Algo 701 – Examen

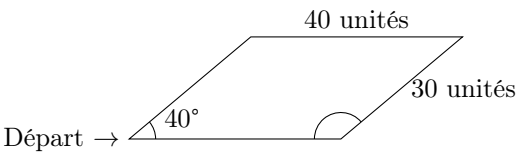
M1 MEEF maths 2022-2023, UGA

12 décembre 2022 – Durée: 2h – Documents non autorisés

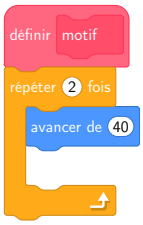
1 Frises en Scratch

Tiré d'un sujet de brevet : Centres étrangers, série générale, 14 juin 2022.

Un professeur donne à ses élèves les éléments de la figure 1 : un motif en forme de parallélogramme et le script, en partie rédigé, qui permet de tracer ce motif. La figure précise le point de départ du lutin, son orientation initiale est vers la droite.



Parallélogramme obtenu



Script du motif

FIGURE 1 – Données de départ pour l'exercice 1.

1. Recopier et compléter le script du motif pour que le bloc « motif » trace le parallélogramme attendu.

Le professeur demande ensuite à ses élèves d'intégrer ce script dans un programme de leur choix permettant de tracer des figures composées de plusieurs de ces motifs. La figure 2 donne deux programmes écrits par des élèves. On rappelle que « s'orienter à 90 » signifie que l'on s'oriente vers la droite.

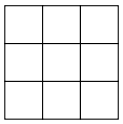
2. Quel évènement permet de lancer le programme de l'élève B ?

3. Parmi les dessins proposés dans la figure 3, indiquer, en justifiant, lequel est obtenu avec le programme de l'élève A et lequel est obtenu avec le programme de l'élève B.

2 Un autre genre de programme

Adapté et étendu à partir d'un sujet de brevet : Amérique du Nord, série générale, 3 juin 2021.

On dispose d'un tableau carré de la forme ci-contre partagé en neuf cases. Chacune de ces cases est colorée en blanc ou en gris. Le tableau constitue alors un motif. Quatre instructions décrites ci-dessous permettent de changer la couleur des cases. Pour chaque question de l'exercice on se placera dans une situation initiale où toutes les cases sont blanches.

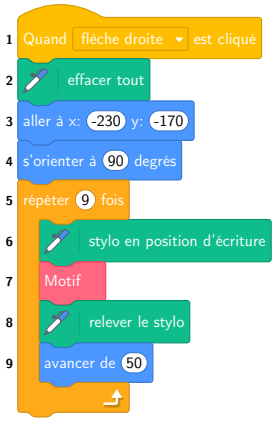


Quatre instructions A, B, C et E permettent de changer l'aspect de certaines cases, lorsqu'on applique ces instructions. L'effet des instruction est décrit dans la table 1. Dans les exemples donnés, « Effet de l'instruction » montre le résultat de l'instruction sur la grille initiale vierge. Si une case du motif est déjà grise et qu'une instruction demande de la griser, alors cette case ne change pas de couleur et reste grise à la suite de cette instruction. La table 2 donne quelques exemples.

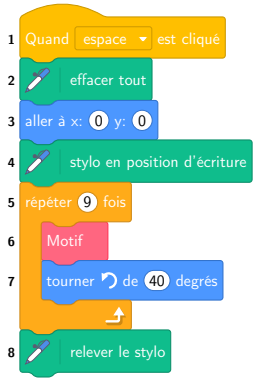
Pour chacune des questions suivantes, on dispose au départ d'un motif dont toutes les cases sont blanches.

1. Représenter le motif obtenu avec la suite d'instructions A B.

2. Parmi les quatre propositions suivantes, deux propositions permettent d'obtenir le motif ci-contre. Lesquelles ?



Programme de l'élève A



Programme de l'élève B

FIGURE 2 – Programmes produits par les élèves

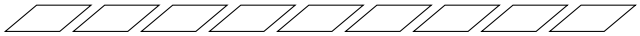


Figure A

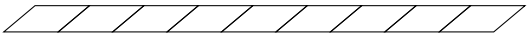


Figure B

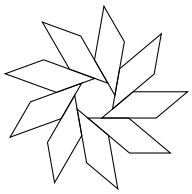


Figure C

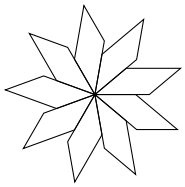


Figure D

FIGURE 3 – Propositions de figures obtenues avec les programmes des élèves

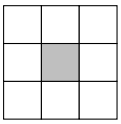
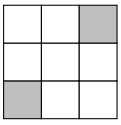
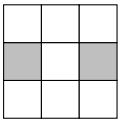
Instruction	Description	Effet de l'instruction
A	La case centrale du motif est grisée.	
B	Dans le motif, la case en bas à gauche et la case en haut à droite sont grisées.	
C	Dans le motif, la case médiane à gauche et la case médiane à droite sont grisées.	
E	Les couleurs du motif sont inversées : les cases blanches deviennent grises et les les cases grises deviennent blanches.	Inverser les couleurs

