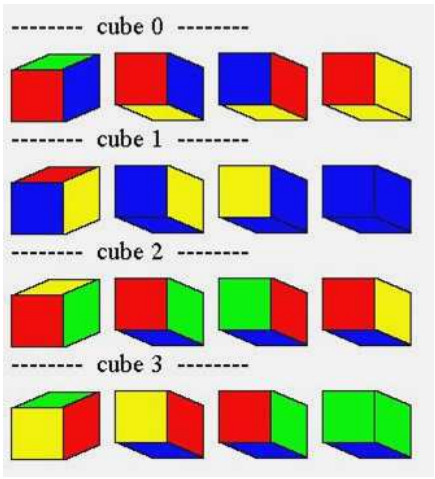
Le puzzle *Instant Insanity*

On dispose de quatre cubes, dont chacune des six faces est coloriée avec une couleur choisie parmi quatre couleurs (rouge, vert, bleu, jaune). L'objectif est d'empiler les quatre cubes l'un sur l'autre de façon que les quatre couleurs soient présentes sur chacune des quatre façades de l'empilage des cubes.

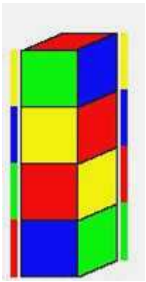
Exemple

- Coloriage des quatre cubes



(pour chaque cube, une vue de dessus et trois vues de dessous avec rotation d'axe vertical afin de visualiser les quatre faces verticales, avec la face de droite qui devient celle de devant)

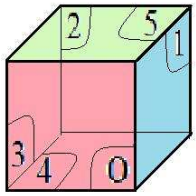
- Une solution du problème, avec les cubes 0, 1, 2, 3 empilés dans cet ordre de haut en bas



(deux faces sont visibles, celles qui sont cachées ont leurs couleurs indiquées sur deux colonnes de part et d'autre de l'empilage visible)

Remarques préalables, et début de la programmation

- l'ordre dans lequel sont disposés les cubes n'a pas d'importance. Nous choisirons de les prendre dans l'ordre de leurs numéros croissant de haut en bas.
- Le problème n'a pas toujours de solutions (par exemple si l'on colorie toutes les faces des quatre cubes avec une seule couleur).



- Prenons un cube, et différencions ses six faces. Plus précisément, choisissons une position du cube, et notons 0 la face de devant, 1 celle de droite, 2 celle qui est derrière, 3 celle de gauche, et 4 et 5 les faces du dessous et du dessus. Pour le programme, appelons *c*[*k*][*i*] la couleur donnée à la face *i* du cube *k*, la couleur portant un numéro de 0 à 3 (0 pour rouge, 1 pour vert, 2 pour bleu et 3 pour jaune). Cela donne par exemple :
-

```
c[0][0]=0; c[0][1]=2;  c[0][2]=0; c[0][3]=3;  c[0][4]=3; c[0][5]=1;
c[1][0]=2; c[1][1]=3;  c[1][2]=2; c[1][3]=2;  c[1][4]=2; c[1][5]=0;
c[2][0]=0; c[2][1]=1;  c[2][2]=0; c[2][3]=3;  c[2][4]=2; c[2][5]=3;
c[3][0]=3; c[3][1]=0;  c[3][2]=1; c[3][3]=1;  c[3][4]=2; c[3][5]=1;
```

Un cube étant maintenant colorié, il existe plusieurs angles de vue pour l'observer. Autrement dit, combien existe-t-il de façons de le poser sur le sol ? On choisit d'abord la face collée au sol (toujours numérotée 4), soit six cas. A chaque fois, on peut le tourner quatre fois de 90° autour d'un axe vertical, avec décalage cyclique des faces verticales. On trouve ainsi 24 placements possibles d'un cube. Avec les quatre cubes empilés l'un sur l'autre, cela fait 24^4 configurations. Mais chaque configuration se trouve répétée quatre fois par rotation verticale, et si l'on met l'empilage sens dessus dessous cela fait deux configurations identiques aussi. On peut donc se contenter de placer le cube 0 de trois façons différentes. Finalement le nombre de configurations distinctes est $3 . 24^3 = 41\,472$.

