Niżej są dwa programy wyliczające, który z graczy ma strategię zwycięską w grze, w której dwaj gracze na przemian wykonują ustaloną liczbę ruchów. Każdy ruch polega na wybraniu jednego ze skończenie wielu elementów. W ten sposób gracze tworzą skończony ciąg pewnych elementów. Jeżeli utworzony w ten sposób ciąg spełnia pewien warunek W, to wygrał I gracz (gracz rozpoczynający grę). W przeciwnym razie zwyciężył II gracz.

Wykorzystywane są następujące oznaczenia i deklaracje zmiennych:

1) n oznacza liczbę ruchów wykonywanych przez każdego z graczy,

2) `T` jest tablicą indeksowaną od 0 liczbami mniejszymi od $2n$, w tej tablicy są zapisywane elementy wybierane przez graczy, na pozycjach parzystych elementy wybierane przez I gracza, a na nieparzystych – przez II. Elementy tej tablicy mogą być też puste, a więc mogą niczego nie zawierać.

1 Stosowana terminologia

Informacje `T[0]`, `T[1]`, ..., `T[ruch - 1]` wraz z wartością zmiennej `ruch` nazywamy pozycją (w rozważanej grze). Jest to pozycja I gracza, jeżeli wartość `ruch` jest parzysta. W pozycji I gracza ruch wykonuje gracz I, ruch polega na określeniu wartości `T[ruch]`. Analogicznie definiujemy pozycję II gracza.

Pozycja (dowolna, I lub II gracza) jest pozycją zwycięską (lub wygrywającą) I gracza, jeżeli w tej pozycji ma on strategię zwycięską, czyli wygrywającą. Analogicznie definiujemy pozycję zwycięską II gracza.

Żadna pozycja nie może być zwycięska dla obu graczy. W przeciwnym razie mogliby grać przeciwko sobie strategiami zwycięskimi i utworzyliby w ten sposób ciąg, który jednocześnie miałby własność W i nie mógłby jej mieć. Z drugiej strony w każdej pozycji jeden z graczy ma strategię zwycięską. Stwierdzenie, że I gracz nie ma strategii zwycięskiej, na mocy prawa de Morgana oznacza, że strategię zwycięską ma gracz II.

2 Program rekurencyjny

Zdefiniujemy funkcję boolowską `SW_I(ruch)`, która przyjmuje wartość `true` wtedy i tylko wtedy, gdy w dalszej grze, po wybraniu przez graczy elementów `T[0]`, `T[1]`, ..., `T[ruch - 1]`, pierwszy gracz ma strategię zwycięską.

Funkcja ta jest następująca i wymaga komentarza.

```
function SW_I(ruch): boolean;
begin
T[ruch] := pierwszy;
if ruch = 2n
then return (T ∈ W)
else if ruch ma wykonać I gracz
    then begin
      while (T[ruch] <> pusty) and not SW_I(ruch+1) do
        T[ruch] := kolejny(T[ruch]);
      return (T[ruch] <> pusty) end
    else begin
      while (T[ruch] <> pusty) and SW_I(ruch+1) do
        T[ruch] := kolejny(T[ruch]);
      return (T[ruch] = pusty) end
end.
```

Przyjmujemy, że elementy zapisywane w tablicy `T` są uporządkowane i jest ich skończenie wiele. Słowo `pierwszy` w programie oznacza stałą, której wartością jest najmniejszy w sensie wybranego porządku element zapisywany w `T`. Analogiczne znaczenie ma słowo `ostatni`. Z kolei słowo `pusty` może być interpretowane jako element, który może być zapisywany w tablicy `T`, ale nie jest używany przez graczy w trakcie gry. W programie występuje jeszcze funkcja `kolejny`, która dla danego elementu tablicy wylicza element następny w sensie wykorzytywanego porządku. Dodatkowo zostało przyjęte, że `kolejny(ostatni) = pusty`. W końcu warunek `ruch ma wykonać gracz I` może zostać zdefiniowany jako stwierdzenie, że zmienna `ruch` przyjmuje wartość parzystą.

