Technologia Więzów
Lista 3

Przemysław Kobylański

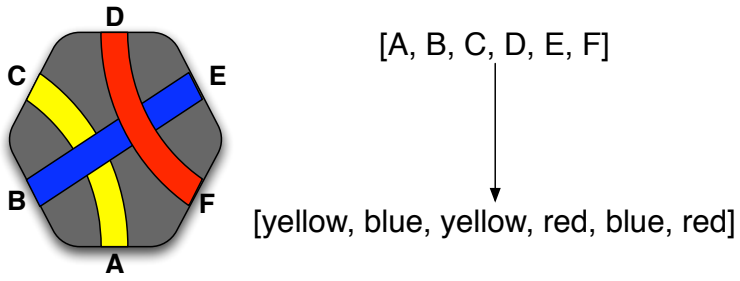
Wybierz jeden z poniższych projektów i wykonaj go korzystając z IBM ILOG Solver albo z jednego z Prologów.

1 Projekt (TANTRIX)

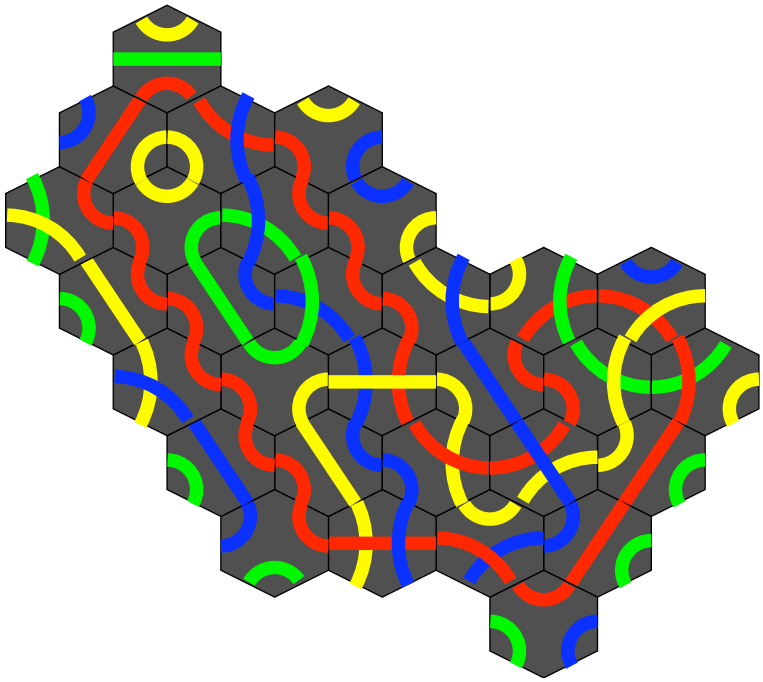
Poniżej zamieszczono fakty opisujące pierwszych trzydzieści płytek z gry TANTRIX, przy czym kolory łuków znajdujących się na płytce opisano zgodnie z zasadą przedstawioną na rys. 1.

```
tile( 1, [blue,red,blue,red,yellow,yellow], yellow).
tile( 2, [blue,red,red,blue,yellow,yellow], yellow).
tile( 3, [blue,blue,yellow,yellow,red,red], yellow).
tile( 4, [blue,red,yellow,blue,yellow,red], red).
tile( 5, [red,blue,blue,red,yellow,yellow], red).
tile( 6, [yellow,blue,red,yellow,red,blue], blue).
tile( 7, [yellow,red,yellow,red,blue,blue], blue).
tile( 8, [red,yellow,red,yellow,blue,blue], blue).
tile( 9, [red,blue,yellow,red,yellow,blue], yellow).
tile(10, [red,blue,red,blue,yellow,yellow], red).
tile(11, [blue,yellow,blue,yellow,red,red], red).
tile(12, [yellow,blue,yellow,blue,red,red], yellow).
tile(13, [yellow,red,red,yellow,blue,blue], blue).
tile(14, [blue,blue,red,red,yellow,yellow], blue).
tile(15, [red,yellow,yellow,red,green,green], red).
tile(16, [yellow,red,red,yellow,green,green], red).
tile(17, [green,red,green,red,yellow,yellow], yellow).
tile(18, [red,green,red,green,yellow,yellow], red).
tile(19, [green,yellow,green,yellow,red,red], red).
tile(20, [yellow,green,yellow,green,red,red], yellow).
tile(21, [red,red,yellow,yellow,green,green], yellow).
tile(22, [green,red,red,green,yellow,yellow], yellow).
tile(23, [red,red,green,green,yellow,yellow], yellow).
tile(24, [blue,red,red,blue,green,green], red).
tile(25, [red,red,blue,blue,green,green], red).
tile(26, [green,red,red,green,blue,blue], red).
tile(27, [blue,green,blue,green,red,red], red).
tile(28, [blue,blue,red,red,green,green], red).
tile(29, [green,blue,green,blue,red,red], red).
tile(30, [red,blue,blue,red,green,green], red).
```