Notons que ces configurations sont considérées comme distinctes en différenciant les six faces du cube, comme si on les coloriait avec six couleurs différentes. Mais en tenant compte des quatre couleurs utilisées, certaines de ces configurations vont être vues comme identiques. Par exemple, si un cube est colorié d'une seule couleur, les 24 configurations se réduisent à une seule. Mais nous ne tiendrons pas compte de cela dans le programme. Celui-ci va fabriquer les 41 472 empilages possibles, même si certains ont les mêmes coloriages, et l'on ne gardera que ceux dont les quatre façades ont chacune les quatre couleurs présentes.

Commençons par construire les 24 positions de chaque cube, en utilisant un tableau *c*[*k*][*p*][*i*] où *k* est le numéro du cube, *p* sa position et *i* le numéro d'une face. Pour la position initiale *p*=0, on fait :

```
for(k=0;k<4;k++) for(i=0;i<6;i++) cube[k][0][i]=c[k][i];
```

Outre la position initiale 0, les autres positions essentielles sont la position *p*=8 obtenue en mettant comme face du bas (4) celle qui était la face 0 initialement, et la position *p*=16 où la face 4 est celle qui était la face 1 initialement. On a aussi la position *p*=4 obtenue en mettant à l'envers la position initiale *p*=0 : la face 4 est celle qui était la face 5, la face 5 est celle qui était la 4, tandis que les faces verticales 0 1 2 3 sont les faces qui étaient 3 2 1 0. On fera de même avec la position 12 obtenue en mettant sens

dessus dessous la position 8, et avec la position 20 en retournant la position 16. Commençons par la position 4 :

```
for(k=0;k<4;k++) for(i=0;i<4;i++)   cube[k][4][i]=c[k][3-i];
for(k=0;k<4;k++)  { cube[k][4][4]=c[k][5];  cube[k][4][5]=c[k][4];}
```

Passons maintenant aux positions 8, 16 et leurs retournements 12 et 20 :

```
for(k=0;k<4;k++) {cube[k][8][4]=c[k][0];  cube[k][16][4]=c[k][1];
                  cube[k][12][4]=c[k][0];  cube[k][20][4]=c[k][1]; }
for(k=0;k<4;k++) {cube[k][8][5]=c[k][2];  cube[k][16][5]=c[k][3];
                  cube[k][12][5]=c[k][2];  cube[k][20][5]=c[k][3]; }
for(k=0;k<4;k++) {cube[k][8][0]=c[k][5];  cube[k][16][0]=c[k][0];
                  cube[k][12][0]=c[k][3];  cube[k][20][0]=c[k][4]; }
for(k=0;k<4;k++) {cube[k][8][1]=c[k][1];  cube[k][16][1]=c[k][5];
                  cube[k][12][1]=c[k][4];  cube[k][20][1]=c[k][2]; }
for(k=0;k<4;k++) {cube[k][8][2]=c[k][4];  cube[k][16][2]=c[k][2];
                  cube[k][12][2]=c[k][1];  cube[k][20][2]=c[k][5]; }
for(k=0;k<4;k++) {cube[k][8][3]=c[k][3];  cube[k][16][3]=c[k][4];
                  cube[k][12][3]=c[k][5];  cube[k][20][3]=c[k][0]; }
```

