

SPÉCIFICATION DU PROJET “Dames chinoises”
De Q1 à Q9 Edition

What's each language's stance on semicolons?

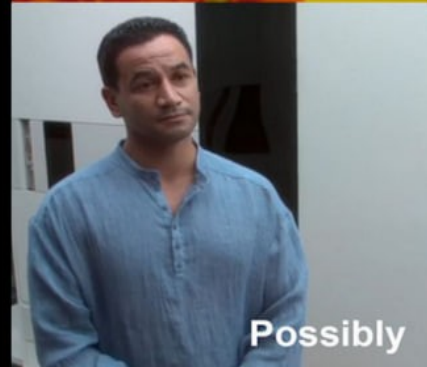


Table des matières

1. Types.....	3
1.1 dimension.....	3
1.2 case.....	3
1.3 couleur.....	3
1.4 case_coloree.....	3
1.5 configuration.....	3
1.6 coup.....	4
1.7 vecteur.....	4
2. Fonctions.....	5
2.1 indice_valide.....	5
2.2 est_case.....	5
2.3 est_dans_losange.....	6
2.4 est_dans_losange_2.....	7
2.5 est_dans_losange_3.....	8
2.6 est_dans_etoile.....	9
2.7 tourner_case.....	10
2.8 translate.....	11
2.9 diff_case.....	11
2.10 sont_cases_alignee.....	12
2.11 dist_entre_coordonnees.....	12
2.12 max_dist_cases.....	13
2.13 min_dist_cases.....	13
2.14 compte_cases.....	13
2.15 sont_cases_voisines.....	14
2.16 calcul_pivot.....	14
2.17 vec_et_dist.....	15

1. Types

1.1 dimension

SPÉCIFICATION – dimension	
PROFIL	$\text{dimension} \stackrel{\text{def}}{=} \mathbb{N}^*$
SÉMANTIQUE	dimension est une dimension d'un plateau, noté dim par la suite, est un paramètre qui encode la taille du plateau. Le plateau à $4\text{dim}+1$ lignes horizontales numérotées de bas en haut de -2dim à 2dim et similairement pour les lignes obliques.

1.2 case

SPÉCIFICATION – case	
PROFIL	$\text{case} \stackrel{\text{def}}{=} \{(i, j, k) \in \mathbb{Z}^3 \text{ tel que } i+j+k=0\}$
SÉMANTIQUE	<p>case est définie par trois coordonnées (i, j, k), la case au centre du plateau de jeu a pour coordonnées $(0, 0, 0)$. Les coordonnées représentent:</p> <ul style="list-style-type: none"> • i le numéro de la ligne horizontale ; • j le numéro de la ligne horizontale lorsqu'on a tourné le plateau d'un tiers de tour dans le sens anti-horaire ; • k le numéro de la ligne horizontale lorsqu'on a tourné le plateau d'un tiers de tour dans le sens horaire.

1.3 couleur

SPÉCIFICATION – couleur	
PROFIL	$\text{couleur} \stackrel{\text{def}}{=} \{ \text{Vert}, \text{Jaune}, \text{Rouge}, \text{Noir}, \text{Bleu}, \text{Maron} \} \cup$ $\{ \text{Libre} \} \cup$ $\{ \text{Code}(\text{nom}) \text{ tel que } \text{nom} \in \text{string et } (\text{String.length nom})=3 \}$
SÉMANTIQUE	couleur c'est les couleurs des joueurs. Le constructeur Code permet d'entrer les noms de joueur restreint à trois caractères. La couleur Libre est une couleur en plus pour coder l'absence de joueur (dans une case ou pour le gagnant d'une partie).

1.4 case_coloree

SPÉCIFICATION – case colorée	
PROFIL	$\text{case_coloree} \stackrel{\text{def}}{=} \text{case} \times \text{couleur}$
SÉMANTIQUE	case_coloree est un pion d'une couleur col se situe sur une case c est codé par un couple (c, col) .

1.5 configuration

SPÉCIFICATION – configuration du jeu	
PROFIL	$\text{configuration} \stackrel{\text{def}}{=} \text{case_coloree list} \times \text{couleur list} \times \text{dimension}$
SÉMANTIQUE	<p>configuration du jeu est donnée par un triplet formé d'une liste de cases colorées, une liste de joueurs et une dimension. La liste de cases colorées donne l'emplacement des pions et leurs couleurs. On veillera à ce que pour chaque case c il y ait au plus un pion sur cette case, c'est-à-dire il y a au plus une couleur col tel que le couple (c, col) est dans la liste; l'absence de pion sur la case c sera codé par l'absence de couple (c, col) dans la liste et non pas avec (c, Libre). La liste de joueur permet de savoir à qui est le tour (tête de liste) et quel sera le tour des suivants (en suivant l'ordre de la liste). Enfin même si elle ne change pas au cours de la partie la dimension dim est donnée dans la configuration car nous devons pouvoir accéder facilement à celle-ci et pouvoir en changer si nous</p>

souhaitons faire une partie sur un plateau de taille différente.

1.6 coup

SPÉCIFICATION – coups unitaires et multiples	
PROFIL	$\text{coup} \stackrel{\text{def}}{=} \{ Du(c_1, c_2) \text{ tel que } c_1, c_2 \in \text{case} \} \cup \{ Sm(cl) \text{ tel que } cl \in \text{case list} \}$
SÉMANTIQUE	coup existe en de deux sortes: <ul style="list-style-type: none">- Du pour les déplacements unitaires- Sm pour les sauts multiples

1.7 vecteur

SPÉCIFICATION – vecteur	
PROFIL	$\text{vecteur} \stackrel{\text{def}}{=} \text{case}$
SÉMANTIQUE	vecteur est le synonyme de case comme un vecteur permettant des translation avec les même propriétés.

2. Fonctions

2.1 indice_valide

SPÉCIFICATION – indice valide	
PROFIL	$\text{indice_valide} : \mathbb{N} \longrightarrow \text{dimension} \longrightarrow \mathbb{B}$
SÉMANTIQUE	$(\text{indice_valide } x \text{ dim})$ vérifie si la coordonnée x est valide dans la dimension dim .
EX. ET PROP.	<p>1) $\text{indice_valide}(x, \text{dim}) = \text{vrai}$ $\forall x \in \mathbb{Z}, \forall \text{dim} \in \text{dimension}, -2 \text{dim} \leq x \leq 2 \text{dim}$</p> <p>2) $\text{indice_valide}(x, \text{dim}) = \text{faux}$ $\forall x \in \mathbb{Z}, \forall \text{dim} \in \text{dimension}, -2 \text{dim} < x \vee x > 2 \text{dim}$</p>

RÉALISATION – indice valide	
ALGORITHME	par la composition booléenne suivante $-2 \text{dim} \leq x \leq 2 \text{dim}$
IMPLIMENT.	<pre>let indice_valide (x:int) (dim:dimension): bool = -2 * dim <= x && x <= 2 * dim ;;</pre>

