Algorithmique III (I41) - Licence d'Informatique

Contrôle terminal (Session 1 - Mai 2022)

Dans la suite, on ne considère que des listes S de longueur n telles que $S([\![1,n]\!]) = [\![1,r]\!]$ où $1 \leqslant r \leqslant n$. Un élément de S est dit *majoritaire* si son nombre d'occurrences est strictement supérieur à $\lfloor \frac{n}{2} \rfloor$. Par exemple 1 est majoritaire dans la liste $[1,2,1,3,1,1,3,1]$ et 2 dans la liste $[2,1,2,2,1]$.

1. [1.0] Formalisez la proposition "*S contient un élément majoritaire.*" en logique des prédicats.

Solution. Voilà une proposition possible :

$$\exists m \in [\![1,r]\!] \quad \#\{i \in [\![1,n]\!] \mid S[i] = m\} > \lfloor \frac{n}{2} \rfloor. \tag{1}$$

2. [1.0] Démontrez qu'une liste contient au plus un élément majoritaire.

Solution. S'il existait deux éléments majoritaires avec pour nombres d'occurrence n_1 et n_2, d'après (1) on aurait $n_1 \geqslant \lfloor \frac{n}{2} \rfloor + 1$ et $n_2 \geqslant \lfloor \frac{n}{2} \rfloor + 1$ et par conséquent $n_1 + n_2 \geqslant 2\lfloor \frac{n}{2} \rfloor + 2 > n$, ce qui est impossible.

3. [2.0] Inspirez-vous du tri par dénombrement pour écrire un algorithme `Majoritaire(L)` de complexité $\Theta(n)$ qui renvoie l'élément majoritaire s'il existe, et 0 sinon. Justifiez.

Solution. L'algorithme opère en 3 étapes. La première recherche la valeur maximale r dans la liste S pour allouer et initialiser à 0 les r valeurs d'un histogramme H (une table d'entiers). La deuxième parcourt la liste S en incrémente le nombre d'occurrences de l'élément courant dans l'histogramme. La troisième cherche la valeur maximale dans H et la compare à $\lfloor \frac{n}{2} \rfloor$ pour conclure. Voir l'algorithme dans Algo. 1 plus loin (on y a également rappelé l'algorithme de recherche de l'indice du maximum dans une liste).

La recherche de la valeur maximale dans S a un coût de $\Theta(n)$ et la mise à jour de l'histogramme H également. La recherche du maximum dans H a un coût de $\Theta(r)$ mais $r = O(n)$ puisque $S([\![1,n]\!]) = [\![1,r]\!]$, l'algorithme `Majoritaire` a bien une complexité en $\Theta(n)$.

NB. L'algorithme calque le fonctionnement du tri par dénombrement comme c'était suggéré, mais peut améliorer (marginalement) la complexité dans le meilleur des cas en sortant de la boucle `TQ` dès que $H[S[i]] > \lfloor \frac{n}{2} \rfloor$, avec la condition `(i <= #S) ET (H[S[i]] <= n DIV 2)` et en testant en sortie de boucle la valeur de i pour décider s'il faut renvoyer `S[i]` ou `0`.

```
ALGORITHME Majoritaire(S):entier
DONNEES
   S: liste d'entiers
VARIABLES
   i,r: entiers
   H: tableau d'entiers
DEBUT
 · r ← S[iMax(S)]
 · Allouer(H,r,0)
 · i ← 1
 · TQ (i <= #S) FAIRE
 ·  · H[S[i]] = H[S[i]] + 1
 ·  · i ← i + 1
 · FTQ
 · r ← iMax(H)
 · SI (H[r] > n DIV 2)
 ·  · RENVOYER r
 · SINON
 ·  · RENVOYER 0
 · FSI
FIN
```

```
ALGORITHME iMax(L):entier
DONNEES
   L: liste d'entiers
VARIABLES
   i,imax: entiers
DEBUT
 · imax ← 1
 · i ← 2
 · TQ (i <= #L) FAIRE
 ·  · SI (L[i] > L[imax]) ALORS
 ·  ·    imax ← i
 ·  · FSI
 ·  · i ← i + 1
 · FTQ
 · RENVOYER imax
FIN
```

Algo. 1. Algorithmes `Majoritaire` et `iMax`.