Il reste à faire tous les cas intermédiaires, correspondant aux rotations d'axe vertical. A partir de la position 0, on construit les positions 1, 2, 3, à partir de la position 4, on a les positions 5, 6, 7, et ainsi de suite.

```
for(k=0;k<4;k++) for(p=1;p<4;p++) for(j=0;j<4;j++) cube[k][p][j]=cube[k][0][(j+p)%4] ;
for(k=0;k<4;k++) for(p=1;p<4;p++) for(j=4;j<6;j++)  cube[k][p][j]=cube[k][0][j] ;
for(k=0;k<4;k++) for(p=1;p<4;p++) for(j=0;j<4;j++) cube[k][8+p][j]=cube[k][8][(j+p)%4] ;
for(k=0;k<4;k++) for(p=1;p<4;p++) for(j=4;j<6;j++)  cube[k][8+p][j]=cube[k][8][j] ;
for(k=0;k<4;k++) for(p=1;p<4;p++) for(j=0;j<4;j++)
        cube[k][16+p][j]=cube[k][16][(j+p)%4] ;
for(k=0;k<4;k++) for(p=1;p<4;p++) for(j=4;j<6;j++)  cube[k][16+p][j]=cube[k][16][j] ;

for(k=0;k<4;k++) for(p=1;p<4;p++) for(j=0;j<4;j++)  cube[k][4+p][j]=cube[k][4][(j+p)%4] ;
for(k=0;k<4;k++) for(p=1;p<4;p++) for(j=4;j<6;j++)  cube[k][4+p][j]=cube[k][4][j] ;
for(k=0;k<4;k++) for(p=1;p<4;p++) for(j=0;j<4;j++)
      cube[k][12+p][j]=cube[k][12][(j+p)%4] ;
for(k=0;k<4;k++) for(p=1;p<4;p++) for(j=4;j<6;j++)  cube[k][12+p][j]=cube[k][12][j] ;
for(k=0;k<4;k++) for(p=1;p<4;p++) for(j=0;j<4;j++)
     cube[k][20+p][j]=cube[k][20][(j+p)%4] ;
for(k=0;k<4;k++) for(p=1;p<4;p++) for(j=4;j<6;j++)  cube[k][20+p][j]=cube[k][20][j] ;
```

La boucle du programme

Pour traiter les 41 472 cas, on fait une boucle pour chaque cube. On commence par le cube 0, avec ses trois configurations. Puis on prend les 24 configurations du cube 1. On ne garde que celles qui donnent des faces de couleur différentes sur chacune des quatre colonnes de l'empilage, ce que fait la fonction *different01*(). Pour les configurations restantes, on ajoute le cube 2 avec ses 24 positions, et l'on ne garde que les configurations donnant des couleurs différentes sur les colonnes, grâce à la fonction

different012(). Avec les configurations restantes, on fait intervenir les 24 positions du cube 3, et l'on garde celles qui sont valables grâce à la fonction *different0123*(). On trouve ainsi les solutions du problème.

```
for(p=0;p<24;p++) if (p%8==0)
for(pp=0;pp<24;pp++)
  { if (different01()==1)
     for(ppp=0;ppp<24;ppp++)
       { if (different012()==1)
          for(pppp=0; pppp<24; pppp++)
            {if (different0123()==1)
                { printf("\n(%d %d %d %d) ",p,pp,ppp,pppp);
                  for(col=0;col<4;col++)
                     { printf("    "); printf("%d %d %d %d",cube[0][p][col],
                                          cube[1][pp][col],cube[2][ppp][col],cube[3][pppp][col]);
                     }
                  nbsolutions++;
                }
            }
       }
  }
```

Avec les fonctions correspondantes, qui ramènent soit OUI (ou 1) soit NON (0) :

```
int different01(void)
  {  int colonne;
     for(colonne=0; colonne<4; colonne++)
     if (cube[1][pp][colonne]==cube[0][p][colonne]) return 0;
     return 1;
  }
 int different012(void)
  {  int colonne;
     for(colonne=0; colonne<4; colonne++)
     if (cube[2][ppp][colonne]==cube[1][pp][colonne]
          || cube[2][ppp][colonne]==cube[0][p][colonne] ) return 0;
     return 1;
  }
int different0123(void)
  {  int colonne;
     for(colonne=0; colonne<4; colonne++)
     if (cube[3][pppp][colonne]==cube[2][ppp][colonne]
          ||  cube[3][pppp][colonne]==cube[1][pp][colonne]
          || cube[3][pppp][colonne]==cube[0][p][colonne] ) return 0;
     return 1;
  }
```

