M1 Informatique - Module Algorithmique et complexité

Intelligence Artificiel pour le Mastermind

Annexe 2 - Pseudo-code détaillé des Algorithmes

Benoît MARECHAL et Marc RASTOIX

Encadrant : Jean-Jacques CHABRIER

Table des matières

1 Stratégie 1 : Implémentation des critères 1 et 2

Listing 1 – Pseudo-code détaillé de la stratégie 1

```
1
2  On prend en paramètre de la stratégie :
3          - un tableau contenant la combinaison à trouver
4          - Le nombre de couleurs
5          - Le nombre de cases
6    Et on renvoi le nombre de tentatives effectuées
7
8
9  Fonction  (combi, n, k) --> Strategie1 --> NbTentatives
10
11 // Initialisations des variables
12
13 coul <- 1 // Couleur courante à testé, on commence avec la 1ière
       couleur
14
15 combiTente <- initialiser à la taille k // Tableau contenant la
       combinaison tentées par l'ia
16
17 combiTrouve <- initialiser à la taille k // Tableau contenant les
       couleurs trouvées
18
19 nbBienPlace <- 0 // Nb de boules bien placé ds la combi tente
20 nbMalPlace <- 0 // Nb de boules mal placé ds la combi tente
21 nbTrouve <- 0 // Nb de boules de couleurs dont ont a trouvé la position
22 nbBoule <- 0 // Nb d'occurence de la couleur courante
23 nbTentative <- 0 // Nb de tentatives total pour résoudre la combinaison
24 pos <- 0 // Position courante testé
25
26
27 // Boucle générant les tentatives jusqu'à résolution de la combinaison
28
29 Tant que nbBienPlace < k Faire
30
31 // On crée la nouvelle combinaison à tenté, qui
32 // determine la  presence ou non d'une couleur.
33 // Tout en tenant compte des  positions des
34 // boules de couleurs deja trouvées.
35
36         Pour i <- 0 jusqu'à k-1 Faire
37                         Si combiTrouve[i] = 0 Alors
38                     combiTente[i] <- coul
39                 Sinon
40                     combiTente[i] <- combiTrouve[i]
41         FinPour
```

```
42
43
44 // On détermine le nombre de bien placés et mal placés
45 nbBienPlace <- appeler Fonction nbBienPlace(combi, combiTente, k)
46 nbMalPlace <-  appeler Fonction nbCommuns(combi, combiTente, k) -
      appeler Fonction nbBienPlace(combi, combiTente, k)
47
48 // On incrémente le nombre de tentatives
49             nbTentative <- nbTentative + 1
50
51 // On détermine le nombre d'occurences de la couleur courante
52             nbBoules <- nbBienPlace - nbTrouve
53
54 // On vérifie que la couleur courante est présente
55           Si nbBoules >= 1 ET coul <= (n + 1) Faire
56
57           Pour x <- 1 jusqu'à x = nbBoules Faire
58 // On met nbMalPlace à un nombre différent de 0
59                 nbMalPlace <- 1
60
61 // Indice de la position testé pour trouver l'emplacement de
62 // la couleur coul. On ne test pas une position qui est déjà
63 // prise par une couleur dont l'emplacement est connu, donc
64 //  on crée une boucle qui cherche une position possible.
65
66                     pos <- 0;
67
68 // Tant qu'on à pas trouver la bonne position
69                 Tant que nbMalPlace != 0 Faire
70 // On détermine une position non encore trouvée
71                         Tant que (pos < k) ET (combiTrouve[pos] != 0)
72                                 pos <- pos + 1
73
74 // On crée la nouvelle combinaison à tenté, qui cherche la
75 // position exacte de la boule de couleur en cour. Tout en
76 // tenant compte des positions des boules de couleurs deja
77 // trouvées
78
79                 Pour de  i = 0 jusqu'à k-1 Faire
80 // Si la case courante, n'est pas une case dont on connais la couleur
81                         Si (combiTrouve[i] == 0) Alors
82 // Si la case n'est pas la case  testé, on met la boule de
83 // couleur supérieur
84                                 Si (i != pos) Alors
85                                         combiTente[i] = coul + 1;
86                                 Sinon
87                                         combiTente[i] <- coul;
88                                 FinSi
```

```
                        Sinon
                                combiTente[i] <- combiTrouve[i];
                        FinSi


// Calcul du nombre de boule mal placé
        nbMalPlace <- nbCommuns(combi,combiTente,k) - nbBienPlace

// Incrémentation du nombre de tentatives
                nbTentative <- NbTentative + 1

// On se prépare à tester la position suivante
                pos <- pos + 1

                    FinPour

// A la sortie de la boucle, on a la position de
// la boule de couleur -> pos - 1
// On ajoute donc cette boule à la combinaison contenant les
// boules Trouvées
                    combiTrouve[pos - 1] <- coul

                    // on incrémente le nombre de boule trouvées
                    nbTrouve <- nbTrouve + 1
                FinPour
            FinPour
            coul<- coul + 1
        FinTantQue
        return nbTentative;

End Fonction
```