On se propose de résoudre différemment le problème de la recherche de l'élément majoritaire, s'il existe. Pour la compréhension de ce qui va suivre, les éléments des listes sont interprétés comme des balles de couleur, chaque valeur entière codant une couleur différente. Toutes les listes sont exploitées comme des piles à l'aide des algorithmes suivants de complexité $\Theta(1)$ que l'on utilisera *sans les écrire* :

- `Empiler(@P,x)` empile x au sommet de la pile P.
- `Depiler(@P)` dépile la valeur au sommet de la pile P et la renvoie.
- `Sommet(P)` renvoie la valeur au sommet de la pile P (sans dépiler).

La pile S est appelée le *sac* et l'algorithme utilise deux piles auxiliaires, une *étagère* E et une *corbeille* C initialement vides. Il opère en deux phases.

Phase 1. On cherche à ranger les balles sur l'étagère de manière à ce qu'il n'y ait jamais deux balles de la même couleur côte à côte. Tant que le sac n'est pas vide on y retire une balle x que l'on range sur l'étagère si celle-ci est vide (étape initiale a dans l'exemple suivant) ou si la première balle qu'elle contient est différente de x (étapes b, e et f), et dans ce cas, on retire également une balle de la corbeille (si elle n'est pas vide) qu'on rajoute sur l'étagère à côté de x (étape d) ; sinon on met x à la corbeille C (étapes c et g).

Exemple. On considère le sac $S =$ ○●●●●●● qui contient des balles de 4 "couleurs" différentes dans lequel ● est majoritaire. La flèche ← indique un futur dépilement et les flèches → et ⇒ indiquent respectivement un ou deux futur(s) empilement(s) :

a) $S \leftarrow$ ○●●●●●●
$E \rightarrow$
$C =$

b) $S \leftarrow$ ●●●●●●
$E \rightarrow$ ○
$C =$

c) $S \leftarrow$ ●●●●●
$E =$ ●○
$C \rightarrow$

d) $S \xleftarrow{1}$ ●●●●
$E \Rightarrow$ ●○
$C \xleftarrow{2}$ ●

e) $S \leftarrow$ ●●●
$E \rightarrow$ ●●●○
$C =$

f) $S \leftarrow$ ●●
$E \rightarrow$ ●●●●○
$C =$

g) $S \leftarrow$ ●
$E =$ ●●●●●○
$C \rightarrow$

h) $S =$
$E =$ ●●●●●○
$C =$ ●

4. [1.5] Démontrez (faites une récurrence) qu'après chaque itération de la phase 1, s'il y a des balles dans la corbeille, elles sont nécessairement de la même couleur que la première balle sur l'étagère.

Solution. On définit le prédicat $P(k)$ sur $[\![1, n]\!]$ suivant : *"Après la k-ème itération, toutes les balles dans la corbeille sont de la même couleur que la première balle sur l'étagère."*

Soit $n := \#S$ le nombre de balles dans le sac S et S_k, E_k et C_k le contenu des 3 piles après la k-ème itération pour $k \in [\![0, n]\!]$ avec $S_0 = S$ et $E_0 = C_0 = \varnothing$ le contenu des 3 piles initialement. Notons $c_k := \#C_k$ le nombre de balles dans la corbeille après la k-ème itération. Par convention, nous noterons $S[i]$ le i-ème terme de la pile avec $S[1]$ son sommet. Le prédicat $P(k)$ s'écrit alors

$$(c_k > 0) \Rightarrow \forall i \in [\![1, c_k]\!] \;\; C_k[i] = E_k[1]. \qquad (P(k))$$

La première balle est empilée sur l'étagère donc C_1 est vide et $P(1)$ est vrai (on rappelle que $A \Rightarrow B$ est toujours vrai si A est faux). Considérons $b := S_k[1]$, la $(k+1)$-ème balle extraite du sac S.

i. Si b est égale à la première balle sur l'étagère, i.e. $b = E_k[1]$, alors b est empilée sur la corbeille qui ne contient alors que des balles identiques à $E_k[1]$, soit parce qu'elle était vide, soit parce qu'elle ne contenait que des balles identiques à $E_k[1]$ d'après l'hypothèse de récurrence $P(k)$. Donc $P(k+1)$ est vrai.

ii. Sinon b est différente de la première balle sur l'étagère i.e. $b \neq E_k[1]$, et elle y est empilée. Soit la corbeille est vide et la preuve est terminée, soit on en extrait une balle c qu'on empile sur l'étagère. Dans ce dernier cas, la corbeille a été vidée ou ne contient que des balles égales à c d'après l'hypothèse de récurrence. Donc $P(k+1)$ est vrai.