Płytki układa się obracając je dowolnie i dostawiając do wcześniej ułożonych, przy czym trzeba zachować zgodność kolorów łuków dochodzących do stykających się krawędzi płytek (jak na rys. 2).



Rysunek 1: Opis kolorów na płytce.



Rysunek 2: Jedno z wielu rozwiązań dla trzydziestu płytek.

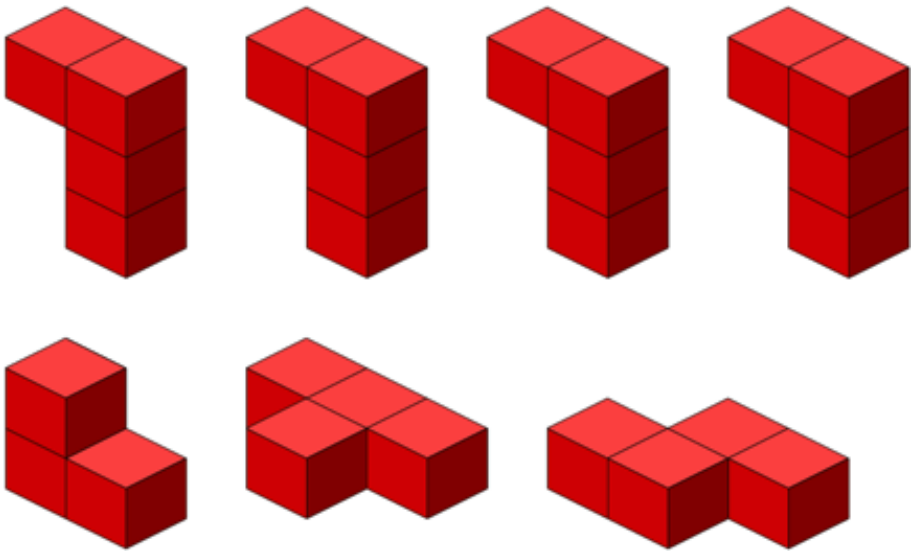
Gra TANTRIX Discovery (odkrywanka) polega na układaniu coraz dłuższych pętli zadanego koloru, przy czym pętla nie powinna zawierać wewnątrz dziury (obszaru wolnego od płytek).

Początkowo z płytek o numerach 1, 2 i 3 układa się żółtą pętlę. Następnie bierze się płytkę o numerze 4 i należy z płytek 1, 2, 3 i 4 zbudować pętlę w kolorze odpowiadającym płytce 4 (jest nim kolor czerwony podany w faktach na trzeciej pozycji), przy czym wcześniej ułożoną pętlę żółtą można zniszczyć.

Postępuje się tak z kolejnymi płytkami aż do płytki o numerze 30. Na rys. 2 przedstawiono jedną z możliwych pętli koloru czerwonego zbudowanych z płytek od 1 do 30.

