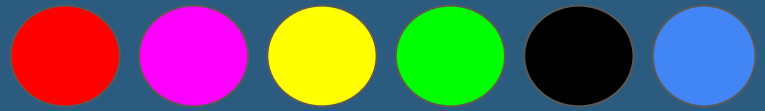
IUT de Montpellier - 2022 / 2023



Développement initiatique

Point SAE MasterMind

Enseignant :
Malo Gasquet

Rappels

- **Rappel** : Les **couleurs** sont représentées par des **nombres** (à partir de 0)
- Si les codes sont composés de 4 couleurs et qu'on a 6 couleurs, les codes peuvent donc aller de **(0,0,0,0)** à **(5,5,5,5)**
- La taille des codes et le nombre de couleurs sont configurables.
- Dans une manche **contre l'ordinateur**, le joueur décide d'un code (dans sa tête) et l'ordinateur fait des propositions. Le joueur lui répond alors un **couple de deux valeurs** indiquant le nombre de couleurs **bien placées** dans la proposition et le nombre de couleurs **mal placées** dans la proposition.

Rappels - Exemple

- **Contexte** : Code avec **4 couleurs**, **6 couleurs en tout**
- Le joueur a choisi le code : **(0,2,0,4)**
- Si l'ordinateur propose **(0,0,0,0)** le joueur lui répond **(2,0)**. En effet, il y a deux **0** bien placés et rien de mal placé (les deux autres **0** ne comptent pas car ils ne sont pas dans le code, ils sont en trop)
- Si l'ordinateur propose **(4,2,1,0)** le joueur lui répond **(1,2)**. En effet, le **2** est bien placé, le **4** et le **0** sont présents mais mal placés, et le **1** ne fait pas partie du code.

Stratégie de l’ordinateur

- Pour réussir à trouver le bon code en un **nombre réduit de propositions**, l’ordinateur va mettre en place la **stratégie** suivante :
    - Faire une première proposition (dans notre contexte : **(0,0,0,0)**)
    - Analyser et stocker le retour du joueur (bien / mal placés) sur cette proposition
    - Trouver un **autre code** en **incrémentant** la proposition précédente et en rejetant tous ceux qui ne sont pas **compatibles** avec le retour du joueur.
      Cela signifie rejeter tous les codes qui, si ils étaient comparés en tant que code solution sur les précédentes propositions de l’ordinateur ne donneraient pas le même retour que celui du joueur.
      En effet, ce code ne peut pas être la solution s' il ne produit pas les mêmes résultats! La proposition précédente est alors incrémentée jusqu’à trouver une proposition compatible qui sera testée auprès du joueur.

Stratégie de l’ordinateur

- L’ordinateur boucle ainsi jusqu’à ce que le joueur lui réponde **(4,0)** (toujours dans notre contexte), c'est-à-dire la **victoire**.

- Si on dépasse le maximum (dans notre contexte **(5,5,5,5)**) c’est que le joueur **triche** et ne donne pas les bons retours!

- Ainsi, à chaque proposition, la **“base d'exigence” de** l’ordinateur vient s’affiner (+ de propositions et de retours du joueur à tester) et il peut ainsi éliminer plus de code et converger vers la solution!

- Avec cette méthode, l’ordinateur fait le moins possible de propositions au joueur. On est certain que l’ordinateur fait, à chaque fois, une proposition plus pertinente que la précédente.

Stratégie de l’ordinateur - Exemple

PHASE PROPOSITION

- **Contexte** : **4** couleurs, **6** couleurs en tout
  Code joueur **: (0,2,0,4)**

- Proposition **#0** : **(0,0,0,0)**. Réponse joueur : **(2,0)**
  (deux 0 bien placés)

Mémoire de l’ordinateur :

#0 : **(0,0,0,0)** donne **(2,0)**

Stratégie de l’ordinateur - Exemple

PHASE RECHERCHE CODE COMPATIBLE
(en interne)

- **Contexte** : **4** couleurs, **6** couleurs en tout
- Incrémentation : **(0,0,0,0)** devient **(0,0,0,1)**
- Si **(0,0,0,1)** était le code, on aurait la réponse **(3,0)** pour la proposition **#0**.
- Cela ne correspond pas à la réponse qu’on a eu précédemment! **(2,0)**. Ce code n’est donc pas compatible! On ne le propose pas au joueur.