2.2 est_case

SPÉCIFICATION – est une case ?	
PROFIL	$\text{est_case} : \text{case} \longrightarrow \mathbb{B}$
SÉMANTIQUE	$(\text{est_case } c)$ vérifie si c est une case.
EX. ET PROP.	<p>1) $\text{est_case}((i, j, k)) = \text{vrai}, \forall (i, j, k) \in \mathbb{Z}^3, i + j + k = 0$</p> <p>2) $\text{est_case}((i, j, k)) = \text{faux}, \forall (i, j, k) \in \mathbb{Z}^3, i + j + k \neq 0$</p>

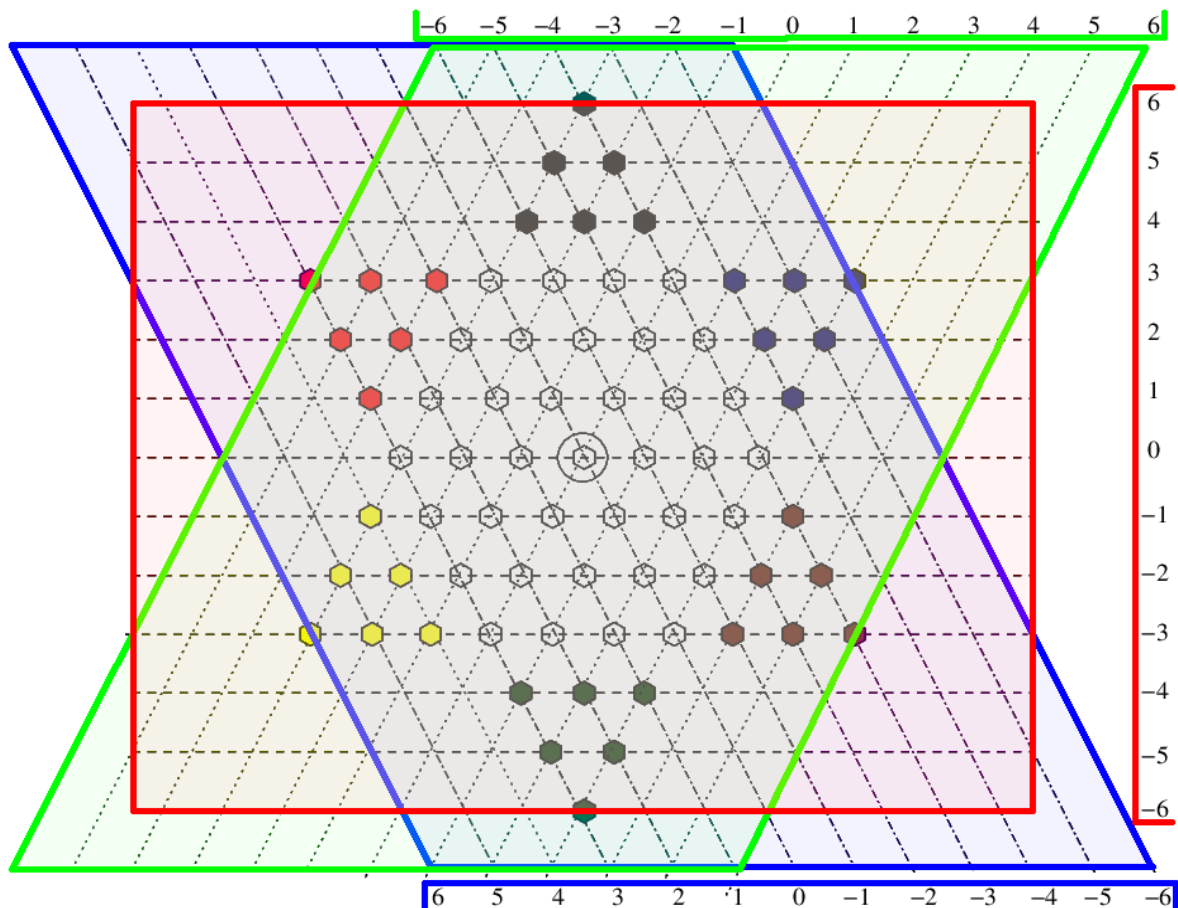


Figure 1. Représentation de la fonction est_case par l'intersection de trois figure ou chacune c'est un des coordonnées

RÉALISATION – est une case ?

ALGORITHME composition booléenne si la somme des coordonnées i, j, k doit être égal à 0.

IMPLIMENT.

```
let est_case (c:case): bool =
  let i, j, k = c in i + j + k = 0
;;
```

2.3 est_dans_losange

SPÉCIFICATION – est dans losange North-South

PROFIL $\text{est_dans_losange} : \text{case} \longrightarrow \text{dimension} \longrightarrow \mathbb{B}$

SÉMANTIQUE ($\text{est_dans_losange } c \text{ dim}$) vérifie si la case c est dans le losange North-South du plateau de dimension dim .

EX. ET PROP.

- 1) $\text{est_dans_losange}((6, -3, -3), 3) = \text{vrai}$
- 2) $\text{est_dans_losange}((0, -3, 3), 3) = \text{vrai}$
- 3) $\text{est_dans_losange}((0, 3, -3), 3) = \text{vrai}$
- 4) $\text{est_dans_losange}((-6, 3, 3), 3) = \text{vrai}$

$\text{est_dans_losange}((0,0,0), \text{dim}) = \text{vrai}, \forall \text{dim} \in \text{dimension}$