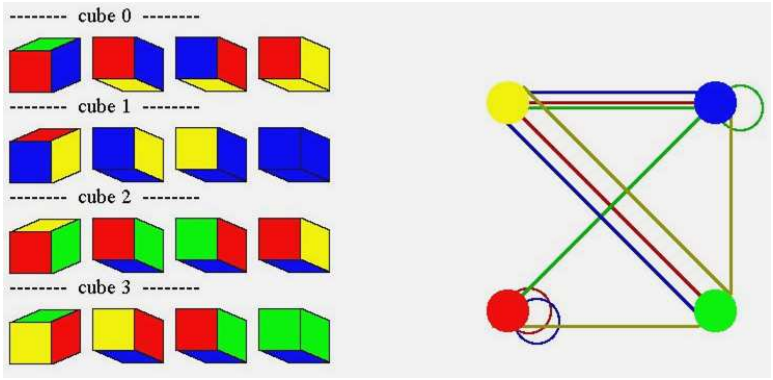
Dans l'exemple choisi, un des plus favorables en nombre de solutions, on trouve 28 solutions, mais certaines sont visuellement identiques, même si elles correspondant à

des positions différentes des cubes. On trouvera ci-dessous ces solutions, chacune avec son numéro, ainsi que les quatre numéros des positions des cubes. Cette étude expérimentale va nous conduire à une ébauche de théorie.

Ebauche théorique

A partir des quatre cubes coloriés, construisons un graphe. Cee graphe possède quatre sommets correspondant aux quatre couleurs (rouge 0, vert 1, bleu 2, jaune 3). On considère ensuite les faces opposées de chaque cube, soit les paires formées des faces 0 et 2 (devant – derrière), 1 et 3 (droite - gauche), 4 et 5 (dessous – dessus). Pour chaque paire avec ses deux couleurs des faces, on trace une arête de jonction sur le graphe, entre les deux couleurs concernées. Chaque cube donne trois arêtes, que l'on numérote avec le numéro du cube. Cela fait au total 12 arêtes, numérotées avec les quatre numéros des cubes.

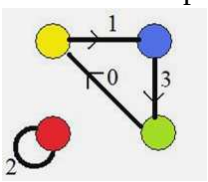
Dans l'exemple choisi, on a le graphe suivant :

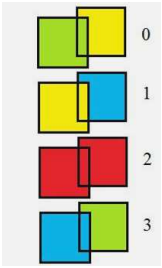


Sur le dessin de droite, à défaut de numéros, les arêtes du graphe portent des couleurs différentes selon le cube concerné. Par exemple le cube 0 a ses faces 0 et 2 rouges toutes deux, d'où une boucle sur le sommet rouge du graphe, il a ses faces 1 et 3 bleue et jaune, d'où une arête, et ses faces 4 et 5 jaune et verte, avec une troisième arête.

Prenons maintenant une des solutions du problème. Seules faces devant-derrière, et droite-gauche jouent un rôle. Dessinons un premier graphe (un sous-graphe du précédent), celui correspondant aux faces devant et derrière de l'empilage trouvé. On trouve quatre arêtes avec leurs numéros de 0 à 3, et l'on constate qu'elles forment un ensemble de cycles disjoints touchant les quatre sommets du graphe initial. On peut même orienter les arêtes (en allant dans l'ordre des faces devant puis derrière) pour suivre chaque cycle dans un sens précis. Puis faisons de même en prenant le graphe correspondant aux faces droite et gauche de l'empilage. On trouve aussi un ensemble de cycles disjoints touchant les quatre sommets du graphe. On constate aussi que le premier et le deuxième sous-graphe n'ont aucune arête numérotée en commun. Cela se comprend aussitôt. Si les deux sous-graphes avaient une même arête numérotée, cela signifierait que le même cube aurait ses deux paires de faces devant-derrière et droite-gauche confondues, ce qui est évidemment impossible.