Mémoire de l’ordinateur :

#0 : **(0,0,0,0)** donne **(2,0)**

Stratégie de l’ordinateur - Exemple

PHASE RECHERCHE CODE COMPATIBLE
(en interne)

- **Contexte** : **4** couleurs, **6** couleurs en tout
- Incrémentation : **(0,0,0,1)** devient **(0,0,0,2)**
- Si **(0,0,0,2)** était le code, on aurait la réponse **(3,0)** pour la proposition **#0**.
- Cela ne correspond pas à la réponse qu’on a eu précédemment! **(2,0)**. Ce code n’est donc pas compatible! On ne le propose pas au joueur.

Mémoire de l’ordinateur :

#0 : **(0,0,0,0)** donne **(2,0)**

Stratégie de l’ordinateur - Exemple

L’ordinateur élimine ainsi beaucoup de codes non compatibles…

Jusqu’à…!

Stratégie de l’ordinateur - Exemple

PHASE RECHERCHE CODE COMPATIBLE
(en interne)

- **Contexte** : **4** couleurs, **6** couleurs en tout
- Incrémentation : **(0,0,0,5)** devient **(0,0,1,0)**
- Si **(0,0,1,0)** était le code, on aurait la réponse **(3,0)** pour la proposition **#0**.
- Cela ne correspond pas à la réponse qu’on a eu précédemment! **(2,0)**. Ce code n’est donc pas compatible! On ne le propose pas au joueur.

Mémoire de l’ordinateur :

#0 : **(0,0,0,0)** donne **(2,0)**

Stratégie de l’ordinateur - Exemple

PHASE RECHERCHE CODE COMPATIBLE
(en interne)

- **Contexte** : **4** couleurs, **6** couleurs en tout
- Incrémentation : **(0,0,1,0)** devient **(0,0,1,1)**
- Si **(0,0,1,1)** était le code, on aurait la réponse **(2,0)** pour la proposition **#0**.
- Cela correspond à la réponse qu’on a eu précédemment! **(2,0)**. Ce code est donc compatible! On va le proposer au joueur!

Mémoire de l’ordinateur :

#0 : **(0,0,0,0)** donne **(2,0)**

Stratégie de l’ordinateur - Exemple

PHASE PROPOSITION

- **Contexte** : **4** couleurs, **6** couleurs en tout
  Code joueur **: (0,2,0,4)**

- Proposition **#1** : **(0,0,1,1)**. Réponse joueur : **(1,1)**
  (Un 0 est bien placé, un autre 0 est mal placé)

Mémoire de l’ordinateur :

#0 : **(0,0,0,0)** donne **(2,0)**
#1 : **(0,0,1,1)** donne **(1,1)**

Stratégie de l’ordinateur - Exemple

PHASE RECHERCHE CODE COMPATIBLE
(en interne)

- **Contexte** : **4** couleurs, **6** couleurs en tout
- Incrémentation : **(0,0,1,1)** devient **(0,0,1,2)**
- Si **(0,0,1,2)** était le code:
  - **(2,0)** pour la proposition **#0**. **OK**
  - **(3,0)** pour la proposition **#1**. **NON COMPATIBLE**
- Le code n’est pas compatible avec les résultats de **toutes** les propositions précédentes, il est rejeté.

Mémoire de l’ordinateur :

#0 : **(0,0,0,0)** donne **(2,0)**
#1 : **(0,0,1,1)** donne **(1,1)**

Stratégie de l’ordinateur - Exemple

PHASE RECHERCHE CODE COMPATIBLE
(en interne)

- **Contexte** : **4** couleurs, **6** couleurs en tout
- Incrémentation : **(0,0,1,2)** devient **(0,0,1,3)**
- Si **(0,0,1,3)** était le code:
  - **(2,0)** pour la proposition **#0**. **OK**
  - **(3,0)** pour la proposition **#1**. **NON COMPATIBLE**
- Le code n’est pas compatible avec les résultats de **toutes** les propositions précédentes, il est rejeté.