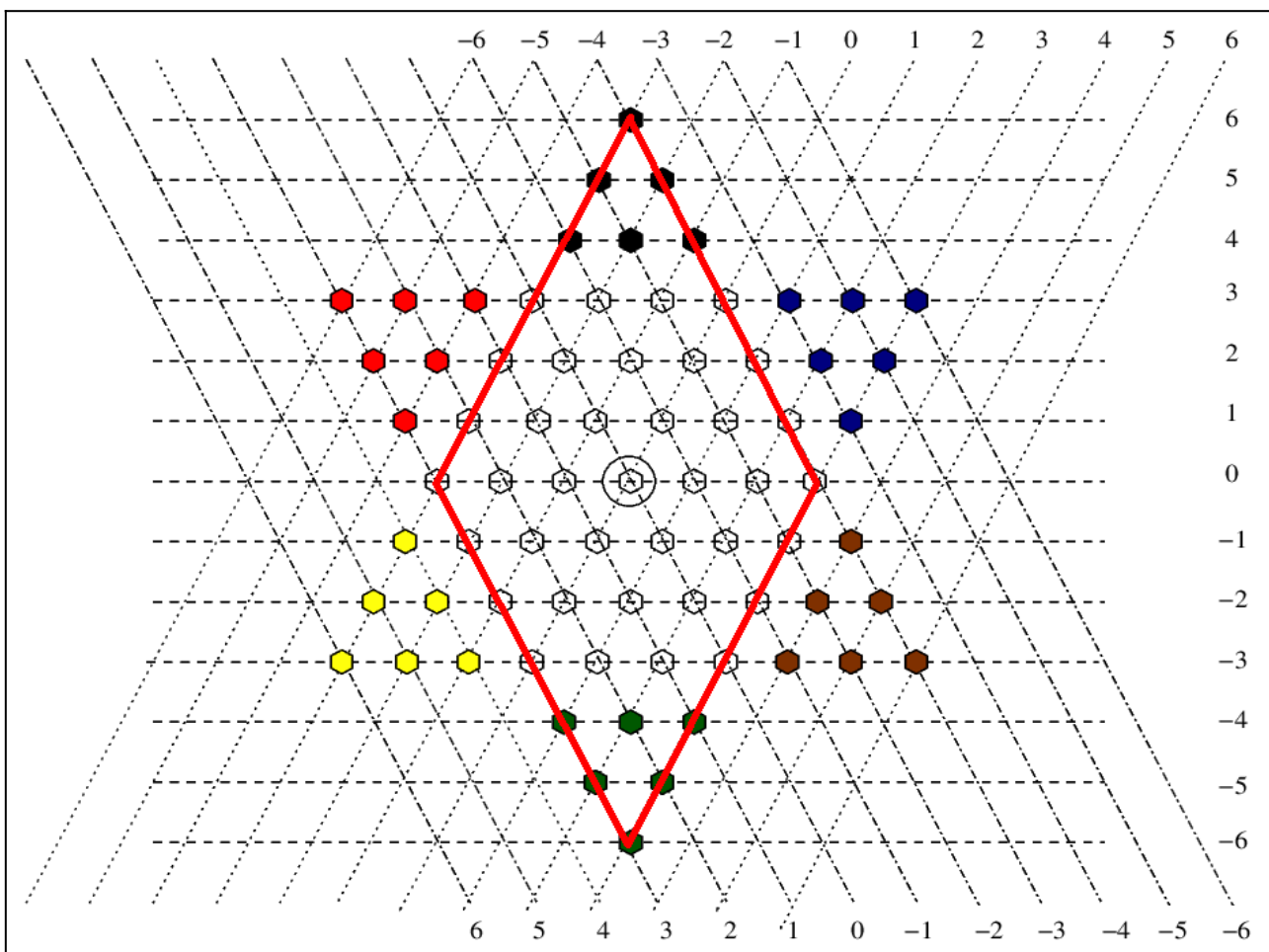


Figure 2. North-South losange

RÉALISATION – est dans losange North-South

ALGORITHME composition booléenne si la case est dans le losange North-South de la figure 2

IMPLIMENT.

```
let est_dans_losange (c:case) (dim:dimension): bool =
  let _, j, k = c in
    -dim <= j && j <= dim &&
    -dim <= k && k <= dim
;;
```

2.4 est_dans_losange_2

SPÉCIFICATION – est dans losange Northwest-Southeast	
PROFIL	$\text{est_dans_losange_2} : \text{case} \rightarrow \text{dimension} \rightarrow \mathbb{B}$
SÉMANTIQUE	$(\text{est_dans_losange_2 } c \text{ } dim)$ vérifie si la case c est dans le losange Northwest-Southeast du plateau de dimension dim .
EX. ET PROP.	<p>1) $\text{est_dans_losange_2}((3, 3, -6), 3) = \text{vrai}$ 2) $\text{est_dans_losange_2}((3, -3, 0), 3) = \text{vrai}$ 3) $\text{est_dans_losange_2}((-3, -3, 6), 3) = \text{vrai}$ 4) $\text{est_dans_losange_2}((-3, 3, 0), 3) = \text{vrai}$</p> <p>$(\text{est_dans_losange_2 } (0,0,0), dim) = \text{vrai}, \forall dim \in \text{dimension}$</p>