TABLE 1 – Effet des différents instructions

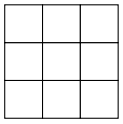
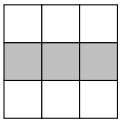
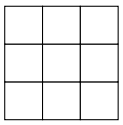
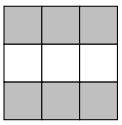
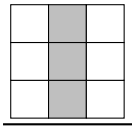
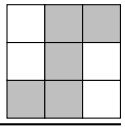
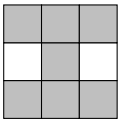
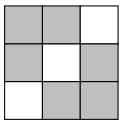
État initial	Programme	État final
	A C	
	A C E	
	A B	

TABLE 2 – Exemples d'effet de différentes suites d'instructions



Proposition 1 : A B C
Proposition 3 : B C E C
Proposition 2 : C E
Proposition 4 : C A E A

3. Donner une suite d'instructions qui permet d'obtenir ce motif : 

4. Déterminer un motif que l'on ne peut pas obtenir avec ces instructions à partir d'un motif entièrement blanc.

5. Combien de motifs différents peut-on atteindre ?

3 Recherche d'un élément majoritaire

Étant donnée une liste L de n éléments (dont certains peuvent apparaître plusieurs fois), le problème de l'élément majoritaire consiste à déterminer s'il existe un élément dont le nombre d'occurrences (c'est-à-dire le nombre de fois où il apparaît dans L) est strictement plus grand que $n/2$. Quelques exemples :

— La liste $[8, 9, 9, 6, 5, 9, 3, 9, 8, 9, 9]$ a un élément majoritaire : 9.

— La liste $[7, 5, 9, 6, 1, 4, 7, 5, 7, 8]$ n'a pas d'élément majoritaire.

— La liste $[3, 5, 5, 3, 5, 3, 3, 5]$ n'a pas d'élément majoritaire.

Dans cet exercice, on étudie différents algorithmes pour déterminer si une liste a un élément majoritaire, et le donner si c'est le cas. Ces algorithmes vont donc prendre en entrée une liste L et renvoyer soit un élément majoritaire, soit l'information qu'il n'y en a pas. Dans tous les cas, l'opération de base consistera à comparer des éléments entre eux pour déterminer s'ils sont égaux (on ne fera pas d'opérations sur les valeurs, ni de tri).

On fixe quelques notations, pour une liste L :

— on note $|L|$ le nombre d'éléments de L,

— on note $L[i]$ l'élément de rang i, pour $0 \leq i \leq |L| - 1$,

— on note $L[i : j]$ la sous-liste de L comportant les éléments de d'indices i inclus à j exclu.

On commence par établir un algorithme simple.

1. Écrire un algorithme qui, étant donnés une liste L et une valeur v, compte le nombre d'occurrences de v dans L.

2. Écrire un algorithme qui résout le problème de l'élément majoritaire de façon « naïve », c'est-à-dire en considérant chaque valeur l'une après l'autre en comptant son nombre d'occurrences.

3. Combien cet algorithme peut-il faire de comparaisons, dans le pire des cas ?

On étudie maintenant une méthode récursive selon l'approche « diviser pour régner ». On commence par diviser la liste L en deux sous-listes L_1 et L_2 de même taille, peu importe comment (l'ordre des éléments n'a pas d'importance). Si L est de longueur impaire, on met un élément de plus dans L_1 que dans L_2.

4. Montrer qu'un élément qui n'est majoritaire ni dans L_1 ni dans L_2 ne peut pas être majoritaire dans L.

5. En déduire un algorithme pour le calcul d'un élément majoritaire en complétant la trame suivante (ou pourra appeler l'algorithme qui compte le nombre d'occurrences d'une valeur) :

```
Algorithme majoritaire-récursif :
    Si |L| = 0 alors renvoyer ...
    Si |L| = 1 alors renvoyer ...
    L1 ← ...
    Chercher un élément majoritaire dans L1 avec majoritaire-récursif.
    Si un élément majoritaire x est trouvé, alors
        ...
    sinon
        L2 ← ...
        chercher un élément majoritaire dans L2 avec majoritaire-récursif.
        Si un élément majoritaire y est trouvé, alors
            ...
        sinon
            ...
```

On admettra que cet algorithme fait un nombre de comparaisons de l'ordre de $n \log n$, où n est le nombre d'éléments de L.

Pour finir, on étudie un algorithme qui permet de résoudre le problème avec une complexité optimale. L'algorithme utilise deux listes auxiliaire C et E peut s'écrire de la façon suivante :

```
C ← [ ]
E ← [ ]
Pour chaque x dans L,
    si |E| > 0 et E[|E| − 1] = x, alors
        ajouter x à la fin de C
    sinon
        ajouter x à la fin de E,
        si |C| > 0, enlever un élément de C et l'ajouter à la fin de E.
m ← E[|E| − 1]
c ← nombre d'occurrences de m dans E + C
si c > n/2, alors répondre que m est élément majoritaire,
          sinon répondre qu'il n'y a pas d'élément majoritaire.
```

On commence par prouver que l'algorithme est correct.

6. Montrer par récurrence qu'à la fin de chaque itération de la boucle, tous les éléments de E sont égaux au dernier élément de E.

7. Montrer que deux éléments consécutifs dans E sont toujours distincts.

8. En déduire que seule le dernier élément de E en fin de boucle peut être majoritaire.

On finit par s'intéresser à la complexité de l'algorithme.

9. Déterminer combien de comparaisons fait cet algorithme en fonction de la taille de la liste L.

10. Justifier qu'il ne peut pas exister d'algorithme significativement plus efficace.