Trier un tableau ? Pas si simple... Comparons différents algorithmes

François Delbot Laurent Pierre

3 octobre 2016

1 Un très mauvais algorithme de tri (1 points)

`Prototype de la fonction : void tri_aleatoire(int *tab,int taille);`

Considérons un tableau non trié. Si on mélange ses éléments aléatoirement, il y a une probabilité non nulle que le résultat soit un tableau trié. On peut donc exploiter cette remarque pour implémenter un algorithme de tri : Tant que le tableau n'est pas trié, mélanger le tableau. Il faut donc, après chaque mélange, vérifier si le tableau est trié. Amusez vous à tester cet algorithme sur des instances de taille $3, 4, 5 \ldots$. Si on considère que toutes les valeurs présentes dans le tableau sont différentes et que notre mélange est uniforme (chaque permutation des éléments du tableau est équiprobable), quelle est la probabilité d'obtenir un tableau trié ? Que pouvons-nous en conclure ?

2 Tri à bulle (1 points)

`Prototype de la fonction : void tri_a_bulle(int *tab, int taille);`

implémentez l'algorithme du tri à bulle :

1. `https://fr.wikipedia.org/wiki/Tri_%C3%A0_bulles`
2. `https://www.youtube.com/watch?v=lyZQPjUT5B4`

3 Tri par selection (2 points)

`Prototype de la fonction : void tri_selection(int *tab,int taille);`

implémentez l'algorithme du tri par selection :

1. `https://fr.wikipedia.org/wiki/Tri_par_s%C3%A9lection`
2. `https://www.youtube.com/watch?v=Ns4TPTC8whw`

4 Tri par insertion (2 points)

`Prototype de la fonction : void tri_insertion(int *tab, int taille);`

implémentez l'algorithme du tri par insertion :

1. `https://fr.wikipedia.org/wiki/Tri_par_insertion`
2. `https://www.youtube.com/watch?v=ROalU379l3U`

5 Tri par dénombrement (2 points)

`Prototype de la fonction : void tri_denombrement(int *tab, int taille, int max);`

Sachant que toutes les valeurs d'un tableau font partie de l'ensemble $\{0..k\}$, on peut facilement compter combien de fois apparait chaque entier dans le tableau à trier. Ensuite il ne reste plus qu'à remplir le tableau avec le bon nombre d'apparitions de chaque entier. Si le chiffre 0 apparait 5 fois, on va remplir les 5 premières cases du tableau par des 0. Ensuite on passe au chiffre 1, puis 2, puis 3 et ainsi de suite, jusqu'à ce que le tableau soit entièrement rempli. Implémentez cet algorithme. Une valeur superieur ou égale à la valeur maximum contenue dans le tableau vous est donnée en argument.

6 Tri shell (2 points)

`Prototype de la fonction : void tri_shell(int *tab,int taille);`

implémentez l'algorithme du tri shell :

1. `https://fr.wikipedia.org/wiki/Tri_de_Shell`
2. `https://www.youtube.com/watch?v=CmPA7zE8mx0`

7 Tri radix (2 points)

`Prototype de la fonction : void tri_radix(int *tab, int taille);`

implémentez l'algorithme du tri radix (tri par base) :

1. `https://fr.wikipedia.org/wiki/Tri_par_base`
2. `https://www.youtube.com/watch?v=LyRWppObda4`

8 Partition d'une partie d'un tableau (2 points)

`Prototype de la fonction : int partition(int *tab, int debut, int fin);`

Ecrire une fonction qui partitionne une sous partie des éléments d'un tableau (délimitée par les indices *debut* et fin) de telle sorte que toutes les valeurs de cette sous partie qui sont inferieures à celle du pivot se trouvent à gauche du pivot, les autres se trouvant à droite du pivot. Par défaut, la première valeur de la sous partie à partitionner servira de pivot (c'est à dire la valeur d'indice *debut*).
Cette fonction doit retourner l'indice du pivot car il risque d'être déplacé.
Exemple :

- tableau en entrée : 1|2|-1|7|4|8|3|9|1|0|2|1|2|3
- debut : 4
- fin : 10
- tableau après partitionnement : 1|2|-1|7|1|0|2|3|4|8|9|1|2|3
- retour de la fonction : 8

9 Fusion de deux sous tableaux (1 points)

`Prototype de la fonction : void fusion(int *tab, int debut,int milieu,int fin);`

Ecrire une fonction qui accepte en arguments un tableau, sa taille et trois indices (*debut*, *milieu* et fin). Ces trois indices délimitent deux sous-parties du tableau, la première dont les cases vont de l'indice *debut* (inclus) à l'indice *milieu* (inclus), et la seconde dont les cases vont de l'indice $milieu + 1$ à l'indice fin. On considère que chacune de ces deux sous parties sont triées par ordre croissant. La fonction qui vous est demandée doit fusionner ces deux sous parties de sorte à ce que les valeurs des cases allant de l'indice *debut* jusqu'à l'indice fin soient triées par ordre croissant.
Exemple :

- tableau donné en entrée : 4|5|7|1|2|3|6|7|2|3|4|5|8|1|1|0|-1|4
- debut : 3
- milieu : 7
- fin : 12
- résultat : 4|5|7|1|2|2|3|3|4|5|6|7|8|1|1|0|-1|4

10 Tri fusion (1 points)

`Prototype de la fonction : void tri_fusion(int *tab, int debut, int fin);`

implémentez l'algorithme du tri fusion au moyen d'une fonction récursive :

1. `https://fr.wikipedia.org/wiki/Tri_fusion`
2. `https://www.youtube.com/watch?v=XaqR3G_NVoo`

On suppose que l'on possède déjà une fonction permettant de fusionner deux parties d'un tableau, délimitées par les indices *debut* à *milieu* et $milieu + 1$ à fin. Le prototype de cette fonction de fusion est le suivant :

```
void fusion(int *tab, int debut,int milieu,int fin);
```

11 Partition d'une partie d'un tableau (2 points)

`Prototype de la fonction : int partition(int *tab, int debut, int fin);`

Ecrire une fonction qui partitionne une sous partie des éléments d'un tableau (délimitée par les indices *debut* et fin) de telle sorte que toutes les valeurs de cette sous partie qui sont inferieures à celle du pivot se trouvent à gauche du pivot, les autres se trouvant à droite du pivot. Par défaut, la première valeur de la sous partie à partitionner servira de pivot (c'est à dire la valeur d'indice *debut*).
Cette fonction doit retourner l'indice du pivot car il risque d'être déplacé.
Exemple :

- tableau en entrée : 1|2|-1|7|4|8|3|9|1|0|2|1|2|3
- debut : 4
- fin : 10
- tableau après partitionnement : 1|2|-1|7|1|0|2|3|4|8|9|1|2|3
- retour de la fonction : 8

12 Tri rapide (2 points)

`Prototype de la fonction : void tri_rapide(int *tab,int debut, int pivot, int fin);`

implémentez l'algorithme du tri rapide (Quick Sort) de manière récursive :

1. `https://fr.wikipedia.org/wiki/Tri_rapide`
2. `https://www.youtube.com/watch?v=ywWBy6J5gz8`

On suppose que l'on possède déjà une fonction permettant de partitionner les valeurs d'une partie d'un tableau et qui retourne l'indice du pivot. Le prototype de cette fonction de partition est le suivant :

```
int partition(int *tab, int debut, int fin);
```