

M1.L3 : Série d'exercices sur les algorithmes / complexité

1 Ordre de complexité des algorithmes de la série M1.L2

Indiquer l'ordre de complexité avec la notation $O(\)$ pour chacun des algorithmes suivants étudiés dans la série précédente :

- 1.1) les algorithmes de conversion travaillent tous en un nombre borné d'étapes quel que soit la donnée qui est fournie en entrée. Qu'en déduit-on comme ordre de complexité ?
 - 1.2) Détermination si un nombre entier n est premier en fonction de n (un exercice de la série M1.L5 offrira une analyse différente de ce problème)
 - 1.3.1) multiplication égyptienne (devinette1) en fonction de y
 - 1.3.2) Algorithme d'Euclide en fonction du max des paramètres x et y
 - 1.4.1) Détermination de la plus petite valeur d'une liste L de taille n
 - 1.4.2) Détermination de la plus petite différence dans une liste non-triée de taille n
 - 1.4.3) **supplément à la question 1.4.2** : On suppose que la liste L fournie à l'algorithme est d'abord triée dans une première étape avant d'effectuer la recherche de la plus petite différence sur cette liste triée. Voici cet algorithme ci-dessous.
- a) Exécutez-le sur une liste de plusieurs éléments dans le désordre pour vérifier son fonctionnement

La plus petite différence-2
entrée : Liste L d'entiers, de taille n , contenant au moins 2 éléments sortie : a, b nombres avec la plus petite différence
$L_{triee} = \text{tri}(L)$ $a \leftarrow L_{triee}(1)$ $b \leftarrow L_{triee}(2)$ $\delta_{min} \leftarrow L(2) - L(1) $ Pour i de 1 à $n - 1$ $\delta \leftarrow L_{triee}(i + 1) - L_{triee}(i) $ Si $\delta < \delta_{min}$ $\delta_{min} \leftarrow \delta$ $a \leftarrow L_{triee}(i)$ $b \leftarrow L_{triee}(i + 1)$ sortir : (a, b)

b) Quel est son ordre de complexité ?

2 Manipulation de liste

On considère l'algorithme *algorithm1*. L'entrée de l'algorithme est une liste L et le nombre n d'éléments dans L avec $n \geq 1$. L'algorithme utilise la fonction **swap(L[i],L[j])** qui échange les éléments dans la liste L aux positions i et j .

algorithm1
entrée : L, n sortie : quelle transformation est effectuée sur L ?
Pour i de 1 à $n - 1$ $jMin \leftarrow i$ Pour j de $i + 1$ à n Si $L[j] < L[jMin]$ $jMin \leftarrow j$ Si $jMin \neq i$ swap ($L[i], L[jMin]$)

Exécutez cet algorithme sur les listes de tailles différentes contenant des nombres entiers ; commencez par les cas limite (n valant 1) puis continuez avec des tailles de liste plus importantes jusqu'à être certain du but de l'algorithme. Aidez-vous d'un dessin pour suivre la progression du déroulement de l'algorithme.

- Votre conclusion : de manière générale que fait cet algorithme?
- Sachant que la fonction **swap** a un coût constant, quel est son ordre de complexité en fonction de n ?

3 Taille de liste

Soit L une liste d'entiers (pas forcément ordonnée et avec de possibles répétitions), comme par exemple {19, 31, 15, 21}.

On cherche ici à écrire un algorithme **taille(L)** qui retourne le nombre d'éléments d'une liste L donnée en entrée. On suppose que l'on dispose d'un autre algorithme **a_element(L, i)** qui nous dit (vrai ou faux) s'il existe un i^{e} élément dans la liste : il répond donc **vrai** si i est inférieur ou égal la taille de la liste et **faux** sinon.¹

¹. Notez qu'il n'est pas évident de toujours avoir un tel algorithme (sans avoir **taille**, bien sûr!). En réalité cela dépend de la représentation fective de L . Mais nous faisons ici l'hypothèse qu'un tel algorithme existe pour L .

a) Comment utiliser l'algorithme $a_element(L,i)$ pour calculer la taille de L en un temps linéaire (par rapport cette taille)? Ecrivez un algorithme pour le faire.

b) En vous inspirant de la recherche dichotomique (cf. cours), pourrait-on avoir une complexité moindre que linéaire en s'aidant par exemple de la suite des puissances de deux ?

Si oui, quelle serait cette complexité et expliquez intuitivement comment procéder.

Essayez d'écrire un tel algorithme.

4 Que font ces algorithmes?

Pour chacun des algorithmes suivants, indiquer (en mots) quelle est la sortie de l'algorithme.

Exprimer leur ordre de complexité avec la notation $O(\)$ de Landau en fonction du paramètre d'entrée n (ou du max des paramètres a et b pour le second algorithme).

Algorithme a)
entrée : n nombre naturel sortie : ??
$m \leftarrow n$ $i \leftarrow 1$ Tant que $m > 0$ $i \leftarrow 2i$ $m \leftarrow m - 1$ Sortir : i

Algorithme b)
entrée : a, b nombres naturels non-nuls sortie : ??
$s \leftarrow 0$ Si $a < b$ Pour i allant de 1 à a $s \leftarrow s + b$ Sinon Pour i allant de 1 à b $s \leftarrow s + a$ Sortir : s

Algorithme c)
entrée : L liste de nombres entiers, n taille de la liste sortie : ??
$s \leftarrow \text{oui}$ Pour i allant de 1 à $n - 1$ Si $L(i + 1) < L(i)$ $s \leftarrow \text{non}$ Sortir : s

Algorithme d)
entrée : L liste de nombres entiers, n taille de la liste, d nombre naturel non-nul sortie : ??
$s \leftarrow \text{non}$ Pour i allant de 1 à $n - 1$ Pour j allant de $i + 1$ à n Si $ L(j) - L(i) < d$ $s \leftarrow \text{oui}$ Sortir : s