Mémoire de l’ordinateur :

#0 : **(0,0,0,0)** donne **(2,0)**
#1 : **(0,0,1,1)** donne **(1,1)**

Stratégie de l’ordinateur - Exemple

PHASE RECHERCHE CODE COMPATIBLE
(en interne)

- **Contexte** : **4** couleurs, **6** couleurs en tout
- Incrémentation : **(0,0,1,3)** devient **(0,0,1,4)**
- Si **(0,0,1,4)** était le code:
  - **(2,0)** pour la proposition **#0**. **OK**
  - **(3,0)** pour la proposition **#1**. **NON COMPATIBLE**
- Le code n’est pas compatible avec les résultats de **toutes** les propositions précédentes, il est rejeté.

Mémoire de l’ordinateur :

#0 : **(0,0,0,0)** donne **(2,0)**
#1 : **(0,0,1,1)** donne **(1,1)**

Stratégie de l’ordinateur - Exemple

L’ordinateur élimine ainsi beaucoup de codes non compatibles…

Jusqu’à…!

Stratégie de l’ordinateur - Exemple

PHASE RECHERCHE CODE COMPATIBLE
(en interne)

- **Contexte** : **4** couleurs, **6** couleurs en tout
- Incrémentation : **(0,2,0,0)** devient **(0,2,0,1)**
- Si **(0,2,0,1)** était le code:
  - **(2,0)** pour la proposition **#0**. **OK**
  - **(2,1)** pour la proposition **#1**. **NON COMPATIBLE**
- Le code n’est pas compatible avec les résultats de **toutes** les propositions précédentes, il est rejeté.

Mémoire de l’ordinateur :

#0 : **(0,0,0,0)** donne **(2,0)**
#1 : **(0,0,1,1)** donne **(1,1)**

Stratégie de l’ordinateur - Exemple

PHASE RECHERCHE CODE COMPATIBLE
(en interne)

- **Contexte** : **4** couleurs, **6** couleurs en tout
- Incrémentation : **(0,2,0,1)** devient **(0,2,0,2)**
- Si **(0,2,0,2)** était le code:
  - **(2,0)** pour la proposition **#0**. **OK**
  - **(1,1)** pour la proposition **#1**. **OK**
- Le code est compatible avec les résultats de **toutes** les propositions précédentes! Il va être proposé.

Mémoire de l’ordinateur :

#0 : **(0,0,0,0)** donne **(2,0)**
#1 : **(0,0,1,1)** donne **(1,1)**

Stratégie de l’ordinateur - Exemple

PHASE PROPOSITION

- **Contexte** : **4** couleurs, **6** couleurs en tout
  Code joueur **: (0,2,0,4)**

- Proposition **#2** : **(0,2,0,2)**. Réponse joueur : **(3,0)**
  (0, 2 et un autre 0 bien placés)

Mémoire de l’ordinateur :

#0 : **(0,0,0,0)** donne **(2,0)**
#1 : **(0,0,1,1)** donne **(1,1)**
#2 : **(0,2,0,2)** donne **(3,0)**

Stratégie de l’ordinateur - Exemple

PHASE RECHERCHE CODE COMPATIBLE
(en interne)

- **Contexte** : **4** couleurs, **6** couleurs en tout
- Incrémentation : **(0,2,0,2)** devient **(0,2,0,3)**
- Si **(0,2,0,3)** était le code:
  - **(2,0)** pour la proposition **#0**. **OK**
  - **(1,1)** pour la proposition **#1**. **OK**
  - **(3,0)** pour la proposition **#2**. **OK**

Mémoire de l’ordinateur :

#0 : **(0,0,0,0)** donne **(2,0)**
#1 : **(0,0,1,1)** donne **(1,1)**
#2 : **(0,2,0,2)** donne **(3,0)**

Stratégie de l’ordinateur - Exemple

PHASE PROPOSITION

- **Contexte** : **4** couleurs, **6** couleurs en tout
  Code joueur **: (0,2,0,4)**

- Proposition **#3** : **(0,2,0,3)**. Réponse joueur : **(3,0)**
  (0, 2 et un autre 0 bien placés)

