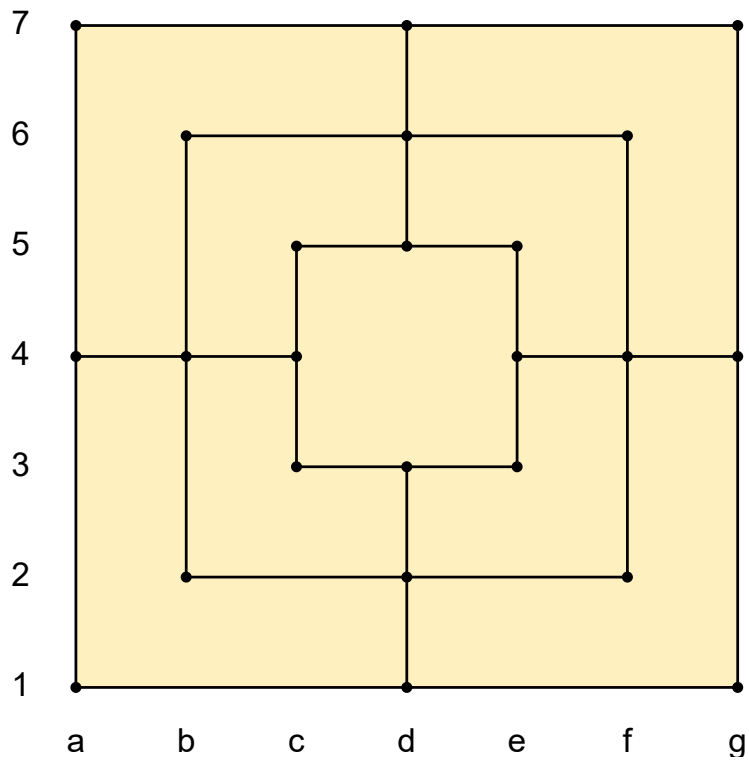
Młynek

Zasady gry i sposób reprezentacji pozycji

7 listopada 2024

Plansza

- Gra toczy się na planszy złożonej z 24 pól.
- Na początku każdy gracz dysponuje dziewięcioma pionkami.
- Celem gry jest zdobycie siedmiu pionków przeciwnika lub uniemożliwienie mu ruchu.



Etapy gry

Gra składa się z trzech etapów:

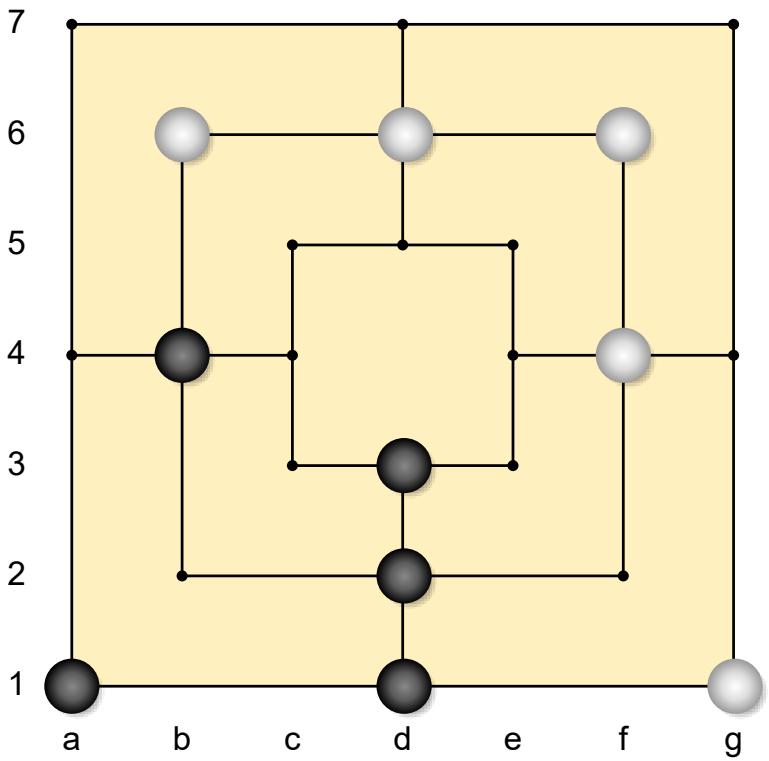
I Gracze naprzemiennie rozmieszczają swoje pionki na planszy. Jeżeli któryś gracz zbuduje „młynek” (układ trzech pionków w wierszu lub kolumnie), to zabiera przeciwnikowi pionka, który nie wchodzi w skład młynka (jeżeli wszystkie pionki przeciwnika tworzą młynki, to wtedy może zabrać dowolny z nich).

