L3 Informatique : Méthodes formelles d'aide à la détection d'erreur
Problème du robot : 1ère partie

Le problème du robot est le suivant :

Une table comporte N creux dans chacun desquels se trouve une pierre. Chaque pierre est de couleur uniforme, bleu, blanc ou rouge. Il s'agit de placer ces pierres de telle manière à ce qu'on obtienne, de gauche à droite, toutes les pierres bleues puis toutes les pierres blanches et enfin toutes les pierres rouges. Pour réaliser cette tâche, on dispose d'un robot qui a les capacités suivantes :

- il peut compter et faire des comparaisons dans $\mathbb{N}$,
- il peut reconnaître la couleur d'une pierre située dans un creux grâce à la fonction « LireCouleur(i) » de précondition : i est un entier compris entre 1 et N, et de postcondition : LireCouleur(i) = la couleur de la pierre située au creux d'indice i,
- il peut comparer deux couleurs,
- il peut échanger les pierres situées dans deux creux différents grâce à la fonction « Echanger(i,j) » de précondition : i et j sont deux entiers différents compris entre 1 et N, et de postcondition : les pierres situées au creux d'indices i et j ont été échangées,
- il peut effectuer une instruction conditionnelle du type « Si…alors…sinon »,
- il peut effectuer une instruction répétitive du type « Tant que… ».

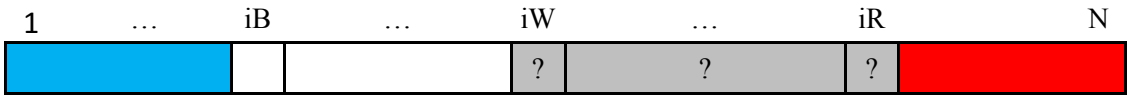
Déterminer l'algorithme que devra exécuter le robot pour réaliser cette tâche en exécutant un nombre d'actions linéaire par rapport à N.

Le problème du robot peut être vu comme un problème classique de tri avec B < W < R, où B, W et R représentent respectivement les couleurs bleu, blanc (white) et rouge. Mais si l'on ajoute l'exigence d'un nombre d'actions linéaire par rapport au nombre de cases, alors il faut réfléchir à une solution autre que celles généralement abordées pour résoudre le problème de tri dans les années précédentes.

Attention, comme il s'agit d'une première approche, la présentation des étapes et leur ordre d'apparition se fera en fonction de leur difficulté « technique », elles diffèreront donc quelque peu de ce qui a été présenté en introduction. En particulier, dans ces notes ne figure pas l'étape 1 car le problème est simple à comprendre et l'étape 2 est reportée après les étapes 3 et 4. Les étapes 5 et 6 constituent la seconde partie de ces quelques notes sur le problème du robot et sera traitée en TD.

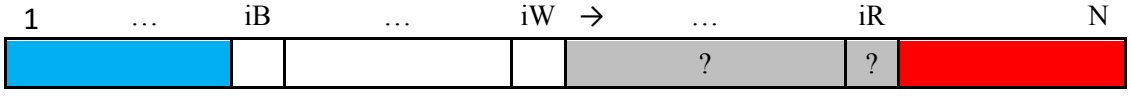
IDÉE & JUSTIFICATION

On peut représenter la table par un tableau à N cases et on utilise 3 indices : iB, iW et iR, tels que toutes les cases strictement à gauche de iB sont bleues (propriété P_B), celles à partir de iB jusqu'à iW exclu sont blanches (propriété P_W) et celles strictement à droite de iR sont rouges (propriété P_R). On ne connaît pas la couleur des cases grises. Voir le tableau ci-dessous :

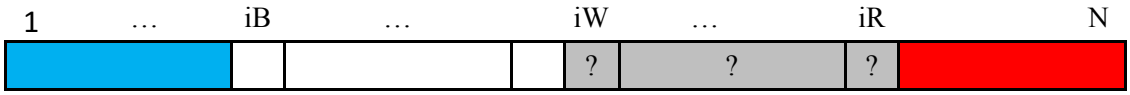


Il s'agit alors de faire évoluer ces 3 indices en fonction de la couleur trouvée à l'indice iW tout en conservant ces 3 propriétés après évolution. On distingue les trois cas possibles (du plus simple au plus compliqué) :

1. **La couleur en iW est W :** il suffit alors d'incrémenter iW :