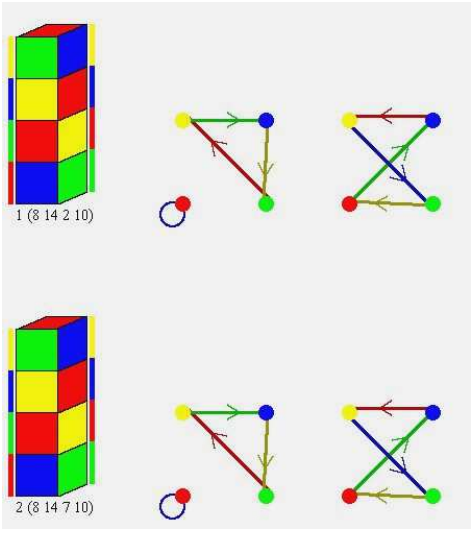
Reste à comprendre pourquoi, pour un couple de faces opposées (par exemple devant puis derrière), on a un ensemble de cycles disjoints. Imaginons que l'on ait l'ensemble des deux cycles (0 1 3) (2) ainsi disposé. En suivant un cycle lorsqu'il n'est pas réduit à une boucle, on s'aperçoit que la couleur de chaque face devant est aussi la couleur d'une face derrière pour un autre cube, et vice versa, ce qui correspond à une solution du problème, sous réserve que l'on fasse de même avec le couple des faces droite-gauche.

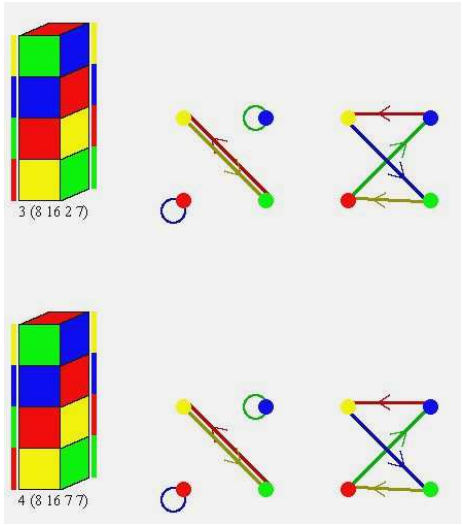


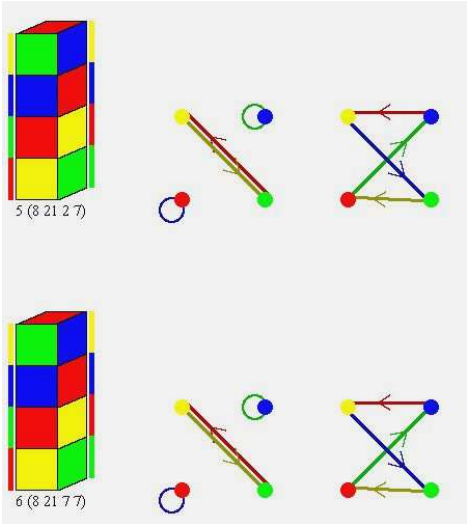


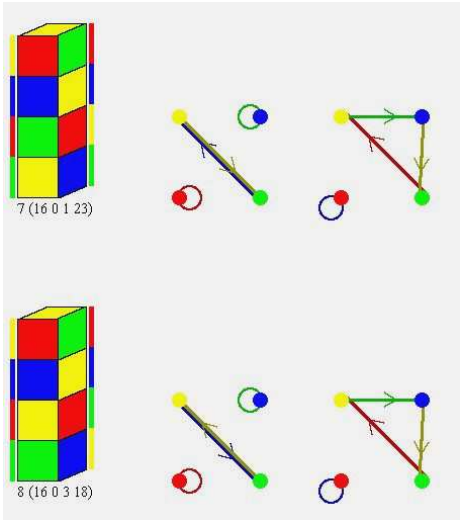
On peut vérifier tout cela avec les 28 solutions de l'exemple choisi. Le programme graphique correspondant découle du programme de calcul précédent, et nous ne le donnons pas ici, vu sa longueur.