5. [0.5] Formalisez la proposition *"Deux balles côte à côte sur l'étagère ne sont jamais de la même couleur."* en logique des prédicats.

Solution. Rien de bien compliqué :

$$\forall i \in [\![1, \#E - 1]\!] \;\; E[i] \neq E[i+1].$$

et si l'on souhaitait intégrer les différents états E_k de l'étagère à chaque itération $k \in [\![0, n]\!]$ de l'algorithme avec $n := \#S$ le nombre de balles :

$$\forall k \in [\![0, n]\!] \; \forall i \in [\![1, \#E_k - 1]\!] \;\; E_k[i] \neq E_k[i+1].$$

6. [1.5] En déduire qu'après la phase 1, s'il y a une couleur majoritaire, c'est nécessairement celle de la première balle sur l'étagère.

Solution. Si toutes les balles de couleur majoritaire ne sont pas sur l'étagère, il y en a dans la corbeille, or on a prouvé précédemment que les balles de la corbeille sont toutes de la même couleur que la première sur l'étagère qui est donc de la couleur majoritaire. Sinon toutes les balles de la couleur majoritaire sont sur l'étagère et la corbeille est vide. Comme il ne peut y avoir deux balles de la même couleur côte à côte sur l'étagère,

ceci n'est possible que si le nombre de balles est impair, disons $n = 2k+1$, et qu'il y a exactement $k+1$ balles de la couleur majoritaire, une à chaque extrémité de l'étagère et toutes en alternance avec une balle d'une autre couleur. La première balle de l'étagère est encore une fois de la couleur majoritaire.

7. [1.0] À l'aide d'un contre-exemple, montrez qu'après la phase 1, la corbeille peut contenir des balles sans qu'il y ait d'élément majoritaire (d'où la nécessité d'une phase 2).

Solution. À l'issue de la phase 1 pour la liste $S = [1, 2, 3, 3]$, l'étagère est $E = [3, 2, 1]$ et la corbeille $C = [3]$ mais 3 n'est pas majoritaire dans la liste S.

8. [2.5] Écrivez l'algorithme `Phase1(@S,@E,@C)` qui réalise la première phase. Calculez sa complexité.

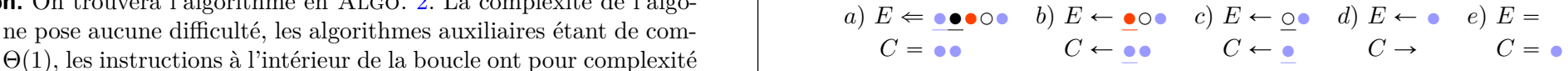
Solution. On trouvera l'algorithme en Algo. 2. La complexité de l'algorithme ne pose aucune difficulté, les algorithmes auxiliaires étant de complexité $\Theta(1)$, les instructions à l'intérieur de la boucle ont pour complexité $\Theta(1)$ et comme on passe n fois dans la boucle, on a $T(n) = n\Theta(1) = \Theta(n)$.

```
ALGORITHME Phase1(@S,@E,@C)
DONNEES S,E,C: listes
VARIABLES
   balle: entier
DEBUT
 · TQ (S != []) FAIRE
 ·  · balle ← Depiler(S)
 ·  · SI ((E = []) OU (balle != Sommet(E))) ALORS
 ·  ·  · Empiler(E, balle)
 ·  ·  · SI (C != []) ALORS
 ·  ·  ·  · Empiler(E, Depiler(C))
 ·  ·  · FSI
 ·  · SINON
 ·  ·  · Empiler(C, balle)
 ·  · FSI
 · FTQ
FIN
```

Algo. 2. Algorithme `Phase 1`.

Phase 2. On cherche à éliminer les balles deux par deux, une de l'étagère et la deuxième, de l'étagère à nouveau ou de la corbeille en s'assurant qu'une des deux est de la couleur potentiellement majoritaire. Soit c la couleur de la première balle sur l'étagère. Tant que l'étagère n'est pas vide, on y retire une balle et si sa couleur est c, alors soit l'étagère est vide et on met cette balle dans la corbeille (étape d), soit elle ne l'est pas et on y retire une deuxième balle (la phase 1 assure qu'elle n'est pas de la même couleur, étape a) ; sinon, soit la corbeille est vide et il n'y a pas d'élément majoritaire, soit on retire une balle de la corbeille (étapes b et c). Si on a pu vider l'étagère et que la corbeille n'est pas vide, l'élément majoritaire est dans la corbeille (étape e), sinon il n'y a pas d'élément majoritaire.

