Problème du drapeau hollandais

Le problème du drapeau : La table

- Un table est subdivisée en N cases numérotés de 1 à N

- Chacune des cases contient une pierre et une seule

- Chaque pierre a une couleur (unique et immuable) parmi (Bleu, Blanc ou Rouge)

Le problème du drapeau : Le robot

- Un robot (programmable) manipulateur de pierre peut effectuer les deux opérations sur les pierres :

  - LireCouleur(i) : quand i est un entier naturel compris entre 1 et N, le robot Prend la pierre dans la case i de la table lis sa couleur et la renvoie après avoir reposé la pierre dans la même case i

  - Echange(i,j) : quand i et j sont deux entiers naturels distincts compris entre 1 et N, prend les deux pierre contenues dans les deux cases i et j et échange leurs places. Si i = j le robot ne fait rien.

Le problème du drapeau : Le but

- A l’aide du robot, il faut réordonner les pierres de la table de telle sorte que :

  - Les pierres Bleues apparaissent en premier
  - Les pierres Rouges apparaissent à la fin
  - Les pierres Blanches apparaissent au milieu

Comprendre le problème

- Vous pouvez répéter la question ?

- Ecrire le programme du robot pour réorganiser les pierres afin de former un drapeau hollandais (Dijkstra, 1976)

Spécifications

- Nous appellerons TI la configuration initiale de la table et TF la configuration finale de la table de N cases

- Hypothèse : Le nombre de pierres rouges sur TI plus le nombre de pierres blanche sur TI plus le nombre de pierres bleues sur TI est égal à N

- P1 : Le nombre de pierres rouges sur TI est égal au nombre de pierres rouges sur TF

- P2 : Le nombre de pierres blanches sur TI est égal au nombre de pierres blanches sur TF

- P3 : Le nombre de pierres bleues sur TI est égal au nombre de pierres bleues sur TF

Spécifications

- Ordre :

  - P4 : Soit i et j deux entiers naturels tels que compris entre 1 et N, dans TF Couleur(i) = Bleu et Couleur(j) ≠ Bleu => i < j (bleues en 1er)

  - P5 : Soit i et j deux entiers naturels tels que compris entre 1 et N, dans TF Couleur(i)=Rouge et Couleur(j) ≠ Rouge => i > j (rouges en fin)

Idées et justifications

- Mettre les pierres bleues au début :

  - Supposons qu'il existe un indice entier IB tel que :
    - PBl (IB) : Pour tout i, $0 < i < IB$, Couleur(i) = Bleu

  - Supposons qu'il existe un indice entier IC tel que :
    - PNBl(IB,IC) : Pour tout j, $IB \leq j < IC$, Couleur(j) ≠ Bleu

  - Lisons la couleur de la pierre à l'indice IC.

Idées et justifications

1		IB-1	IB		IC-1	IC			N
	…			…		????	…	????	?????

Deux cas :

- Couleur(IC) = Bleu
- Couleur(IC) ≠ Bleu

Idées et justifications

- Couleur(IC) = Bleu : On échange les pierres d’indice IB et IC

Idées et justifications

1		**IB-1**	**IB**		**IC-1**	**IC**			**N**
	…			…			????	…	????

Après échange

1		IB-1	IB		IC-1	IC			N
				…			????	…	????

Après l'échange :

- la propriété PBl est vraie jusqu'au rang IB+1 : PBl(IB+1)
- La propriété PNBl est vrai jusqu'au rang IC+1 : PNBl(IB+1,IC+1)

Idées et justifications

- Couleur(IC) ≠ Bleu : pas d’échange

Idées et justifications

1		**IB-1**	**IB**		**IC-1**	**IC**			**N**
	…			…			????	…	????

1		IB-1	IB		IC-1	IC			N
				…			????	…	????

Pas d’échange :

- la propriété PBl est toujours jusqu’au rang IB PBl(IB)
- La propriété PNBl est vrai jusqu’au rang IC+1 PNBl(IB,IC+1)

Idées et justifications

- Quand s’arrêter ? Quand IC > N (on est sorti du tableau)

- Comment initialiser ? IB = 1 et IC = 1 sont des valeurs correctes
  - PBl(1) et PNBl(1,1) sont trivialement vrais : Pour tout $x \in \emptyset$, … est toujours vrai

L'algorithme Bleu au début

Variable IB, IC : Entiers Naturels

DébutCode

IB ← 1; IC ← 1

Tant que IC ≤ N faire

Si LireCouleur(IC) = Bleu alors Echange(IB,IC); IB ← IB+1; IC ← IC+1

Sinon IC ← IC+1

FinSi

FinTantQue

FinCode

L’algorithme Bleu au début

```
Variable IB, IC : Entiers Naturels
DébutCode
    IB ←1; IC ←1
    Tant que IC ≤ N faire /*a*/
        Si LireCouleur(IC) = Bleu alors Echange(IB,IC) /*b*/; IB ←IB+1 /*c*/; IC←IC+1 /*d*/
        Sinon /*e1*/ IC ← IC+1 /*e2*/
        FinSi /*f*/
    FinTantQue
FinCode
```