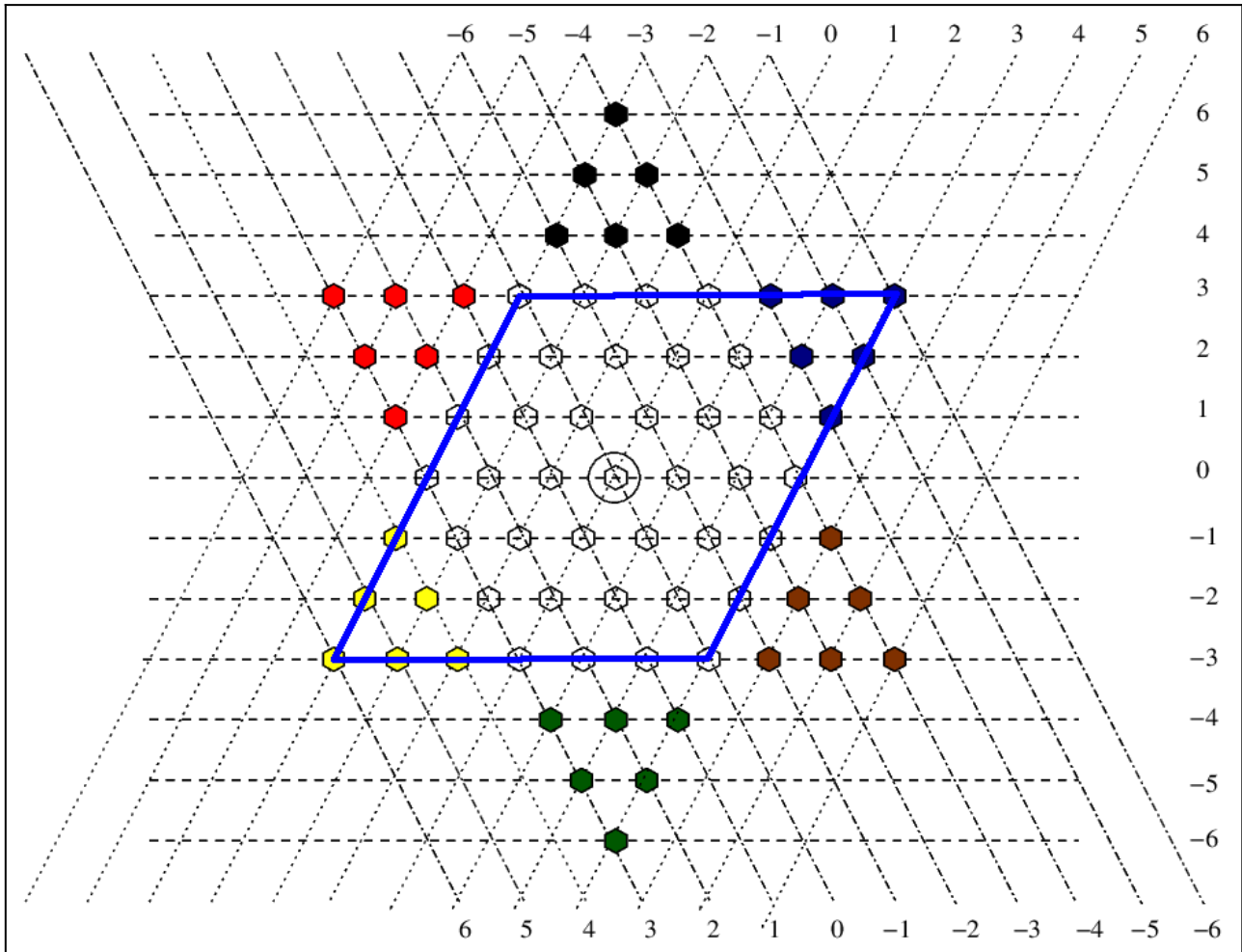


Figure 3. Northwest-Southeast losange

RÉALISATION – est dans losange Northwest-Southeast	
ALGORITHME	composition booléenne si la case est dans le losange Northwest-Southeast de la figure 3
IMPLIMENT.	<pre> let est_dans_losange_2 (c:case) (dim:dimension): bool = let i, _, k = c in -dim <= i && i <= dim && -dim <= k && k <= dim ;; </pre>

2.5 est_dans_losange_3

SPÉCIFICATION – est dans losange Northeast-Southwest	
PROFIL	$\text{est_dans_losange_3} : \text{case} \longrightarrow \text{dimension} \longrightarrow \mathbb{B}$
SÉMANTIQUE	$(\text{est_dans_losange_3})$ vérifie si la case c est dans le losange Northeast-Southwest du plateau de dimension dim .
EX. ET PROP.	<p>1) $\text{est_dans_losange_3}((3, -6, 3), 3) = \text{vrai}$ 2) $\text{est_dans_losange_3}((3, 0, -3), 3) = \text{vrai}$ 3) $\text{est_dans_losange_3}((-3, 6, -3), 3) = \text{vrai}$ 4) $\text{est_dans_losange_3}((-3, 0, 3), 3) = \text{vrai}$</p> <p>$(\text{est_dans_losange_3}(0,0,0), \text{dim}) = \text{vrai}, \forall \text{dim} \in \text{dimension}$</p>

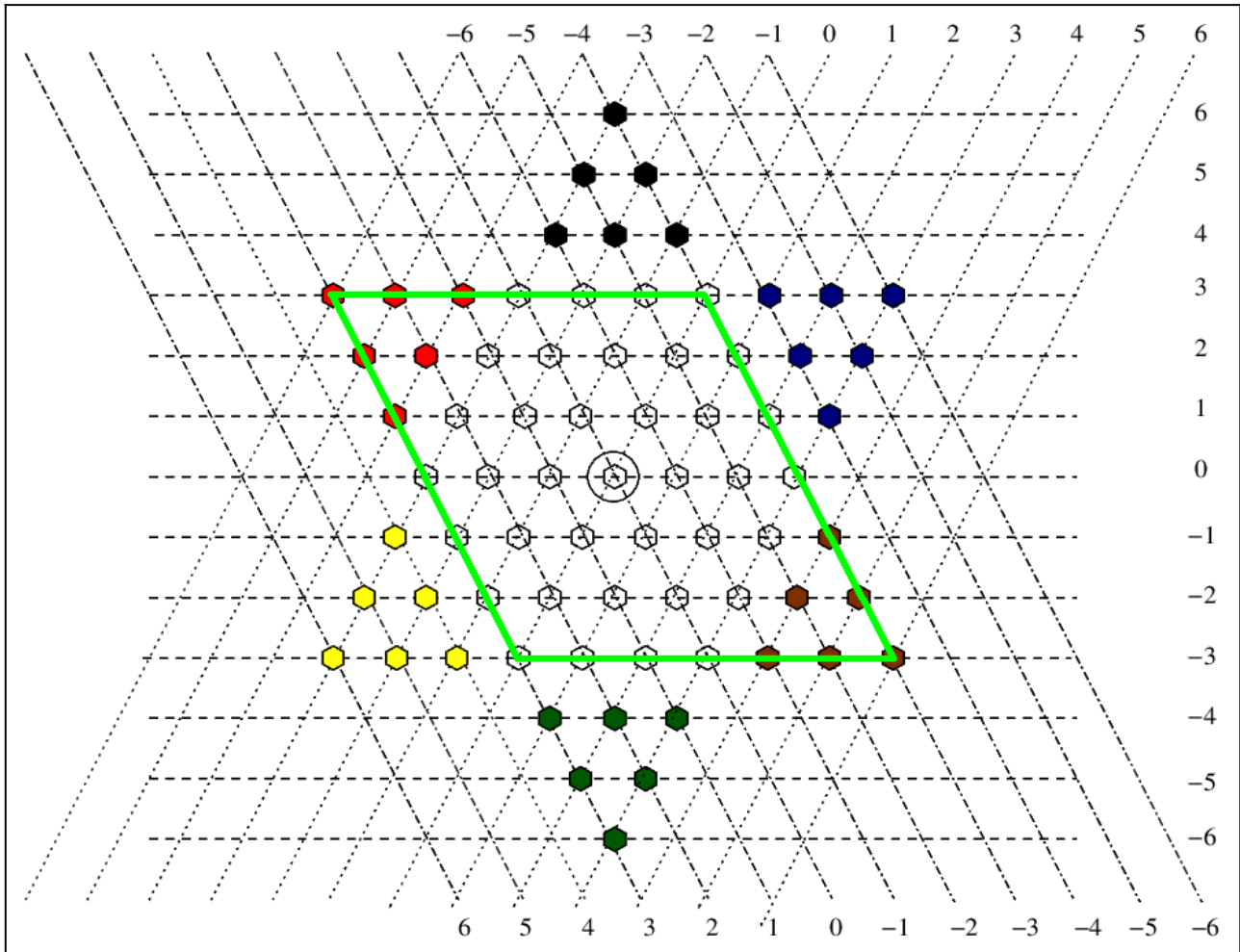


Figure 4. Northeast-Southwest losange

RÉALISATION – est dans losange Northeast-Southwest	
ALGORITHME	composition booléenne si la case est dans le losange Northeast-Southwest de la figure 4
IMPLIMENT.	<pre> let est_dans_losange_3 (c:case) (dim:dimension): bool = let i, j, _ = c in -dim <= i && i <= dim && -dim <= j && j <= dim ;; </pre>