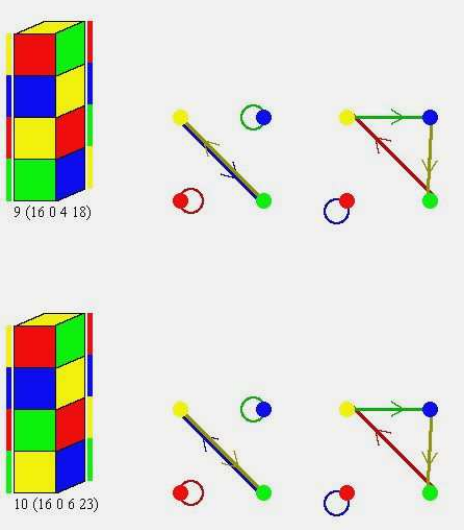
Toutes les solutions

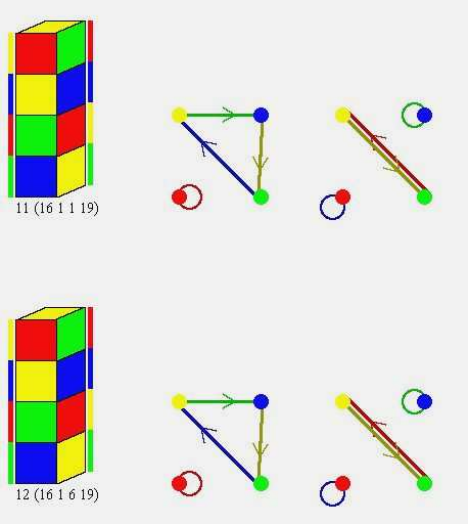


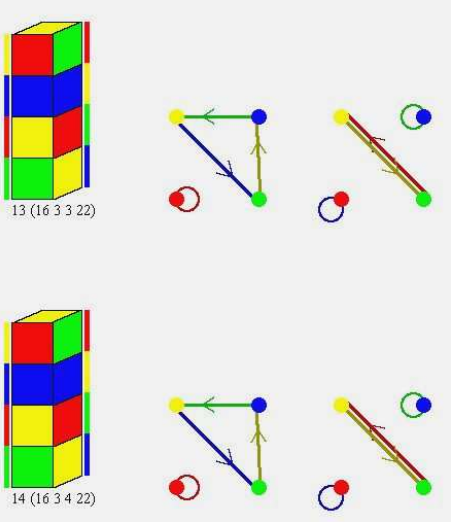


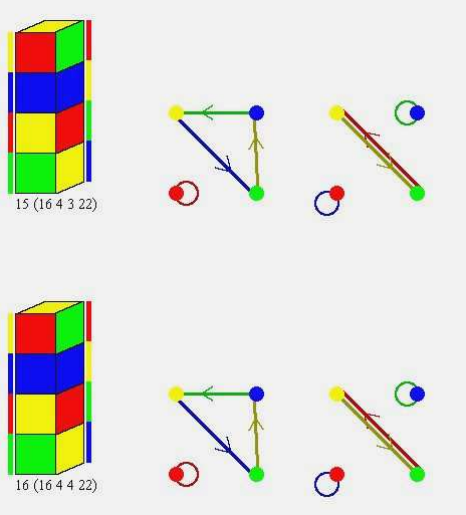


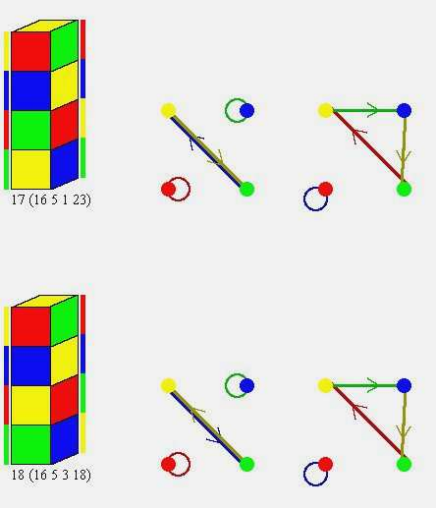


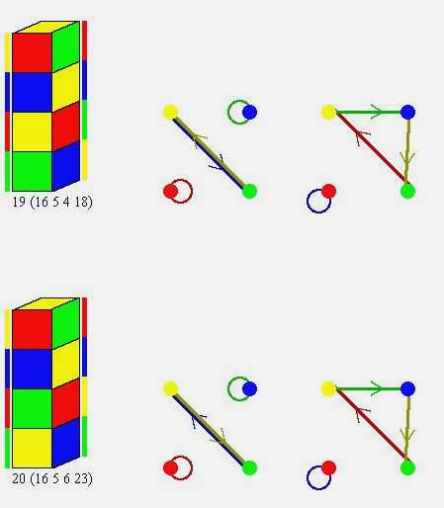

9 (16 0 4 18)
10 (16 0 6 23)