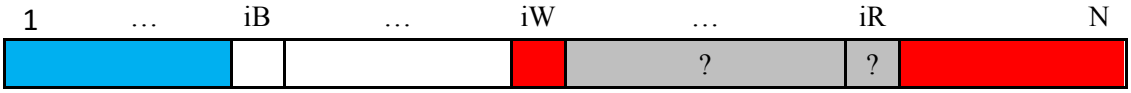


Pour obtenir :

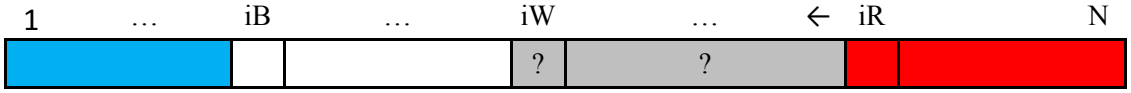


Et les trois propriétés sont maintenues.

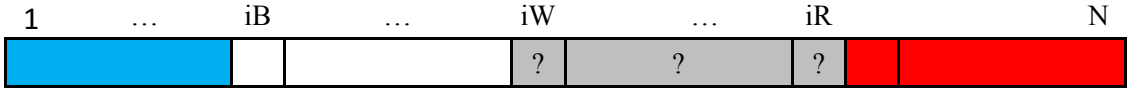
2. La couleur en iW est R :



Il faut alors échanger les pierres en iW et iR pour obtenir :

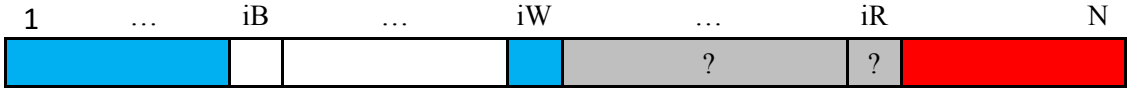


Ce qui donne en décrémentant iR :

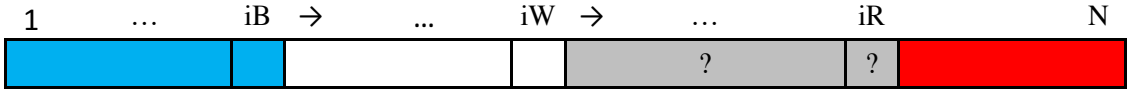


Et les trois propriétés sont maintenues.

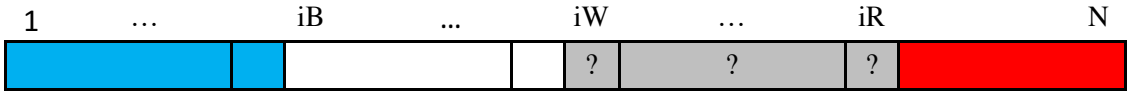
3. La couleur en iW est B :



Il faut alors échanger les pierres en iW et iB pour obtenir :



Cette fois-ci non seulement la couleur en iB est B, mais celle en iW est W, on incrémente alors iB et iW pour obtenir :



Et les trois propriétés sont maintenues.

Le principe général du traitement étant décrit, ce traitement devant être répété jusqu'à obtention du résultat souhaité, il faut regarder les conditions initiales et finales de ce traitement.

- Valeurs initiales de iB, iW et iR :
  Au départ, aucune case n'a été explorée et il semble logique d'initialiser iW à 1 de même que iB puisqu'étant donné le traitement, il est clair que l'on aura toujours iB ≤ iW. Enfin on initialise iR à N (contrairement à iB et iW qui ne peuvent être qu'incrémentés, iR ne peut être que décrémenté). Selon l'idée qu'une propriété sur les éléments d'un ensemble est vérifiée par défaut sur un ensemble vide on peut donc conjecturer que les propriétés P_W, P_B et P_R sont vraies avec cette initialisation (voir la démonstration formelle ci-dessous).
- Valeurs d'arrêt du traitement :
  Quand le traitement doit-il s'arrêter ? Évidemment quand toutes les pierres sont bien placées. À quoi cela correspond-il pour les indices iB, iW et iR ? Seules les pierres des cases d'indices compris entre iW et iR (iW et iR inclus) sont de couleur inconnue. On peut donc s'arrêter lorsque l'ensemble de ces indices est vide, c'est-à-dire lorsque iW > iR.