2.6 est_dans_etoile

SPÉCIFICATION – est dans étoile	
PROFIL	$\text{est_dans_etoile} : \text{case} \longrightarrow \text{dimension} \longrightarrow \mathbb{B}$
SÉMANTIQUE	(est_dans_etoile) vérifie si la case c est dans l'étoile du plateau de dimension dim
EX. ET PROP.	<p>1) $\text{est_dans_etoile}((6, -3, -3), 3) = \text{vrai}$ 2) $\text{est_dans_etoile}((3, -6, 3), 3) = \text{vrai}$ 3) $\text{est_dans_etoile}((-3, -3, 6), 3) = \text{vrai}$ 4) $\text{est_dans_etoile}((-6, 3, 3), 3) = \text{vrai}$ 5) $\text{est_dans_etoile}((-3, 6, -3), 3) = \text{vrai}$ 6) $\text{est_dans_etoile}((3, 3, -6), 3) = \text{vrai}$</p> <p>$(\text{est_dans_etoile}(0,0,0), \text{dim}) = \text{vrai}, \forall \text{dim} \in \text{dimension}$</p>

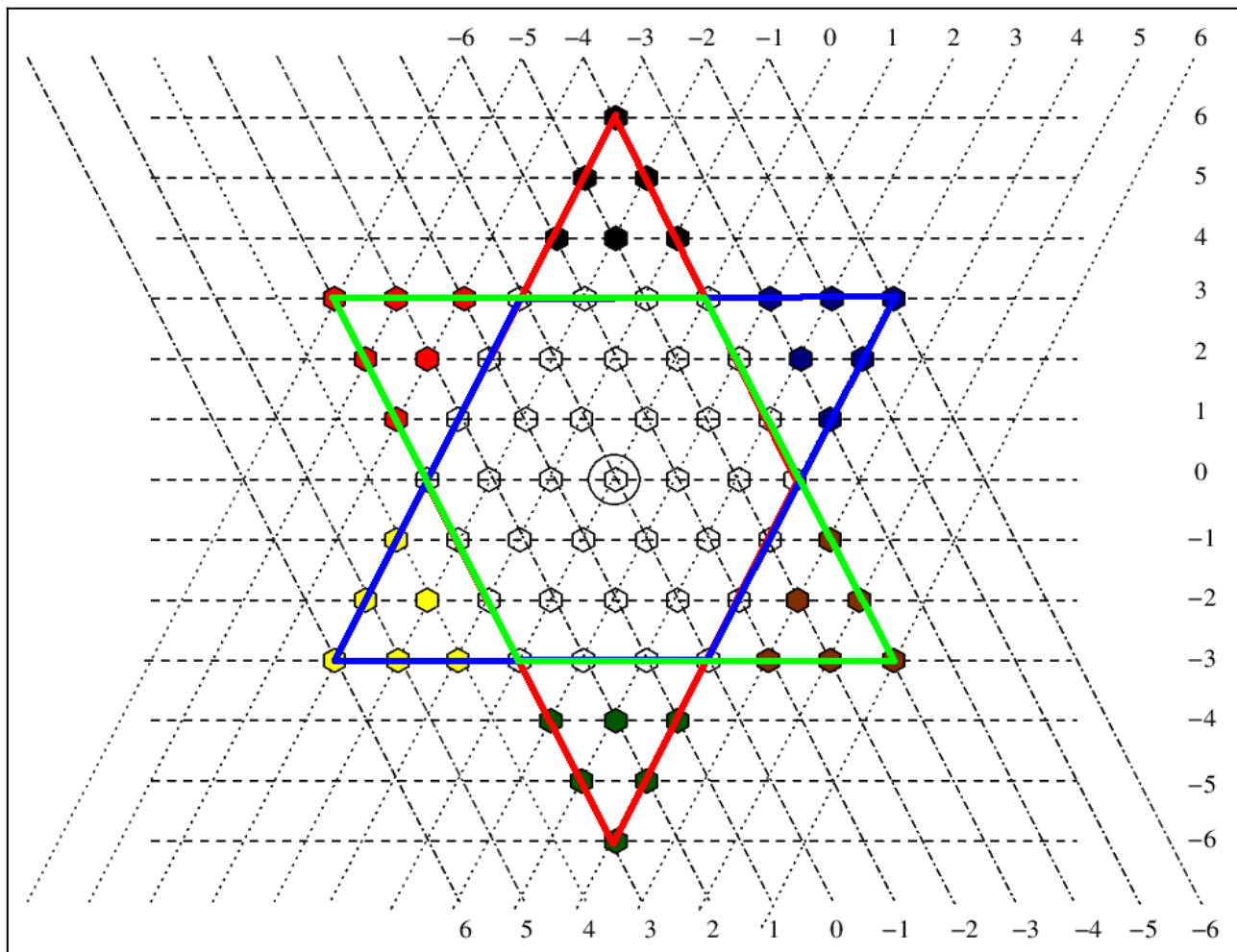


Figure 5. L'union de trois losanges forme une étoile

RÉALISATION – est dans étoile	
ALGORITHME	composition booléenne et fonctions si la case est dans l'union de trois losanges
IMPLIMENT.	<pre> let est_dans_etoile (c:case) (dim:dimension): bool = (* l'union de trois losange est un étoile *) est_dans_losange c dim est_dans_losange_2 c dim est_dans_losange_3 c dim ;; </pre>

2.7 tourner_case

SPÉCIFICATION – tourner case	
PROFIL	$\text{tourner_case} : \mathbb{N} \longrightarrow \text{case} \longrightarrow \text{case}$
SÉMANTIQUE	$(\text{tourner_case } m \ c)$ c'est la case c après avoir fait tourner le plateau de m sixième de tour dans le sens anti-horaire.
EX. ET PROP.	<p>1) $\text{tourner_case}(0, (4, -2, -2)) = (4, -2, -2)$ 2) $\text{tourner_case}(1, (4, -2, -2)) = (2, -4, 2)$ 3) $\text{tourner_case}(2, (4, -2, -2)) = (-2, -2, 4)$ 4) $\text{tourner_case}(3, (4, -2, -2)) = (-4, 2, 2)$ 5) $\text{tourner_case}(4, (4, -2, -2)) = (-2, 4, -2)$ 6) $\text{tourner_case}(5, (4, -2, -2)) = (2, 2, -4)$ 7) $\text{tourner_case}(6, (4, -2, -2)) = (4, -2, -2)$</p> <p>$\forall n \in \mathbb{N}, \forall (i, j, k) \in \text{case}, m = n \bmod 6,$ $\text{sim} = 0, \text{tourner_case}(m, (i, j, k)) = (i, j, k)$ $\text{sim} \neq 0, \text{tourner_case}(m, (i, j, k)) = \text{tourner_case}((m-1), (-k, -i, -j))$</p>