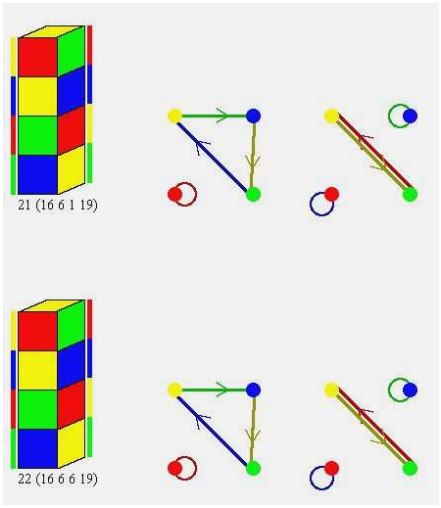

11 (16 1 1 19)
12 (16 1 6 19)

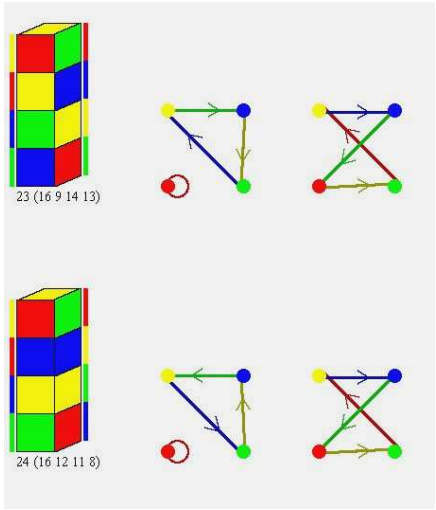

13 (16 3 3 22)
14 (16 3 4 22)

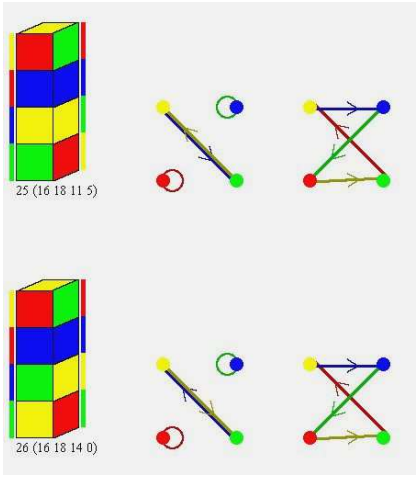

15 (16 4 3 22)
16 (16 4 4 22)

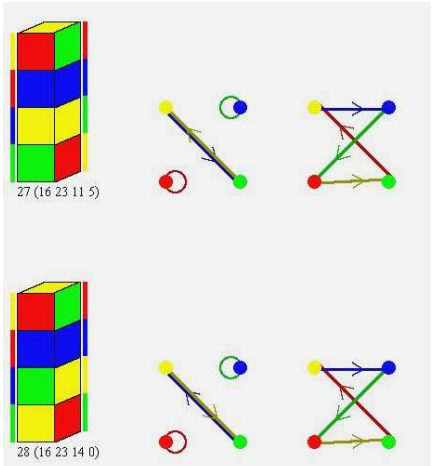

17 (16 5 1 23)
18 (16 5 3 18)


19 (16 5 4 18)
20 (16 5 6 23)







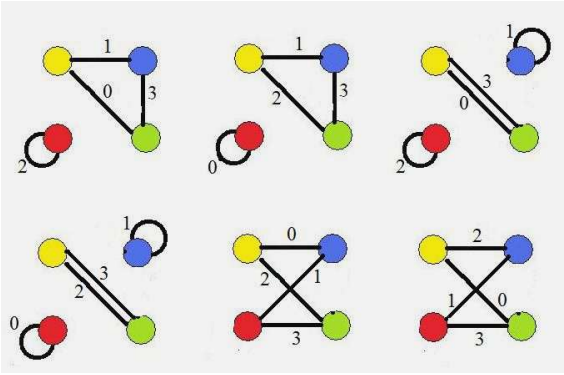


On remarque que de nombreuses solutions sont en fait identiques, notamment celles qui ont les deux mêmes factorisations, quel que soit l'ordre dans lequel on les choisit (cela revient à tourner l'empilage de 90°), et quel que soit le sens des cycles (un changement de sens revient à un retournement des cubes). Nous sommes maintenant en mesure d'élaborer la théorie du jeu.