II Po wyłożeniu pionków gracze naprzemiennie wykonują ruchy swoimi pionkami. Pionki mogą przemieszczać się po liniach do sąsiednich niezajętych pól. Ponownie można budować młynki.

III Jeżeli przeciwnik pozostał z trzema pionkami, to na tym etapie jego pionki mogą skakać na dowolne pole. Tu także można budować młynki.

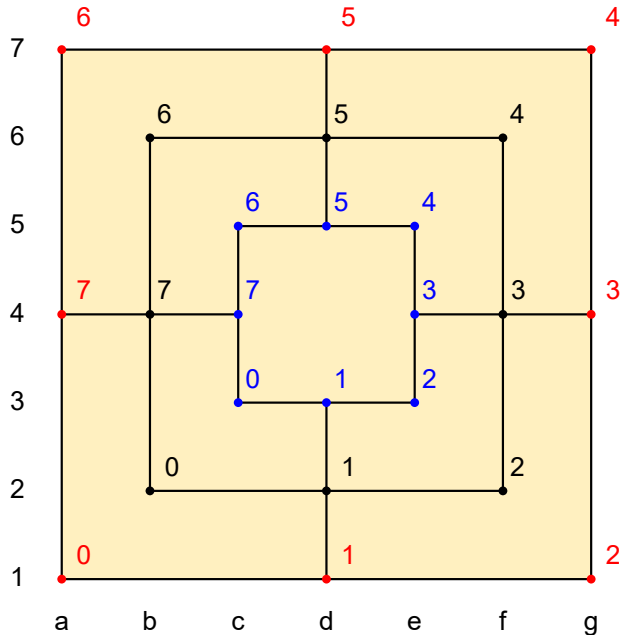
Przykład

Rysunek: Białe umieściły pionka na polu `b6`, tworząc młynek, i dzięki temu mogą zabrać czarnego pionka z pola `a1` lub `b4` — czarne pionki tworzące młynek są nietykalne.



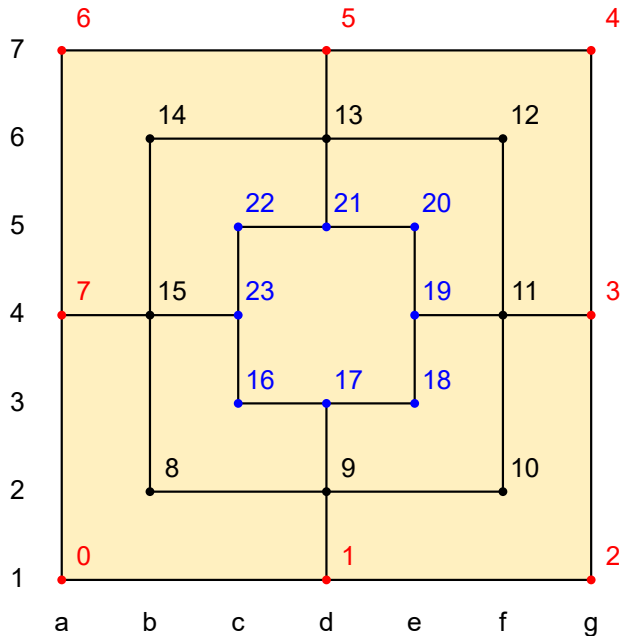
Tablica dwuwymiarowa

Plansza składa się z trzech kwadratów po osiem pól, zatem można ją reprezentować jako tablicę dwuwymiarową `[3][8]`, gdzie pierwszy indeks to numer kwadratu, a drugi to indeks pola. Na polu o indeksie parzystym można iść tylko do przodu ($j+1 \mod 8$) lub do tyłu ($j-1 \mod 8$), gdzie j to indeks pola (drugi indeks tablicy), natomiast na polu o indeksie nieparzystym można poruszać się też między kwadratami.



Tablica jednowymiarowa

Idąc krok dalej tablicę dwuwymiarową można zastąpić tablicą jednowymiarową. Tutaj pomocna w generowaniu stanów potomnych może okazać się lista sąsiedztwa każdego pola w postaci tablicy `int[24][]`, np. dla pola `17` mamy trzech sąsiadów {`9`,`16`,`18`}, dla pola `0` mamy dwóch sąsiadów {`1`,`7`} itd. Alternatywnie można zauważyć, że pola w środkowym kwadracie mają indeks o 8, a w wewnętrznym o 16 większy od indeksów pól w zewnętrznym kwadracie.



Lista figur

Zamiast trzymać całą planszę, można przechowywać jedynie położenie figur obu przeciwników (np. w postaci dwóch kontenerów `HashSet`), co może zaoszczędzić pamięć, jednakże odradzam Państwu ten sposób przechowywania, bo w tym przypadku może być problematyczne obliczanie wartości funkcji mieszającej (wyniku metody `hashCode`) niż w przypadku tablicy.