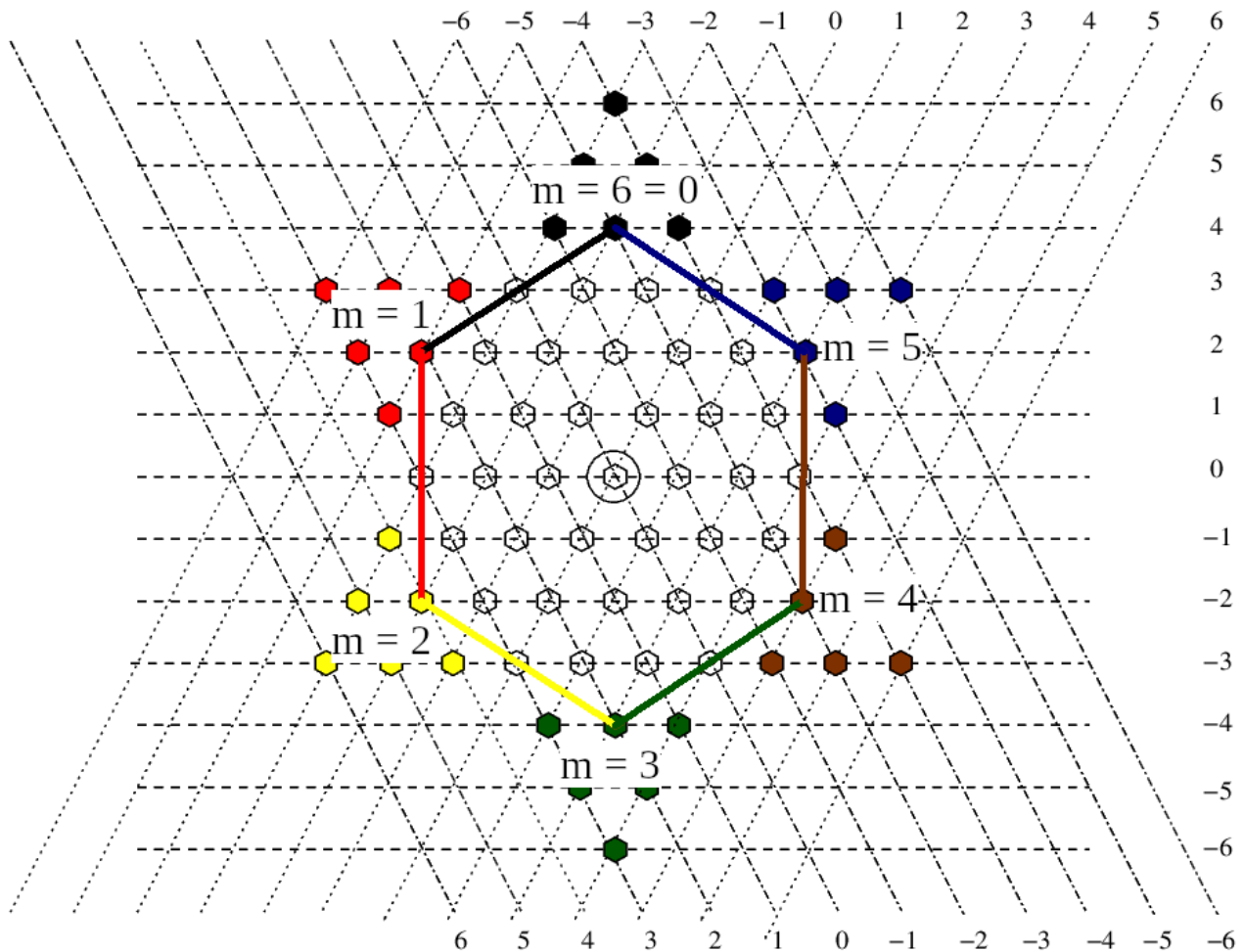


Figure 5. La case après avoir tourner 6 fois

RÉALISATION – tourner case	
On va utiliser une fonction locale <i>tourner</i> avec une équation de récurrence :	
(1) $\text{tourner}(0, (i, j, k)) = (i, j, k)$ (2) $\text{tourner}(m, (i, j, k)) = \text{tourner}((m-1), (-k, -i, -j))$	
ALGORITHME	composition conditionnelle, composition fonction et analyse par cas de filtrage
IMPLIMENT.	<pre>let tourner_case (m:int) (c:case): case = (* réduction de nombre de fait pour tourner *) let m = m mod 6 in</pre>

```

(* équation réursive *)
let rec tourner_case_rec (m:int) (c:case): case =
  let i, j, k = c in
  match m with
  | 0 -> i, j, k
  | m -> tourner_case_rec (m - 1) (-k, -i, -j)
in tourner_case_rec m c
;;

```

2.8 translate

SPÉCIFICATION – translate	
PROFIL	$\text{translate} : \text{case} \longrightarrow \text{vecteur} \longrightarrow \text{case}$
SÉMANTIQUE	$(\text{translate } c \ v)$ calcule la case par le translation du vecteur v à partir de c .
EX. ET PROP.	<p>1) $\text{translate}((0, -2, 2), (0, 1, -1)) = (0, -1, 1)$</p> <p>2) $\text{translate}((0, 0, 0), (0, -2, 2)) = (0, -2, 2)$</p> <p>3) $\text{translate}((0, -2, 2), (0, 0, 0)) = (0, -2, 2)$</p> <p>4) $\text{translate}((0, 0, 0), (0, 0, 0)) = (0, 0, 0)$</p> <p>$\forall (c_1, c_2, c_3), (v_1, v_2, v_3) \in \text{case}, (c_1 + v_1, c_2 + v_2, c_3 + v_3) \in \text{case}$</p>

RÉALISATION – translate	
ALGORITHME	addition de coordonnées de case c et vecteur v
IMPLIMENT.	<pre> let translate (c:case) (v:vecteur): case = (* les coordonnées de la case c *) let c1, c2, c3 = c (* les coordonnées du vecteur v *) and v1, v2, v3 = v in (* translation des coordonnées de v vers c *) v1 + c1, v2 + c2, v3 + c3 ;; </pre>

2.9 diff_case

SPÉCIFICATION – diff_case	
PROFIL	$\text{diff_case} : \text{case} \longrightarrow \text{case} \longrightarrow \text{vecteur}$
SÉMANTIQUE	$(\text{diff_case } c_1 \ c_2)$ c'est le vecteur de translation de c_2 vers c_1 , calculer par la différence entre les cases c_1 et c_2 .
EX. ET PROP.	<p>1) $\text{diff_case}((0, -2, 2), (0, 1, -1)) = (0, -3, 3)$</p> <p>2) $\text{diff_case}((0, 0, 0), (0, -2, 2)) = (0, 2, -2)$</p> <p>3) $\text{diff_case}((0, -2, 2), (0, 0, 0)) = (0, -2, 2)$</p> <p>4) $\text{diff_case}((0, 0, 0), (0, 0, 0)) = (0, 0, 0)$</p> <p>$\forall (i_1, j_1, k_1), (i_2, j_2, k_2) \in \text{case}, (i_1 - i_2, j_1 - j_2, k_1 - k_2) \in \text{vecteur}$</p>