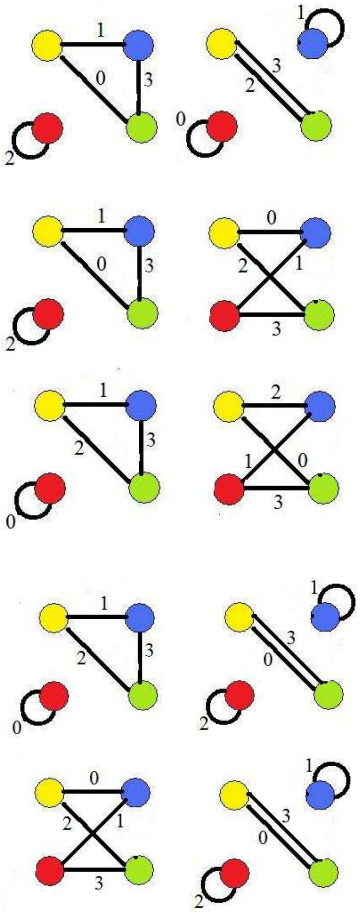
Approche théorique du problème

- On commence par tracer le graphe associé aux quatre cubes coloriés choisis.

- On détermine tous les ensembles de cycles disjoints recouvrant le graphe (tous les sommets sont atteints), avec les quatre numéros d'arêtes présents. Cela s'appelle une factorisation numérotée du graphe. Dans l'exemple précédent, on trouve exactement six ensembles de cycles disjoints numérotés couvrants, au sens près d'orientation possible des arêtes :



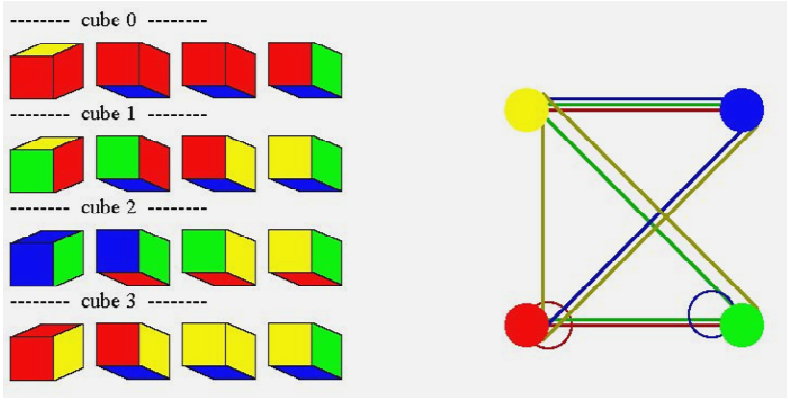
- On prend toutes les paires de factorisations n'ayant aucune arête numérotée identique. Dans l'exemple choisi, on trouve 5 possibilités :



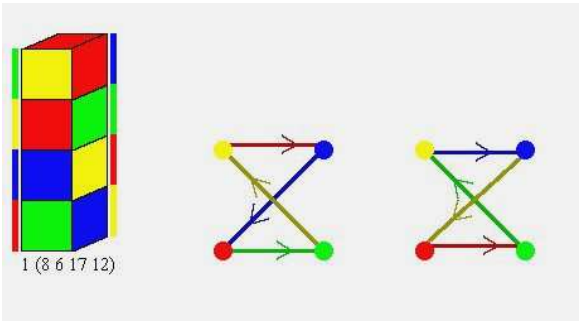
Cela donne cinq solutions distinctes du problème.

Conclusion

En choisissant des coloriages des cubes au hasard, la probabilité est faible d'avoir une solution. On obtient exactement une solution lorsqu'il existe deux factorisations numérotées sans arête en commun. Comme dans le cas classique suivant :



Il existe trois factorisations numérotées du graphe, mais seule une paire parmi elles a ses factorisations incompatibles (pas d'arête numérotée commune), d'où une solution :



Référence bibliographique :

A. Tucker, *Applied Combinatorics*, John Wiley & sons, 2002.