2 Stratégie 2 : Implémentation des critères 1 et 2

Listing 2 – Pseudo-code détaillé de la stratégie 2

```
On prend en paramètre de la stratégie :
- un tableau contenant la combinaison à trouver,
- le nombre de couleurs,
- le nombre de case.
- un booléen pour savoir si les répétitions sont autorisées ou non

Et on renvoie le nombre de tentatives effectuées.
Fonction(combi,n,k,repetition) --> Strategie 2 --> NbTentatives

// Initialisation des variables
```

```
11
12 bienplace <- 0 // Nombre de pions bien placés
13 malplace <- 0 // Nombre de pions mal placés
14 nbtentative <- 0 // Nombre de combinaisons tentées
15 temp <- 1
16 drap <- 1
17 nbi <- 1 // Nombre d'occurrence restant à placer de la couleur i en
       cours
18 nombredejatrouve <- 0 // Nombre de boules qu'on a déjà reussi à placer
19 combitemp <- initialisé à la taille k et remplit de -1
20 combitente <- initialisé à la taille k et remplit de -1 // Combinaison
       proposée au programme
21 combitrouve <- initialisé à la taille k et remplit de -1 // Combinaison
       finale
22
23
24 Pour chaque couleur identifiée par i de 1 à n
25 // Si la solution n'est pas encore trouvée
26         Si bienplace != k
27 // On affecte tous les pions pas encore trouves a la couleur courante i
28 // Et on parcourt combitrouve avec la variable j
29
30         Si combitrouve[j] = -1 Alors
31                 combitente[j] <- i
32
33 // On cherche le nombre de pions biens et mals places
34         bienplace <- appeler fonction nbBienPlace(combi, combitente, k)
35         malplace <- appeler fonction nbCommuns(combi, combitente, k) -
              appeler fonction nbBienPlace(combi, combitente, k)
36
37                 // Cela fait une tentative de plus
38                 nbtentative <- nbtentative + 1
39
40                    //Nombre d'occurence de la couleur en i en cours
41                 nbi=bienplace-nombredejatrouve;
42
43 // Tant que tous les pions de la couleur en cours i ne
44 // sont pas bien placés
45                 Tant que nbi>0
46                         temp <- 1
47                         drap <- 1 // le drapeau est baissé
48 // On recopie combitrouve dans combitemp
49                         combitemp[l]=combitrouve[l];
50 // Tant que le pion courant de couleur courante i
51 // est mal placé
52                         Faire
53 // Si on a déjà teste une position non valable
54                                 Si temp!=-1Alors
```

```
55 // On passe à la couleur suivante pour le pion temp
56                                         combitente[temp] <- i+1
57                                                 combitemp[temp]
                                                       <- -1
58 // On baisse le drapeau
59                                         drap=-1;
60                                 FinSi
61                         Pour j allant de (temp+1) à k exclu
62 // Si le pion courant a deja ete trouve
63                         Si combitrouve[j] != -1 Alors
64 //On le copie dans la combinaison a la bonne position
65                                 combitente[j] <- combitrouve[j];
66                         Sinon
67 //Sinon si le drapeau n'est pas mis et donc que la couleur n'a pas deja
       ete affectee
68                                 Si drap == -1 Alors
69 //On tente la couleur courante
70                                         combitente[j] <- i
71                                         combitemp[j] <- i
72                                         temp <- j
73 //On lève le drapeau
74                                         drap <- 0
75 Sinon
76 // Sinon on passe à la couleur suivante pour le pion courant
77                                         combitente[j] <- i+1;
78                                 FinSi
79                         FinSi
80                 FinPour
81 //On cherche le nombre de pions biens et mals places
82 bienplace <- appeler fonction nbBienPlace(combi, combitemp, k)
83 malplace <- appeler fonction nbCommuns(combi, combitemp, k) - appeler
       fonction nbBienPlace(combi, combitemp, k)
84
85 // Cela fait une tentative de plus
86 nbtentative <- nbtentative + 1
87                         Si malplace!=0
88 // On recopie combitemp dans combitrouve
89                                 combitrouve[] <- combitemp[]
90 // Un pion de plus trouve
91                         nombredejatrouve <- nombredejatrouve
92 // Un pion de moins de couleur i a placer
93                         nbi <- nbi + 1
94                 FinTantQue
95         FinSi
96         //on retourne le nombre de tentatives
97             retourne nbtentative
98 FinPour
```