RÉALISATION – diff_case	
ALGORITHME	différence entre les coordonnées des cases c_1 et c_2
IMPLIMENT.	<pre> let diff_case (c1:case) (c2:case): vecteur = let i1, j1, k1 = c1 and i2, j2, k2 = c2 in (* la différence entre les coordonnées c1 et c2 *) i1 - i2, j1 - j2, k1 - k2 ;; </pre>

2.10 sont_cases_alignee

SPÉCIFICATION – sont_cases_alignee	
PROFIL	$\text{sont_cases_alignee} : \text{case} \longrightarrow \text{case} \longrightarrow \mathbb{B}$
SÉMANTIQUE	$(\text{sont_cases_alignee } c1 \ c2)$ vérifie si les cases $c1$ et $c2$ sont alignées.
EX. ET PROP.	<p>1) $\text{sont_cases_alignee}((0, -2, 2), (0, 2, -2)) = \text{vrai}$ 2) $\text{sont_cases_alignee}((-2, 0, 2), (2, 0, -2)) = \text{vrai}$ 3) $\text{sont_cases_alignee}((-2, 2, 0), (2, -2, 0)) = \text{vrai}$</p> <p>4) $\text{sont_cases_alignee}((-2, 2, 0), (2, 0, -2)) = \text{faux}$ pas de coordonnées qui est égal à l'autre $\forall c \in \text{case}, \text{sont_cases_alignee}(c, c) = \text{faux}$ on effet c'est vrai, car chaque de coordonnées est égal, mais on idée les cases doivent être de différentes valeurs, donc c'est faux</p>

RÉALISATION – sont_cases_alignee	
ALGORITHME	analyse par cas de filtrage
IMPLIMENT.	<pre> let sont_cases_alignee (c1:case) (c2:case): bool = let i1, j1, k1 = c1 and i2, j2, k2 = c2 in match () with (* comparasion entre chaque de coordonnées *) (* les doivent être de différentes valeurs *) _ when c1 = c2 -> false (* s'ils sont alignées sur i, donc vrai *) _ when i1 = i2 -> true (* s'ils sont alignées sur j, donc vrai *) _ when j1 = j2 -> true (* s'ils sont alignées sur k, donc vrai *) _ when k1 = k2 -> true _ -> false (* si les cases ne sont pas alignées *) ;; </pre>

2.11 dist_entre_coordonnees

SPÉCIFICATION – dist_coords	
PROFIL	$\text{dist_coords} : \text{case} \longrightarrow \text{case} \longrightarrow \mathbb{N}^3$
SÉMANTIQUE	$(\text{dist_coords } c1 \ c2)$ est un triplet de distances entre les coordonnées des cases $c1$ et $c2$.
EX. ET PROP.,	<p>1) $\text{dist_coords}((0, 0, 0), (0, -2, 2)) = (0, 2, 2)$ 2) $\text{dist_coords}((0, 2, -2), (0, 0, 0)) = (0, 2, 2)$ 3) $\text{dist_coords}((0, 2, -2), (0, -2, 2)) = (0, 4, 4)$</p> <p>$\forall c \in \text{case}, \text{dist_coords}(c, c) = (0, 0, 0)$</p>

RÉALISATION – dist_coords	
ALGORITHME	calcul de chaque coordonnées utilisent la fonction <i>abs</i>
IMPLIMENT.	<pre> let dist_coords (c1:case) (c2:case): int * int * int = let i1, j1, k1 = c1 and i2, j2, k2 = c2 in let di = abs (i1 - i2) (* distance entre les coordonnées i *) and dj = abs (j1 - j2) (* distance entre les coordonnées j *) and dk = abs (k1 - k2) (* distance entre les coordonnées k *) in di, dj, dk (* triplet des distances entres i, j et k *) ;; </pre>

2.12 max_dist_cases

SPÉCIFICATION – max_dist_cases	
PROFIL	$\text{max_dist_cases} : \text{case} \longrightarrow \text{case} \longrightarrow \mathbb{N}$
SÉMANTIQUE	$(\text{max_dist_cases } c1 \ c2)$ est la distance maximale entre les coordonnées des cases $c1$ et $c2$.
EX. ET PROP.	1) $\text{max_dist_cases}((0, 0, 0), (0, -2, 2)) = 2$ 2) $\text{max_dist_cases}((0, 2, -2), (0, 0, 0)) = 2$ 3) $\text{max_dist_cases}((0, 2, -2), (0, -2, 2)) = 4$ $\forall c \in \text{case}, \text{dist_coords}(c, c) = 0$

RÉALISATION – max_dist_cases	
ALGORITHME	utilisation des fonctions <code>dist_coords</code> et <code>max</code>
IMPLIMENT.	<pre>let max_dist_cases (c1:case) (c2:case): int = let di, dj, dk = dist_coords c1 c2 in max di (max dj dk) ;;</pre>

2.13 min_dist_cases

SPÉCIFICATION – min_dist_cases	
PROFIL	$\text{min_dist_cases} : \text{case} \longrightarrow \text{case} \longrightarrow \mathbb{N}$
SÉMANTIQUE	$(\text{min_dist_cases } c1 \ c2)$ est la distance minimale entre les coordonnées des cases $c1$ et $c2$.
EX. ET PROP.	1) $\text{min_dist_cases}((0, 0, 0), (0, -2, 2)) = 0$ 2) $\text{min_dist_cases}((-2, 3, -1), (0, 0, 0)) = 1$ 3) $\text{min_dist_cases}((0, 3, -3), (-2, -2, 4)) = 2$ 3) $\text{min_dist_cases}((0, 3, -3), (-3, -3, 6)) = 3$ $\forall c \in \text{case}, \text{dist_coords}(c, c) = 0$

RÉALISATION – min_dist_cases	
ALGORITHME	utilisation des fonctions <code>dist_coords</code> et <code>min</code>
IMPLIMENT.	<pre>let min_dist_cases (c1:case) (c2:case): int = let di, dj, dk = dist_coords c1 c2 in min di (min dj dk) ;;</pre>

2.14 compte_cases

SPÉCIFICATION – compte_cases	
PROFIL	$\text{compte_cases} : \text{case} \longrightarrow \text{case} \longrightarrow \mathbb{N}$
SÉMANTIQUE	$(\text{compte_cases } c1 \ c2)$ est le nombre de cases entres les cases $c1$ et $c2$. Pour déterminer ce nombre on prendent la distance maximale si ils sont alignées, sinon la distance minimale.
EX. ET PROP.	1) $\text{compte_cases}((0, -1, 1), (0, 1, -1)) = 1$ 2) $\text{compte_cases}((2, -3, 1), (2, 0, -2)) = 2$ 3) $\text{compte_cases}((0, -2, 2), (0, 2, -2)) = 3$ 4) $\text{compte_cases}((3, -3, 0), (-2, 2, 0)) = 4$ 5) $\text{compte_cases}((0, -3, 3), (0, 3, -3)) = 5$ $\forall c \in \text{case}, \text{compte_cases}(c, c) = 0$