Napisz program , który dla jak największego $n \in \{3, 4, \ldots, 30\}$ ułoży z płytek o numerach od 1 do n pętlę w kolorze odpowiadającym płytce o numerze n.

Uwaga: na ocenę 3.0 program musi rozwiązać wszystkie przypadki do $n \leq 10$, na ocenę 4.0 program musi rozwiązywać wszystkie przypadki do $n \leq 20$, na ocenę 5.0 program musi rozwiązywać wszystkie przypadki do $n \leq 30$.



Rysunek 3: Zestaw siedmiu klocków

2 Problem (kostka)

Na rysunku 3 przedstawiono zestaw klocków. Napisz program, który umieszcza wszystkie siedem klocków w sześcianie $3 \times 3 \times 3$.

3 Problem (sieć więzów)

Napisz funkcje w C (ew. klasy w C++), które budują sieć więzów $\mathcal{R} = (X, D, C)$ oraz są implementacjami omówionych na wykładzie algorytmów narzucających zgodność łukową i ścieżkową.

Biblioteka powinna zawierać również implementację przeszukiwania tak aby program mógł znajdować rozwiązania CSP.

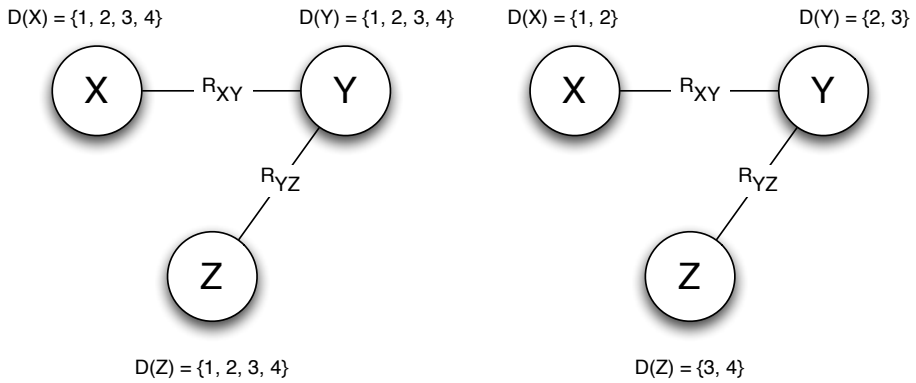
Rozpatrzmy problem znalezienia rozwiązania spełniającego ograniczenia:

$$\begin{cases} X, Y, Z \in \{1, 2, 3\}, \\ X < Y, \\ Y < Z. \end{cases}$$

Na rys. 4 przedstawiono odpowiednią sieć więzów przed i po propagacji ograniczeń, natomiast na rys. 5 przedstawiono w pseudokodzie jak mógłby wyglądać program budujący sieć i znajdujący rozwiązania.

Program z rys. 5 powinien wypisać trzy rozwiązania `<1,2,3>, <1,2,4>, <1,3,4>, <2,3,4>`.

W jakim czasie twoja biblioteka zbuduje sieć i znajdzie wszystkie 92 rozwiązania dla problemu 8 hetmanów? Czy zastosowałeś odpowiednie struktury danych by przyspieszyć propagację ograniczeń i przeszukiwanie?



Rysunek 4: Sieć a) przed i b) po propagacji ograniczeń.

```
Domain d1 = {1, 2, 3, 4}, d2 = {1, 2, 3, 4}, d3 = {1, 2, 3, 4};
Relation r = {<1,2>, <1,3>, <1,4>, <2,3>, <2,4>, <3,4>};
Network n;
add_node(1, d1);
add_node(2, d2);
add_node(3, d3);
add_edge(1, 2, r);
add_edge(2, 3, r);
Search s;
init_search(s, n);
Solution x;
while(next_solution(s, x)) print_solution(x);
```

Rysunek 5: Budowa sieci więzów i poszukiwanie rozwiązania.