Mémoire de l’ordinateur :

#0 : **(0,0,0,0)** donne **(2,0)**
#1 : **(0,0,1,1)** donne **(1,1)**
#2 : **(0,2,0,2)** donne **(3,0)**
#3 : **(0,2,0,3)** donne **(3,0)**

Stratégie de l’ordinateur - Exemple

PHASE RECHERCHE CODE COMPATIBLE
(en interne)

- **Contexte** : **4** couleurs, **6** couleurs en tout
- Incrémentation : **(0,2,0,3)** devient **(0,2,0,4)**
- Si **(0,2,0,4)** était le code:
  - **(2,0)** pour la proposition **#0**. **OK**
  - **(1,1)** pour la proposition **#1**. **OK**
  - **(3,0)** pour la proposition **#2**. **OK**
  - **(3,0)** pour la proposition **#3**. **OK**

Mémoire de l’ordinateur :

#0 : **(0,0,0,0)** donne **(2,0)**
#1 : **(0,0,1,1)** donne **(1,1)**
#2 : **(0,2,0,2)** donne **(3,0)**
#3 : **(0,2,0,3)** donne **(3,0)**

Stratégie de l’ordinateur - Exemple

PHASE PROPOSITION

- **Contexte** : **4** couleurs, **6** couleurs en tout
  Code joueur **: (0,2,0,4)**

- Proposition **#3** : **(0,2,0,4)**. Réponse joueur : **(4,0)**
  Tout est bien placé ! **Victoire**!

  L’ordinateur n’a fait que **5 propositions** pour gagner!

Mémoire de l’ordinateur :

#0 : **(0,0,0,0)** donne **(2,0)**
#1 : **(0,0,1,1)** donne **(1,1)**
#2 : **(0,2,0,2)** donne **(3,0)**
#3 : **(0,2,0,3)** donne **(3,0)**

Côté Java

- Cette **stratégie** s'implémente grâce à **3 algorithmes** dans votre programme :
  - **passeCodeSuivantLexico**
  - **estCompat**
  - **passeCodeSuivantLexicoCompat**

Côté Java - passeCodeSuivantLexico

- **passeCodeSuivantLexico** a pour but **“d’incrémenter”** le code passé en paramètre en tenant compte de la longueur du code et du nombre de couleurs possibles.

- C’est cette fonction qu’utilise l’ordinateur pour obtenir un nouveau code qu’il va évaluer pour voir s’il est **compatible** ou non.

- La fonction prend deux paramètres : le **code** (**cod1**) à incrémenter (donc, un tableau dont la taille correspond à la longueur du code) et le **nombre de couleurs** possibles (**nbCouleurs**).

- Le code à incrémenter est **directement modifié** via le tableau passé en paramètre.

- Si jamais on atteint la limite (on tente d’incrémenter le dernier code de l’ensemble) cette fonction met le code à 0 et retourne **false**. Cela signifie que le joueur a triché! Si tout se passe bien, elle retourne **true**.

Côté Java - passeCodeSuivantLexico

- Exemple : **cod1 = {0,2,5,4}**; **nbCouleurs = 6**
  - Après exécution : **cod1 = {0,2,5,5}**
  - Retourne **true**
- Exemple : **cod1 = {2,2,3}**; **nbCouleurs = 4**
  - Après exécution : **cod1 = {2,3,0}**
  - Retourne **true**
- Exemple : **cod1 = {2,4,4,4,4}**; **nbCouleurs = 5**
  - Après exécution : **cod1 = {3,0,0,0,0}**
  - Retourne **true**

Côté Java - passeCodeSuivantLexico

- Exemple : **cod1 = {0,1,1,1}**; **nbCouleurs = 2**
    - Après exécution : **cod1 = {1,0,0,0}**
    - Retourne **true**
- On peut voir cette opération comme l'incrémentation d'un nombre représenté en **base nbCouleurs**! (dans l'exemple ci-dessus, en **base binaire**!)
- Exemple : **cod1 = {6,6,6,6}**; **nbCouleurs = 7**
    - Après exécution : **cod1 = {0,0,0,0}**
    - Retourne **false**

Côté Java - estCompat

- **estCompat** a pour but de tester la **compatibilité** d'un code avec les réponses déjà données par le joueur à un ensemble de propositions de l'ordinateur.