3 Stratégie 3 : Implémentation des critères 1, 2, 3 et 4

Listing 3 – Pseudo-code détaillé de la stratégie 3

```
1
2  // On prend en paramètre de la stratégie :
3  // - un tableau contenant la combinaison à trouver
4  // - le nombre de couleurs
5  // - le nombre de case
6  // - un booléen pour savoir si les répétitions sont autorisées ou non
7  // Et on renvoie le nombre de tentatives effectuées.
8
9
10 Fonction(combi,n,k,repetition) --> Strategie 3 --> NbTentatives
11
12 // Initialisations
13
14 coul <- 1; // Couleur testé, on commence avec la 1ier couleur : 1
15 combiTente <- initialisé à la taille k // Tableau contenant la
       combinaison
16 // tenté par l'ia
17 combiTrouve <-initialisé à la taille k // Tableau contenant les
       couleurs
18         // trouvées
19 nbBienPlace <- 0 // Indique le nombre de boules bien placées
20 nbMalPlace <- 0 // Indique le nombre de boule à la mauvaise position
21 nbTrouve <- 0 // Nombre de boules de couleurs dont ont a trouvé la
22         // position
23 nbTentative <- 0
24 pos <- 0
25
26 coulSuivante <- 1 // Contient la couleur suivante servant à remplir
27         // la combinaison lorsqu'on cherche la position de la couleur
              courante
28
29 nbBoulesSuivante <- 0
30
31
32 // Boucle générant les tentatives jusqu'à la résolution du MM
33
34 // Tant que le combinaison n'est pas trouvée
35 Tant que (nbBienPlace(combi, combiTente, k) < k )  Faire
36
37 // On place la couleur suivante dans la couleur courante
38         coul <- coulSuivante
39
40 // On crée la nouvelle combinaison à tenté, qui determine la
41 // presence ou non d'une couleur. Tout en tenant compte des
```

```
42 // positions des
43 // boules de couleurs deja trouvées
44             Pour i de 0 à k Faire
45                 Si (combiTrouve[i] = 0) Faire
46                     combiTente[i] <- coul
47                 Sinon
48                     combiTente[i] <- combiTrouve[i]
49             Fin Pour
50
51 // On détermine le nombre de bien placés et mal placés
52 nbBienPlace <- nbBienPlace(combi,combiTente,k)
53 nbMalPlace <- nbCommuns(combi,combiTente,k) - nbBienPlace(combi,
       combiTente,k)
54
55 // incrémentation du nombre de combi tenté
56             nbTentative <- nbTentative + 1
57
58 // Analyse des indications données par le nbBienPlace et nbMalPlace
59
60 // Si le nombre de bien placé = k, pas la peine de continué on a fini
       la résolution
61         Si (nbBienPlace = k) Faire
62                 Stopper l'iteration
63         Fin si
64
65 // Détermination du nombre de boules de la couleur courante
66             nbBoules <- nbBienPlace - nbTrouve
67
68
69 // Tester si la couleur est présente
70         Si nbBoules >= 1 ET coul <= (n + 1) Faire
71 // Si la couleur suivante est pas présente on la passe en couleur
       courante
72         Sinon Faire
73                 Si   (nbBoulesSuivante>0) Faire
74                      coul <- coulSuivante
75                         nbBoules <- nbBoulesSuivante
76                    Fin si
77  // Initialisation du nb de boules de la couleur suivante
78                 nbBoulesSuivante <- 0
79                 coulSuivante <- coul + 1
80
81 // Recherche de la position de chaque couleurs
82                 Pour x de 1à nbBoules Faire
83 // On initialise nbMalPlace à un nombre différent de 0