Preuve partie Si

- En /*a*/, après i tours (complets) de boucle, les prédicats PBl(IB) et PNBl(IB,IC) sont vrais, de plus IC = i+1
- En /*b*/ (après i tours (complets) de boucle) les prédicats PBl(IB+1) et PNBl(IB+1,IC+1) sont vrais, de plus IC = i+1
- En /*c*/ (après i tours (complets) de boucle) les prédicats PBl(IB) et PNBl(IB,IC+1) sont vrais, de plus IC = i+1
- En /*d*/ (après i tours (complets) de boucle) les prédicats PBl(IB) et PNBl(IB,IC) sont vrais, de plus IC = i+2
- En /*f*/ après être passé par le si, on a fait i+1 tours (complets) de boucle (et donc en /*a*/ après i+1 tours complets), les prédicats PBl(IB) et PNBl(IB,IC) sont vrais, de plus IC = (i+1)+1

Preuve partie Sinon

- En /*a*/ après i tours (complets) de boucle les prédicats PBl(IB) et PNBl(IB,IC) sont vrais, de plus IC = i+1

- En /*e1*/ (après i tours (complets) de boucle) les prédicats PBl(IB) et PNBl(IB,IC+1) sont vrais, de plus IC = i+1

- En /*e2*/ (après i tours (complets) de boucle) les prédicats PBl(IB) et PNBl(IB,IC) sont vrais, de plus IC = i+2

- En /*f*/ après être passé par le sinon, on a fait i+1 tours (complets) de boucle (et donc en /*a*/ après i+1 tours complets), les prédicats PBl(IB) et PNBl(IB,IC) sont vrais, de plus IC = (i+1)+1

Preuve Remarque

- La propriété en /*f*/ et en /*a*/ est indépendante de l'exécution du Si ou du Sinon

- Après N tours de boucle :
  - Uniquement des échanges : P1, P2 et P3 Ok

  - IC = N+1
  - PBl (IB) : Pour tout i, $0 < i < IB$, Couleur(i) = Bleu
  - PNBl(IB,N+1) : Pour tout j, $IB \leq j < N+1$, Couleur(j) ≠ Bleu
  - Donc P4 (mais pas P5, mais c'était pas le but !)

Mettre les pierres rouges à la fin

- On souhaite de la même façon mettre les pierres rouges aux indices de valeurs les plus hautes sur notre table.

- Nous allons définir un indice IR tel que pour toute valeur v vérifiant IR<v<N+1, LireCouleur(v) = Rouge

- Nous allons définir un indice IC tel que pour toute valeur v’ vérifiant 0<v’<IC, LireCouleur(v’) ≠ Rouge

- On lit la pierre d’indice IC

1		IC-1	IC	…	…	IR	IR+1		N
			???	???	???	???			
Cas 1									
Cas 2									

- Cas 1 : on a pierre rouge, on échange les pierres d’indice IR et IC
  - On a des pierres rouges à partir de l’indice IR (inclus)
  - On n’a pas de pierres rouges entre les indices 0 et IC (bornes exclue)
  - On veut lire à la prochaine étape la couleur de la pierre à l’indice IC

- Cas 2 : la pierre n’est pas rouge
  - On a des pierres rouges à partir de l’indice IR (exclus)
  - On n’a pas de pierres rouges entre les indices 1 et IC (bornes incluses)
  - On veut lire à la prochaine étape la couleur de la pierre d’indice IC+1

L’algorithme deux

```
Variable IR, IC : Entiers Naturels
DébutCode
    IR ←N; IC ←1
    Tant que IC < IR faire
        Si LireCouleur(IC) = Rouge alors Echange(IR,IC); IR ←IR-1;
        Sinon IC ← IC+1
        FinSi
    FinTantQue
FinCode
```

Preuve : Partie 1 L’algorithme se termine

- Le problème est qu’il faut reconstruire tout le raisonnement à partir du code de l’algorithme.
- Commençons par montrer que l’on sort de la boucle. Observons la valeur de l’expression IR-IC+i+1 où i désigne le nombre de tours de boucle totalement effectué. On remarque que IR-IC+i+1 = cste = N (après i tours de boucle complet)
- IR-IC décroît d’exactement 1 à chaque tour de boucle
- Conséquence la boucle s’effectue N-1 fois (IR-IC finit par être égal à 0 et ainsi on a i=N-1 dans l’équation précédente)
- Le programme s’arrête, donc il produit un résultat. Lequel ?