- On génère les réponses produites en testant les anciennes propositions de l'ordinateur sur le code passé en paramètre et on les compare aux véritables réponses du joueur.

- Si **toutes les réponses** générées pour chaque proposition avec le code testé sont les mêmes que celles proposées par le joueur, **le code est donc compatible**, la fonction retourne donc **true**.

- A partir du moment où **au moins une des réponses générées n'est pas identique à celle du joueur**, le code n'est donc **pas compatible** et on retourne donc **false** (le code testé ne peut pas être le code choisi par le joueur…!)

Côté Java - estCompat

- **estCompat** prend donc **5 paramètres** :
    - Le code dont on souhaite tester la compatibilité (**cod1**). C'est un **tableau** dont la taille correspond à la longueur du code.
    - Toutes les propositions déjà faites par l'ordinateur (**cod**). C'est un **tableau à deux dimensions** (matrice). Chaque case du tableau contient un code (donc, un tableau d'entiers qui correspondent à une proposition faite par l'ordinateur au joueur).
    - Toutes les réponses données par le joueur à l'ordinateur par rapport à ses propositions (**rep**). C'est un **tableau à deux dimensions** (matrice). Chaque case du tableau contient **un tableau de deux cases** correspondant au couple : **(nbBienPlacés, nbMalPlacés)**.

Côté Java - estCompat

- **estCompat** prend donc **5 paramètres** :
    - Le **nombre de propositions** déjà faites par l'ordinateur (**nbCoups**). Cela correspond à la zone "remplie" de la première dimension des matrices **cod** et **rep** qui correspondent respectivement aux propositions de l'ordinateur et aux réponses associées. Il est donc intéressant de noter que, par exemple, la réponse à la proposition **cod[2]** est stockée dans **rep[2]**.
    - Le **nombre de couleurs** possibles (**nbCouleurs**)

Côté Java - estCompat

- Exemple : **cod1 = {0,0,1,0}**; **cod = {{0,0,0,0}}**, **rep = {{2,0}}**,
  nbCoups = 1, **nbCouleurs = 6**

  - Si {0,0,1,0} était le code, **cod[0],** donc {0,0,0,0} donnerait {3,0} comme réponse, ce qui est différent de **rep[0],** {2,0}
  - Retourne **false**

- Exemple : **cod1 = {0,0,1,1}**; **cod = {{0,0,0,0}}**, **rep = {{2,0}}**,
  nbCoups = 1, **nbCouleurs = 6**

  - Si {0,0,1,1} était le code, **cod[0],** donc {0,0,0,0} donnerait {2,0} comme réponse, ce qui est égal à **rep[0],** {2,0}
  - Retourne **true**

Côté Java - estCompat

- Exemple : **cod1 = {0,2,0,1}**; **cod = {{0,0,0,0}, {0,0,1,1}}**, **rep = {{2,0}, {1,1}}**,
  nbCoups = 2, **nbCouleurs = 6**

  - Si {0,2,0,1} était le code, **cod[0],** donc {0,0,0,0} donnerait {2,0} comme réponse, ce qui est égal à **rep[0],** {2,0}
  - Si {0,2,0,1} était le code, **cod[1],** donc {0,0,1,1} donnerait {2,1} comme réponse, ce qui est n'est pas égal à **rep[1],** {1,1}
  - Retourne **false**

Côté Java - estCompat

- Exemple : **cod1 = {0,2,0,2}**; **cod = {{0,0,0,0}, {0,0,1,1}}**, **rep = {{2,0}, {1,1}}**,
  nbCoups = 2, **nbCouleurs = 6**

  - Si {0,2,0,2} était le code, **cod[0],** donc {0,0,0,0} donnerait {2,0} comme réponse, ce qui est égal à **rep[0],** {2,0}
  - Si {0,2,0,2} était le code, **cod[1],** donc {0,0,1,1} donnerait {1,1} comme réponse, ce qui est est égal à **rep[1],** {1,1}
  - Retourne **true**