Exemple. Les flèches ← et ⇐ indiquent respectivement un ou deux futur(s) dépilement(s) et la flèche → un futur empilement. Après la phase 1 pour la liste $S =$ ●○●●●●●, on a obtenu :



9. [2.5] Écrivez l'algorithme `Phase2(@E,@C)` qui réalise la seconde phase et renvoie la couleur de l'élément majoritaire s'il existe et 0 sinon.

Solution. On trouvera l'algorithme en Algo. 3 plus loin.

10. [1.0] Appliquez les deux phases à $S := [1, 2, 1, 3, 1, 1]$ en détaillant les opérations (cf. exemples) mais avec les entiers.

Solution. Étapes de la première phase :

$S \leftarrow [1, 2, 1, 3, 1, 1]$	$S \leftarrow [2, 1, 3, 1, 1]$	$S \leftarrow [1, 3, 1, 1]$	$S \leftarrow [3, 1, 1]$
$E \rightarrow []$	$E \rightarrow [1]$	$E \rightarrow [2, 1]$	$E \rightarrow [1, 2, 1]$
$C = []$	$C = []$	$C = []$	$C = []$

$S \leftarrow [1, 1]$	$S \leftarrow [1]$	$S = []$
$E \rightarrow [3, 1, 2, 1]$	$E = [1, 3, 1, 2, 1]$	$E = [1, 3, 1, 2, 1]$
$C = []$	$C \rightarrow []$	$C = [1]$

```
ALGORITHME Phase2(@E,@C):entier
DONNEES E,C: listes
VARIABLES
   balle, couleur: entier
DEBUT
 · couleur ← Sommet(E)
 · TQ (E != []) FAIRE
 ·  · balle ← Depiler(E)
 ·  · SI (balle = couleur) ALORS
 ·  ·  · SI (E = []) ALORS
 ·  ·  ·  · Empiler(C, balle)
 ·  ·  · SINON
 ·  ·  ·  · Depiler(E)
 ·  ·  · FSI
 ·  · FSI
 ·  · SINON
 ·  ·  · SI (C = []) ALORS
 ·  ·  ·  · RENVOYER 0
 ·  ·  · SINON
 ·  ·  ·  · Depiler(C)
 ·  ·  · FSI
 ·  · FSI
 · FTQ
 · SI (C != []) ALORS
 ·  · RENVOYER Sommet(C)
 · SINON
 ·  · RENVOYER 0
 · FSI
FIN
```

Algo. 3. Algorithme `Phase 2`.

Étapes de la seconde phase :

$E \Leftarrow [1,3,1,2,1]$	$E \Leftarrow [1,2,1]$	$E \leftarrow [1]$	$E = []$
$C = [1]$	$C = [1]$	$C \rightarrow [1]$	$C = [1,1]$

11. [1.0] Calculez le nombre de comparaisons de couleurs dans le pire des cas pour les deux phases.

Solution. Dans tous les cas, il y en a exactement $n-1$ dans la première phase si l'on considère que la condition `balle != Sommet(E)` n'est pas évaluée la première fois parce que la condition `E = []` est satisfaite. Dans la phase 2, le pire des cas correspond à la situation où il y a exactement $\lfloor \frac{n}{2} \rfloor + 1$ balles majoritaires et qu'elles sont toutes sur l'étagère E après la première phase (sauf une dans la corbeille si n est pair). On compare alors la couleur d'une balle sur deux de l'étagère à la couleur majoritaire et on fait donc $\lfloor \frac{n}{2} \rfloor$ comparaisons de couleurs.

12. [0.5] L'algorithme (incluant les deux phases) opère-t-il *in situ* ?

Solution. Oui puisque les n valeurs circulent entre les différentes listes et il n'y a ni pas de création d'espace autres que pour un nombre constant de variables locales.

NB. l'hypothèse initiale $S([\![1,n]\!]) = [\![1,r]\!]$ où $1 \leqslant r \leqslant n$ n'était nécessaire que pour assurer une complexité linéaire au premier algorithme de type dénombrement en limitant la dispersion de la liste. Plus généralement, l'algorithme de recherche de l'élément majoritaire étudié ici opère sur des listes *quelconques*, est *in situ* et *toujours* linéaire, que la liste ait une dispersion en $O(n)$ ou pas.