W przypadku skrajnym położenie figur można zakodować na pojedynczych bitach, czyli dwie liczby całkowite o długości trzydziestu dwu bitów wystarczyłyby do przechowywania położenia figur obu przeciwników (tzw. reprezentacja bitmapowa).

Ruch jako napis

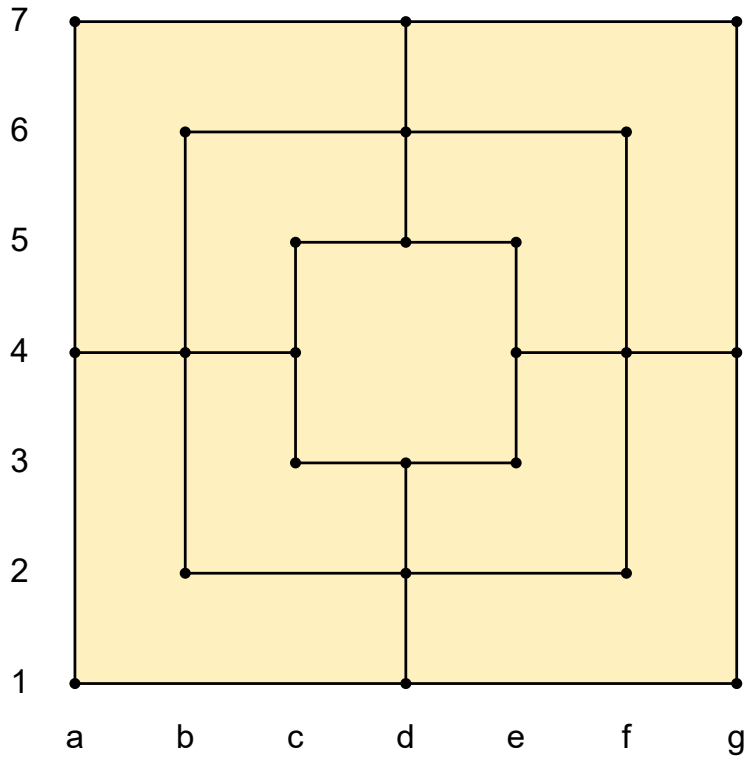
Ruch można zakodować za pomocą napisu złożonego z:

- jednej liczby — w ten sposób wskażemy pole, na które chcemy położyć pionka (na pierwszym etapie gry),
- dwóch liczb — w ten sposób wskażemy pole, na który chcemy położyć pionka i pole, z którego chcemy zabrać pionka przeciwnika (gdy utworzyliśmy młynek na pierwszym etapie gry) **lub** wskażemy pole, z którego chcemy ruszyć i na które chcemy przemieścić naszego pionka (na drugim etapie gry),
- trzech liczb — w ten sposób wskażemy pole, z którego chcemy ruszyć i na które chcemy przemieścić naszego pionka oraz pole, z którego chcemy zabrać pionka przeciwnika (gdy utworzyliśmy młynek).

Przykład I

Liczba stanów na poszczególnych poziomach drzewa, zaczynając od podanej planszy.

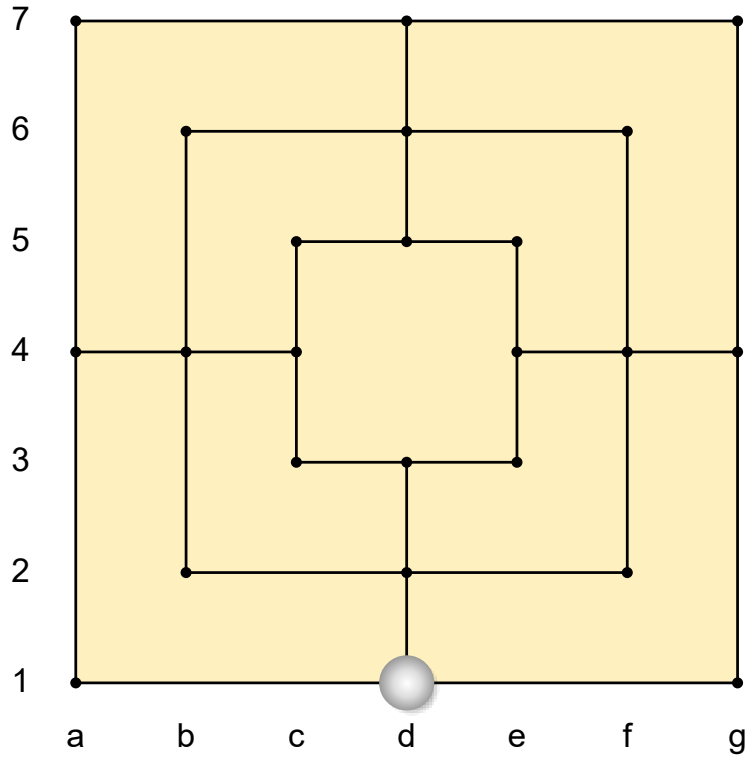
głębokość	liczba stanów
1	24
2	552
3	12144
4	255024
5	5140800
6	99274176



Przykład II

Liczba stanów na poszczególnych poziomach drzewa, zaczynając od podanej planszy (ruch czarnych).