Preuve : Partie 1 L'algorithme se termine

- Commençons par montrer que l'on sort de la boucle. Observons la valeur de l'expression IR-IC+i+1 où i désigne le nombre de tours de boucle totalement effectué. On remarque que IR-IC+i+1 = cste = N

  - Initialement (0 tour de boucle) cette propriété est vrai car IR=N, IC=1 et i=0 donc IR-IC+i+1 = N-1+0+1 = N

  - Admettons que cette propriété soit vraie après k tours de boucle et supposons que notre programme effectue un k+1 $i^{ème}$ tour de boucle

Preuve : Partie 1 L'algorithme se termine

- Supposons IR-IC+k+1 = N après k tours (complets) de boucle
- Soit AIR et AIC les valeurs des variables IR et IC après k tours de boucle
- Donc AIR-AIC+k+1 = N

- Nous devons distinguer deux cas dans l'exécution de notre algorithme en fonction du résultat de LireCouleur(IC) :

  - Cas 1 : Lirecouleur(IC) = Rouge. À la fin de notre boucle, IR = AIR - 1 et IC = AIC.
    Calculons IR-IC+(k+1)+1 à la fin de notre k+1 $i^{ème}$ tour de boucle
    IR-IC+(k+1)+1 = AIR-1-AIC+k+1+1 = AIR-AIC+k+1 = N

  - Cas 2 : A faire

Preuve : Partie 2 Rouge à la fin

- Montrons qu’il n’y a que des pierres rouges après l’indice IR et écrivons tout d’abord le prédicat
  - PR(AllRed) : Pour tout j, AllRed < j ≤ N, Couleur(j) = Rouge

- Montrons qu’il n’y a pas des pierres rouges avant l’indice IC et écrivons tout d’abord le prédicat
  - PNR(NoRed) : Pour tout j, 0 < j < NoRed, Couleur(j) ≠ Rouge

- Remarque : PR(N) et PNR(1) sont trivialement vrais

L’algorithme deux : Rouge à la fin

```
Variable IR, IC : Entiers Naturels
DébutCode
    IR ←N; IC ←1
    Tant que IC < IR faire /* PR(IR) et PNR(IC) */
        Si LireCouleur(IC) = Rouge alors Echange(IR,IC); IR ←IR-1;
        Sinon IC ← IC+1
        FinSi
    FinTantQue
FinCode
```

Preuve Remarque

- La propriété PR(IR) et PNR(IC) est indépendante de l'exécution du Si ou du Sinon

- Terminaison après N-1 tours de boucle
- Après N-1 tours de boucle :
  - Uniquement des échanges : P1, P2 et P3 Ok
  - PR(IR) et PNR(IC) donc P5

L’algorithme Drapeau Holandais

DébutCode

Bleu au début;

Rouge à la fin

FinCode

Terminaison après 2N-1 tours de boucle

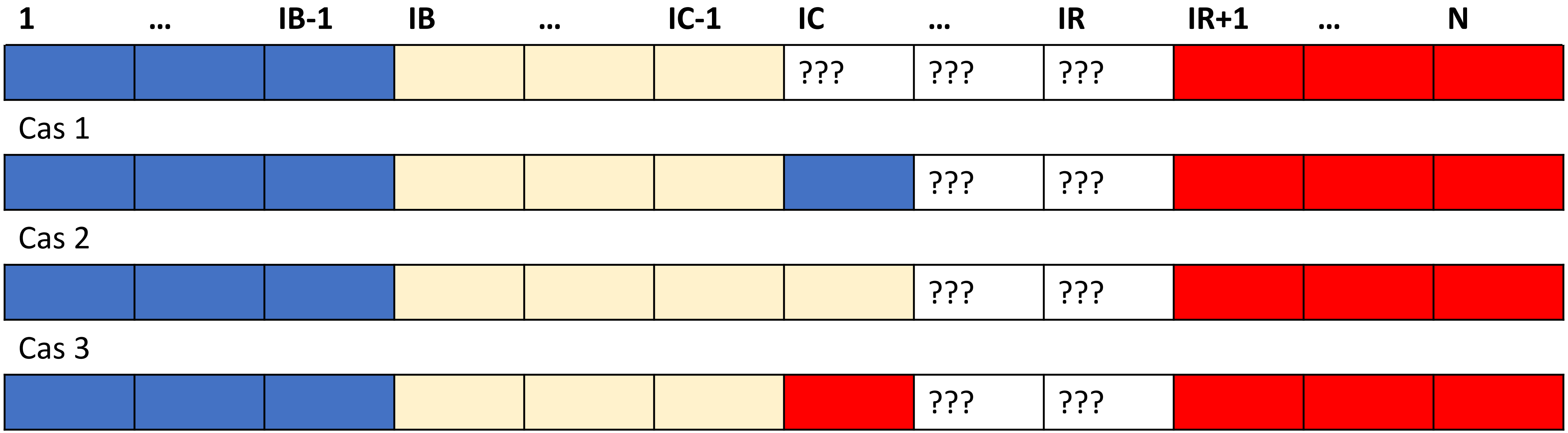
P1, P2, P3, P4 et P5

Complexité

- Le robot lit la couleur de chaque pierre deux fois, sauf une
(une seule lecture pour cette dernière)

- Peut-il faire la même chose en lisant une fois et une seule la couleur de chaque pierre ?