84                 nbMalPlace <- 1
85
86 // indice de la position testé pour trouver l'emplacement de
```

```
87  // la couleur coul. On ne test pas une position dont on
88  // connais deja la
89  // couleur. Donc on crée une boucle qui cherche une
90  // position possible.
91                  pos <- 0
92
93                  Tant que nbMalPlace > 0 Faire
94  // Recherche d'une position a tester qui n'a pas deja été trouvé
95                          Tant que (pos < k) ET (combiTrouve[pos] != 0)
96                                  pos <- pos  + 1
97
98  // On crée la nouvelle combinaison à tenté, qui cherche
99  // la position exacte de la boule de couleur
100 // en cour. Tout en tenant compte des positions des
101 // boules de couleurs deja trouvées
102                         Pour i de 0 à k Faire
103
104 // Si la case courante n'est pas une case dont on connais la couleur
105                                         Si combiTrouve[i] = 0 Faire
106 // Si la case n'est pas la case testé, on met la boule de couleur
    supérieur
107
108                                         Si (i != pos)
109                                         combiTente[i] <- coulSuivante
110                                         Sinon
111                                         combiTente[i] <- coul
112                                                 Fin si
113                                     Sinon
114                                         combiTente[i] <- combiTrouve[i]
115                                 Fin si
116                         Fin pour
117
118 // Test de la nouvelle tentative puis calcul du nombre de boules mal
    placé
119
120                 nbBienPlace <- appel de nbBienPlace(combi,combiTente,k)
121                 nbCommuns <- appel de nbCommuns(combi,combiTente,k)
122                 nbMalPlace <- nbCommuns - nbBienPlace
123
124 // On incrémente le nombre de tentatives
125                 nbTentative <- nbTentative + 1
126
127 // Analyse des indication donnée (Application du Critère 3)
128                 nbCoulSuivante <- (nbCommuns - nbTrouve - 1)
129
130 // Si la couleur suivante testé n'est pas présente on passe à la
    suivante
131                         Si nbCoulSuivante = 0
```

```
132                                 coulSuivante <- coulSuivante + 1
133
134                             Sinon
135                                 nbBoulesSuivante <- nbCoulSuivante;
136
137                             // On incrémente la position testé
138                             pos <- pos + 1
139                             Fin si
140                         Fin pour
141
142 // A la sortie de la boucle Pour, on a la position de la boule de
143 // couleur -> pos - 1
144 // On ajoute donc cette boule à la combinaison contenant les
145 // boules Trouvées
146                         combiTrouve[pos - 1] <- coul;
147 // On incrémente le nombre de boule trouvées
148                         nbTrouve <- nbTrouve + 1
149
150                 Fin Si
151
152               Jusqu'à nbBoulesSuivante >0 ET nbBienPlace != k
153         Sinon
154                 coulSuivante <- coulSuivante + 1
155
156         Fin Tant que
157
158         // Renvoi du nombre de tentatives effectuées
159         retourner nbTentative
160 Fin fonction
```