RÉALISATION – compte_cases	
ALGORITHME	analyse par cas de filtrage et utilisation des fonctions <i>sont_cases_alignee</i> , <i>max_dist_cases</i> et <i>min_dist_cases</i> .
IMPLIMENT.	<pre> let compte_cases (c1:case) (c2:case): int = match () with (* si les cases sont égal *) _ when c1 = c2 -> 0 (* si les cases sont alignées *) _ when sont_cases_alignee c1 c2 -> max_dist_cases c1 c2 - 1 (* sinon ... *) _ -> min_dist_cases c1 c2 - 1 ;; </pre>

2.15 sont_cases_voisines

SPÉCIFICATION – sont_cases_voisines	
PROFIL	$\text{sont_cases_voisines} : \text{case} \longrightarrow \text{case} \longrightarrow \mathbb{B}$
SÉMANTIQUE	$(\text{sont_cases_voisines } c1 \ c2)$ vérifie si les cases $c1$ et $c2$ sont voisines.
EX. ET PROP.	<p>1) $\text{sont_cases_voisines}((0, 0, 0), (0, -1, 1)) = \text{vrai}$ 2) $\text{sont_cases_voisines}((0, 0, 0), (0, 1, -1)) = \text{vrai}$ 3) $\text{sont_cases_voisines}((0, 0, 0), (-1, 0, 1)) = \text{vrai}$</p> <p>$\forall c \in \text{case}, \text{sont_cases_voisines}(c, c) = \text{faux}$</p>

RÉALISATION – sont_cases_voisines	
ALGORITHME	composition booléenne et utilisation des fonctions <i>sont_cases_alignee</i> et <i>max_dist_cases</i> .
IMPLIMENT.	<pre> let sont_cases_voisines (c1:case) (c2:case): bool = (* si les cases sont alignées et la distances entre eux est 1 *) sont_cases_alignee c1 c2 && max_dist_cases c1 c2 = 1 ;; </pre>

2.16 calcul_pivot

SPÉCIFICATION – calcul_pivot	
PROFIL	$\text{calcul_pivot} : \text{case} \longrightarrow \text{case} \longrightarrow \text{case option}$
SÉMANTIQUE	$(\text{calcul_pivot } c1 \ c2)$ calcul le pivot entre les cases $c1$ et $c2$ si ils sont alignées et le nombre de cases entre les deux est impair, sinon <i>None</i> .
EX. ET PROP.	<p>1) $\text{calcul_pivot}((0, -1, 1), (0, 1, -1)) = \text{Some } (0, 0, 0)$ 2) $\text{calcul_pivot}((1, 0, -1), (-1, 0, 1)) = \text{Some } (0, 0, 0)$ 3) $\text{calcul_pivot}((-1, 1, 0), (1, -1, 0)) = \text{Some } (0, 0, 0)$ 4) $\text{calcul_pivot } (0, -2, 2) \ (2, 0, -2) = \text{None}$ 5) $\text{calcul_pivot } (2, 0, -2) \ (-2, 2, 0) = \text{None}$ 6) $\text{calcul_pivot } (-2, 2, 0) \ (0, -2, 2) = \text{None}$</p> <p>$\forall c \in \text{case}, \text{calcul_pivot}(c, c) = \text{None}$</p>

RÉALISATION – calcul_pivot	
ALGORITHME	composition conditionnelle et utilisation de fonctions <i>mod</i> , <i>compte_cases</i> , <i>translate</i> et <i>sont_cases_alignee</i> .
IMPLIMENT.	<pre> let calcul_pivot (c1:case) (c2:case): case option = (* si le nombre de cases entre c1 et c2 est impair *) let est_impair = (compte_cases c1 c2) mod 2 = 1 (* les coordonnées du vecteur de translation de c2 vers c1 *) and i, j, k = diff_case c1 c2 in (* le vecteur de translation de c2 vers le mi-chemin de c1 *) </pre>

```

let v = i/2, j/2, k/2 in
(* les coordonnées de pivot *)
let p = translate c2 v in
if est_impair && sont_cases_alignee c1 c2
then Some(p) (* si impair et alignées, pivot existe *)
else None    (* sinon, pivot n'existe pas *)
;;

```

2.17 vec_et_dist

SPÉCIFICATION – vec_et_dist	
PROFIL	$\text{vec_et_dist} : \text{case} \longrightarrow \text{case} \longrightarrow \text{vecteur} \times \mathbb{N}$
SÉMANTIQUE	$(\text{vec_et_dist } c1 \ c2)$ est le couple (v, d) avec v le vecteur de translation d'un déplacement unitaire des cases alignées $c1$ vers $c2$ et avec d la distance entre $c1$ et $c2$. Si le vecteur unitaire n'existe pas, alors on renvoie $((0, 0, 0), 0)$.
EX. ET PROP.	1) $\text{vec_et_dist}((0, -2, 2), (0, 0, 0)) = ((0, 1, -1), 2)$ 2) $\text{vec_et_dist}((0, 0, 0), (0, -2, 2)) = ((0, -1, 1), 2)$ 3) $\text{vec_et_dist}((0, -2, 2), (0, -2, 2)) = (0, 0, 0), 0$ 4) $\text{vec_et_dist}((0, -2, 2), (-2, 2, 0)) = (0, 0, 0), 0$ $\text{vec_et_dist}(c, c) = ((0, 0, 0), 0), \forall c \in \text{case}$

RÉALISATION – vec_et_dist	
ALGORITHME	composition conditionnelle et utilisation des fonctions <i>sont_cases_alignee</i> , <i>max_dist_cases</i> et <i>diff_case</i> .
IMPLIMENT.	<pre> let vec_et_dist (c1:case) (c2:case): vecteur * int = (* si c1 = c2 ou non alignées renvoie nuls *) if c1 = c2 not (sont_cases_alignee c1 c2) then (0, 0, 0), 0 else (* sinon ... *) (* la distance entre les cases *) let d = max_dist_cases c1 c2 (* les coordonnées du vecteur de translation de c2 vers c1 *) and i, j, k = diff_case c1 c2 in (* les coordonnées du vecteur de translation unitaire de *) (* c2 vers c1 *) let i, j, k = i/d, j/d, k/d in (* le vecteur de translation unitaire de c1 vers c2 *) let v = i * (-1), j * (-1), k * (-1) in if est_case v then v, d (* si c'est un vecteur *) else (0, 0, 0), 0 (* sinon *) ;; </pre>