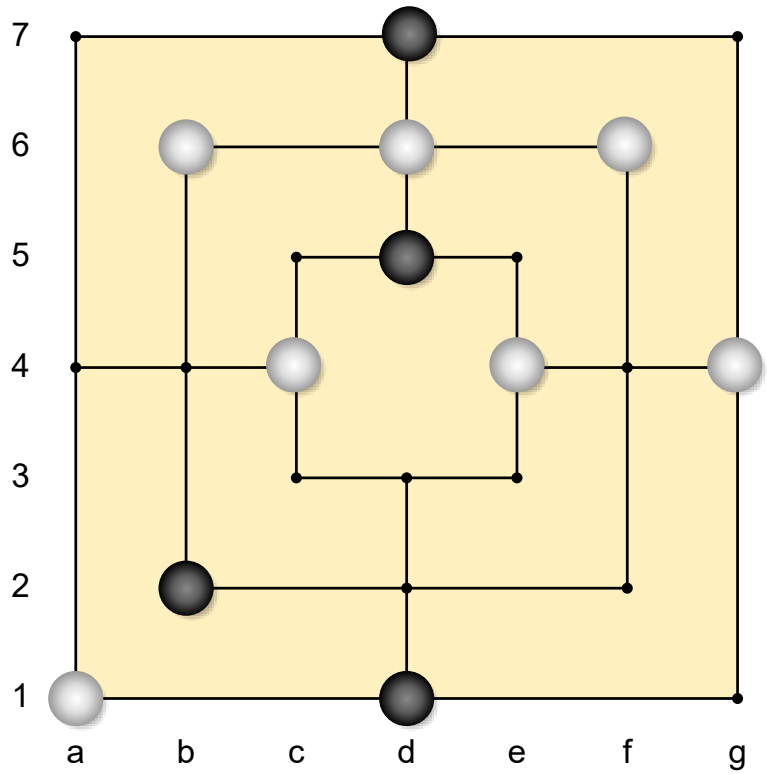
głębokość	liczba stanów
1	23
2	506
3	10626
4	214200
5	4136424
6	78065088



Przykład III

Liczba stanów na poszczególnych poziomach drzewa, zaczynając od podanej planszy.

głębokość	liczba stanów
1	15
2	251
3	3509
4	100781
5	1507836
6	49700133



Ogólne uwagi

- Klasa reprezentująca stan gry ma dziedziczyć po klasie `sac.game.GameStateImpl` i należy zaimplementować metody `hashCode`, `generateChildren`, `toString`.
- Ruch gracza koduje flaga `maximizingTurnNow` — proszę pamiętać o przełączaniu jej w stanach potomnych.
- W stanach potomnych proszę pamiętać o ustawieniu nazwy ruchu (metoda `setMoveName`), by później łatwiej znajdować szukany stan na podstawie ruchu.
- Przygotuj klasę oceniającą pozycję (dziedziczącą po klasie `StateFunction`) — w wersji podstawowej niech wyłapuje stany wygrane lub przegrane, a dla zwykłego stanu niech zwraca różnicę liczby pionków białych i czarnych.

`https://wikizmsi.zut.edu.pl/wiki/SI/L/z2`

Szkic klasy

```
class ... extends GameStateImpl {
    // plansza jako tablica jedno/dwuwymiarowa

    // być może liczba pionków każdego z graczy

    // liczba pionków do rozmieszczenia:
    // != 0 - pierwszy etap
    // == 0 - drugi/trzeci etap
    // na początku = 18 i z każdym ruchem "--"
}
```

Szkic metody grającej

```
GameState gra = new ...();
GameSearchAlgorithm alg = new ...();
String ruch;
while (!gra.isWinTerminal() && !gra.isNonWinTerminal()) {
    List<GameState> children = gra.generateChildren();
    // wczytaj ruch z System.in za pomocą obiektu klasy Scanner
    for (GameState c : children)
        if (ruch.equals(c.getMoveName())) {
            gra = c;
            break;
        }
    // w razie podania błędnego ruchu powtórz wczytanie

    if (gra.isWinTerminal() || gra.isNonWinTerminal())
        break;
    children = gra.generate_children();
    alg.setInitial(gra);
    alg.execute();
    ruch = alg.getFirstBestMove();
    // wykonać ruch gracza komputerowego analogicznie jak poprzednio
}
```