Pour démontrer formellement la validité de l'initialisation, on pose tout d'abord de façon formelle les prédicats P_b, P_w et P_R :

- $P_B : \forall i, (\ i \in \mathbb{N} \cap [1, iB[) \rightarrow (\text{LireCouleur}(i) = B)$
- $P_W : \forall i, (\ i \in \mathbb{N} \cap [iB, iW[) \rightarrow (\text{LireCouleur}(i) = W)$
- $P_R : \forall i, (\ i \in \mathbb{N} \cap]iR, N]) \rightarrow (\text{LireCouleur}(i) = R)$

Ces trois prédicats sont bien de la forme générique : $a \rightarrow b$. Or, lorsque a est faux, $a \rightarrow b$ est vrai. L'initialisation entraîne les propriétés : $[1, iB[= [iB, iW[= [1,1[= \emptyset$ et $]iR, N] =]N, N] = \emptyset$, donc les trois implications de P_b, P_w et P_R sont vraies puisque leur prémisse (qui se réduit à $i \in \emptyset$) est fausse (aucun élément ne peut appartenir à l'ensemble vide).

ALGORITHME

On peut maintenant écrire l'algorithme correspondant. Il faut cependant porter une attention particulière aux préconditions des fonctions LireCouleur() et Echanger() : vérifier que les indices utilisés sont bien compris entre 1 et N et de plus que les deux indices de la fonction Echanger() sont bien distincts. Si la première condition ne semble pas poser de problème particulier : à l'intérieur de la boucle on a en effet $1 \leq iB \leq iW \leq iR \leq N$, inéquations qu'il faudra établir de façon formelle à l'étape 5, il faut en revanche faire attention à ne pas modifier structurellement l'idée établie à l'étape 3 : il s'agit simplement de rendre « virtuel » l'échange des deux pierres, lorsque les deux indices coïncident, en ne faisant rien au niveau de l'algorithme (voir l'appel à Echanger(i,j) conditionné par i ≠ j dans l'algorithme ci-dessous).

On rappelle la donnée du problème : une table avec N creux contenant chacun une pierre de couleur uniforme bleu (B), blanc (W) ou rouge (R).

```
VARIABLES
  entier iB, iW, iR ;
DÉBUT
  iB ← 1 ; iW ← 1 ; iR ← N ;
  TANT QUE (iW ≤ iR) FAIRE
        SI (LireCouleur(iW) = W) ALORS
              iW ← iW + 1 ;
        SINON
              SI (LireCouleur(iW) = R) ALORS
                    SI (iW ≠ iR) ALORS
                          Echanger(iW, iR) ;
                    FINSI
                    iR ← iR – 1 ;
               SINON // LireCouleur(iW) = B
                    SI (iW ≠ iB) ALORS
                          Echanger(iW, iB) ;
                    FINSI
                    iB ← iB + 1 ; iW ← iW + 1 ;
              FinSI
        FinSI
  FinTantQue
FIN
```

SPÉCIFICATION

On propose ici deux variantes en fonction du fait que l'on exige (1) ou pas (2) que les trois couleurs apparaissent effectivement sur la table.

Notations : On appelle T la table avec la disposition initiale des pierres et T' la table avec la disposition finale. On pose $C = \{B, W, R\}$ et pour $X = C$ ou X sous ensemble de C on notera $X^* = \bigcup_{i \geq 0} X^i$ et $X^+ = \bigcup_{i \geq 1} X^i$ où $X^i = \underbrace{X \times X \times \ldots \times X}_{i\ fois}$. Enfin, pour ne pas surcharger l'écriture, si Yest un élément de C, on notera Y^* l'ensemble $\{Y\}^*$ et Y^+ l'ensemble $\{Y\}^+$.

1. Les trois couleurs apparaissent dans la donnée.
   - Précondition :
     $N \in \mathbb{N} \cap [3, +\infty[$,
     $T \in C^N \cap C^*BC^* \cap C^*WC^* \cap C^*RC^*$
   - Postcondition :
     $T' \in C^N \cap B^+W^+R^+$,
     $(|T'|_B = |T|_B) \wedge (|T'|_W = |T|_W) \wedge (|T'|_R = |T|_R)$

   Remarque : on peut formuler la postcondition en utilisant la notion de permutation plutôt que d'écrire les égalités sur les occurrences des couleurs :
   Variante de la postcondition :
   $T' \in C^N \cap B^+W^+R^+$,
   $T' = Perm(T)$ où $Perm(T)$ est une permutation des éléments de T.

2. Il peut y avoir moins de trois couleurs dans la donnée.
   Dans ce cas, il est même possible d'envisager que la table ne comporte aucun creux…
   - Précondition :
     $N \in \mathbb{N}$,
     $T \in C^N$
   - Postcondition :
     $T' \in C^N \cap B^*W^*R^*$,
     $(|T'|_B = |T|_B) \wedge (|T'|_W = |T|_W) \wedge (|T'|_R = |T|_R)$

   On peut aussi appliquer ici la remarque faite au 1.