Idées d’un autre algorithme



Prédicats liés à ce nouvel algorithmes

- Un prédicat pour affirmer « que des pierres bleues au début »
  - PBl(IB) : Pour tout i, $0 < i < IB$, Couleur(i) = Bleu

- Un prédicat pour affirmer « que des pierres rouges à la fin »
  - PR(IR) : Pour tout j, $IR < j \leq N$, Couleur(j) = Rouge

- Un prédicat pour affirmer « que des pierres blanches dans la zone »
  - PW(IB,IC) : Pour tout k, $IB \leq k < IC$, Couleur(k) = Blanc

- Un prédicat pour « zone des pierres de couleur non connue »
  - PCoulInc (IC,IR) : Pour tout k, $IC \leq k \leq IR$, Couleur(k) $\in$ {Bleu, Blanc, Rouge}

Etude des cas : Cas 2

- Puisque La couleur lue est blanche on obtient :
  - PCoulInc (IC+1,IR) : La zone des pierres de couleur inconnue est devenue plus petite puisque nous avons lu la case IC
  - PW (IB, IC+1) : La zone des pierres blanches a grandi d’une case
  - PBl(IB) : sans changements
  - PR(IR) : sans changements

- S’il reste des pierres de couleur inconnue on peut lire la couleur de la pierre suivante (en IC+1)

Etude des cas : Cas 1

- Puisque La couleur lue est bleue on obtient :
  - PCoulInc (IC+1,IR) : La zone des pierres de couleur inconnue est devenue plus petite puisque nous avons lu la case IC
- On échange les pierres d’indices IB et IC
  - PW (IB+1, IC+1) : La zone des pierres blanches est décalée d’une case
  - PBl(IB+1) :La zone des pierres bleue compte une pierre de plus
  - PR(IR) : sans changements sous réserve que IB ≤ IC ≤ IR

- S’il reste des pierres de couleur inconnue on peut lire la couleur de la pierre suivante (en IC+1)

Etude des cas : Cas 3

- Puisque La couleur lue est Rouge on obtient :
  - PCoulInc (IC+1,IR) : La zone des pierres de couleur inconnue est devenue plus petite puisque nous avons lu la case IC
- On échange les pierres d’indices IR et IC
  - PCoulInc (IC,IR-1) : La zone des pierres de couleur inconnue est modifiée
  - PW (IB, IC) : La zone des pierres blanches est inchangée sous réserve que IB ≤ IC ≤ IR
  - PBl(IB) :La zone des pierres bleue est inchangée sous réserve que IB ≤ IC ≤ IR
  - PR(IR-1) : la zone des pierres rouge a grandie car il y a une pierre rouge à l’indice IR
- S’il reste des pierres de couleur inconnue on peut lire la couleur de la pierre suivante (en IC)

Peut-on commencer ?

- Pour que les prédicats soit vrai au début il suffit de poser :
  - IB = 1 (pas de pierres bleues connues)
  - IC = 1 (pas de pierres blanches connues)
  - IR = N (pas de pierres rouges connues)

Quand s’arrêter ?

- Il faut arrêter quand il n’existe plus de pierres de couleur inconnue. C’est-à-dire que $IR < IC$

Algorithme drapeau hollandais version 2

```
Variables IB, IC, IR : Entiers;
DébutCode
    IB ←1; IC←1; IR ←N;
    Tant que IC ≤ IR faire
        CasOù LireCouleur(IC) vaut
            Bleu : Echange(IB,IC); IB←IB+1; IC←IC+1;
            Blanc : IC ← IC + 1;
            Rouge : Echange(IC,IR); IR ← IR - 1;
        FinCasOù
    FinTq
FinCode
```

Complexité

- L'algorithme fait N tours de boucle
  - Si i est le nombre de tours de boucle totalement effectué IR-IC+i+1 = Cste = N
  - IR – IC décroit de 1 à chaque tour de boucle on a donc N tours de boucle

- Chaque tour de boucle contient exactement 1 lecture
- Chaque tour de boucle contient au plus un échange

- On a au plus N lectures et N échanges