Côté Java - estCompat

- Exemple : **cod1 = {0,2,0,4}**; **cod = {{0,0,0,0}, {0,0,1,1}, {0,2,0,2}, {0,2,0,3}}**,
  rep = {{2,0}, {1,1}, {3,0}, {3,0}}, **nbCoups = 4**, **nbCouleurs = 6**

  - Si {0,2,0,4} était le code, **cod[0],** donc {0,0,0,0} donnerait {2,0} comme réponse, ce qui est égal à **rep[0],** {2,0}
  - Si {0,2,0,4} était le code, **cod[1],** donc {0,0,1,1} donnerait {1,1} comme réponse, ce qui est est égal à **rep[1],** {1,1}
  - Si {0,2,0,4} était le code, **cod[2],** donc {0,2,0,2} donnerait {0,3} comme réponse, ce qui est est égal à **rep[2],** {0,3}
  - Si {0,2,0,4} était le code, **cod[2],** donc {0,2,0,3} donnerait {0,3} comme réponse, ce qui est est égal à **rep[3],** {0,3}

  - Retourne **true**

Côté Java - passeCodeSuivantLexicoCompat

- Le rôle de cette fonction est très simple ! Elle **incrémente** la dernière proposition de l'ordinateur et teste sa compatibilité jusqu'à trouver un **code compatible** à proposer au joueur.

- Pour cela, elle utilise habilement les deux fonctions que nous venons de développer.

- Cette fonction renvoie **true** si un code compatible a été trouvé et **false** sinon (si elle renvoie **false**, c'est que la limite a été dépassé et que le joueur a triché!).

- A la sortie, le code initialement passé en paramètre est devenu le prochain **code compatible** à soumettre au joueur.

Côté Java - passeCodeSuivantLexicoCompat

- Exemple : **cod1 = {0,0,0,0}**; **cod = {{0,0,0,0}}**, **rep = {{2,0}}**,
  nbCoups = 1, **nbCouleurs = 6**

  - Après exécution : **cod1 = {0,0,1,1}**
  - Retourne **true**

- Exemple : **cod1 = {0,0,1,1}**; **cod = {{0,0,0,0}, {0,0,1,1}}**, **rep = {{2,0}, {1,1}}**,
  nbCoups = 2, **nbCouleurs = 6**

  - Après exécution : **cod1 = {0,2,0,2}**
  - Retourne **true**

Côté Java - passeCodeSuivantLexicoCompat

- Exemple : **cod1 = {0,2,0,2}**; **cod = {{0,0,0,0}, {0,0,1,1}, {0,2,0,2}}**,
  rep = {{2,0}, {1,1}, {3,0}}, **nbCoups = 3**, **nbCouleurs = 6**

  - Après exécution : **cod1 = {0,2,0,3}**
  - Retourne **true**

- Exemple : **cod1 = {0,2,0,3}**; **cod = {{0,0,0,0}, {0,0,1,1}, {0,2,0,2}, {0,2,0,3}}**,
  rep = {{2,0}, {1,1}, {3,0}, {3,0}}, **nbCoups = 4**, **nbCouleurs = 6**

  - Après exécution : **cod1 = {0,2,0,4}**
  - Retourne **true**

Côté Java - passeCodeSuivantLexicoCompat

- Exemple : **cod1 = {0,2,0,2}**; **cod = {{0,0,0,0}, {0,0,1,1}, {0,2,0,2}, {0,2,0,3}}**,
  rep = {{2,0}, {1,1}, {3,0}, {0,0}}, **nbCoups = 4**, **nbCouleurs = 6**

  - Après exécution : **cod1 = {0,0,0,0}**
  - Retourne **false**

- Le joueur a **triché** (ou s'est trompé) en répondant **(0,0)** à la proposition **(0,2,0,3)**. Même sans connaître le code du joueur, ces réponses ne sont compatibles avec aucun code, ce qui fait que (dans cette configuration) aucun code à proposer n'a été trouvé. Le programme peut donc détecter ce genre d'erreur!

Conclusion

A vous de jouer!

Concentrez-vous sur la partie de base avant de passer aux extensions!