1) + 0 = 2n + 1$.

Comme ce coût est de l'ordre de n , on dit que le coût de l'algorithme de recherche d'un élément est **linéaire**.

Algorithme de recherche du maximum

Activité d'introduction : un tableau, des feuilles et des aimants

On dispose de 6 feuilles de papiers. Sur chacune d'elle une valeur entière a été écrite.

Ces 6 feuilles de papier sont aimantées au tableau, faces cachées.

On ne peut voir la face visible que d'une seule carte à la fois.

On dispose au tableau d'une case appelée maxi dans laquelle on peut stocker une valeur (entière).

1. Décrire la démarche permettant de répondre à la question : quelle est la valeur maximale ?

2. Dans quel cas faut-il absolument retourner toutes les feuilles de papier pour répondre ?

Spécification de l'algorithme

- Entrées : tableau T de taille n
- Sortie : entier `maxi`
- Rôle : chercher l'élément maximal de T
- Précondition : T est un tableau non vide d'entiers
- Postcondition : `maxi` est un entier, `maxi` est l'élément maximal de T

Algorithme

Coût de l'algorithme

3. Donnez un exemple de tableau que l'on peut considérer comme le pire cas en termes d'opérations élémentaires.
4. Complétez tableau suivant donnant le nombre d'opérations élémentaires dans le pire cas :

Nombre d'affectations	Nombre de comparaisons	Nombre d'opérations arithmétiques

5. Quel est le coût de cet algorithme ?
6. Ecrire une fonction Python appelée `maximum` qui prend en paramètres un tableau T d'entiers et renvoie la valeur maximale du tableau T .

Algorithme de calcul d'une moyenne

On souhaite écrire un algorithme qui calcule la moyenne des valeurs d'un tableau T d'entiers.

Spécification de l'algorithme

1. Proposez une spécification d'un tel algorithme.

- Entrées :
- Sortie :
- Rôle :
- Précondition :
- Postcondition :

Algorithme

2. Proposez un algorithme correspondant à cette spécification.

3. Quel est le coût de cet algorithme ?

Nombre d'affectations	Nombre de comparaisons	Nombre d'opérations arithmétiques

4. Ecrivez une fonction Python appelée moyenne qui prend en paramètres un tableau T d'entiers, non vide, et renvoie la moyenne des valeurs du tableau T . On veut une version avec un parcours par indice et une autre avec un parcours par valeur.