W programie występują dość niejasne instrukcje **return** `(T[ruch] <> pusty)` i podobna z równością. Instrukcje te każą zwrócić wartość logiczną podanych wyrażeń boolowskich. Trzeba zauważyć, że wyjście z pierwszej instrukcji **while** ma miejsce w dwóch sytuacjach: albo gdy zachodzi warunek `T[ruch] <> pusty` i równocześnie `SW_I(ruch+1) = true`, a więc, gdy `T[ruch]` zawiera zwycięskie posunięcie I gracza, albo też gdy `T[ruch] = pusty` i nie udało się znaleźć zwycięskiego posunięcia I gracza. Tak więc w obu przypadkach wyrażenie `T[ruch] <> pusty` ma taką samą wartość logiczną, jak stwierdzenie o istnieniu strategii zwycięskiej I gracza. Podobna sytuacja ma miejsce w przypadku drugiej instrukcji **while**.

3 Wersja iteracyjna programu

Poniższa funkcja ma zwrócić wartość `true` wtedy i tylko wtedy, gdy I gracz ma strategię wygrywającą w rozważanej grze z testem `W` określającym, który z graczy zwyciężył.

```
function SW_I: boolean;
var pozW_I: boolean; ruch: integer;
begin
  poz_W := false;
  ruch := 0;
  while ruch ⩾ 0 do
  begin
    if ruch = 2n
    then begin pozW_I := (T ∈ W); ruch := ruch - 1 end
    else if T[ruch] = pusty
      then begin T[ruch] := pierwszy; ruch := ruch + 1 end
      else if ((ruch II gracza) = pozW_I) and (T[ruch] <> ostatni)
        then
        begin T[ruch] := kolejny(T[ruch]); ruch := ruch + 1 end
        else begin T[ruch] := pusty; ruch := ruch - 1 end
  end;
  return pozW_I
end.
```

Napis `(T ∈ W)` oznacza wartość boolowską `true` wtedy i tylko wtedy, gdy rozgrywce opisanej w tablicy `T` wygrał gracz I. Podobnie, napis `(ruch II gracza)` to wyrażenie boolowskie przyjmujące wartość `true` wtedy i tylko wtedy, gdy wartość zmiennej `ruch` jest nieparzysta.

W tej chwili problem polega na wykazaniu poprawności podanej funkcji. Oczywiście, najłatwiej to zrobić metodą niezmienników. Najpierw jednak zdefiniujemy pomocnicze pojęcia.

Przypuśćmy, że mamy daną pozycję `T[0], T[1], ... , T[ruch]`. Wtedy ważną pozycją wcześniejszą nazywamy każdą pozycję postaci `T[0], T[1], ... , T[i-1], x` dla wszystkich `i` $\leqslant$ `ruch` i wszystkich `x` $<$ `T[i]` w sensie ustalonego porządku elementów wpisywanych do tablicy `T`. Przyjmujemy dodatkowo, że jeżeli `T[ruch] = pusty` lub jest nieokreślony, to pojęcie ważnych pozycji wcześniejszych też jest definiowane, ale nie ma `x` takich, że `x` $<$ `T[ruch]`.

O tak rozumianych pozycjach będziemy mówić, że są przegrywającymi, jeżeli gracz, który utworzył tę pozycję (wpisał do tablicy `x`) nie ma w niej strategii zwycięskiej.

Głównym niezmiennikiem pętli z funkcji `SW_I` jest stwierdzenie, że

1) wszystkie ważne pozycje wcześniejsze od `T[0], T[1], ... , T[ruch]` są przegrywające oraz

2) jeżeli wartość `T[ruch]` jest niepusta, to I gracz ma strategię wygrywającą w pozycji `T[0], T[1], ... , T[ruch]` wtedy i tylko wtedy, gdy zmienna `pozW_I` ma wartość `true`.