Zadania na konkurs próbny

Piotr Zieliński

24 maja 2001 roku

Zadanie A – Gra w kapsle

Wejście: `A.IN`
Wyjście: `A.OUT`
Program: `A.PAS`, `A.C` lub `A.CPP`

W grze w kapsle bierze udział dwóch zawodników. Każdy z nich posiada własny kapsel. Zawodnicy na zmianę przemieszczają swoje kapsle. Celem gry jest trafienie w kapsel przeciwnika (czyli umieszczenie swojego kapsla w takim miejscu, aby stykał się z kapslem przeciwnika). Ten, komu się to uda, wygrywa.

Przemieszczaniem kapsli rządzą oczywiście pewne reguły. Dla uproszczenia analizy przyjmijmy jednak, iż zawodnik może przesunąć swój kapsel w dowolne miejsce, tak jednak, aby odległość środka kapsla po przesunięciu od środka kapsla przed przesunięciem była nie większa niż pewna stała d (różna dla każdego gracza).

Aby uściślić problem do końca zakładamy, że kapsle są kołami (domkniętymi) o promieniach odpowiednio r_1 i r_2 oraz maksymalne odległości na jakie mogą być przesunięte (ich środki) to d_1 i d_2. Gra odbywa się na nieograniczonej płaszczyźnie. Początkowo środek kapsla zawodnika nr 1 znajduje się w punkcie (x_1, y_1), a środek kapsla zawodnika nr 2 znajduje się w punkcie (x_2, y_2). Można założyć, że na początku kapsle ani się nie pokrywają, ani się nie stykają. Pierwszy ruch wykonuje zawodnik nr 1.

Twoim zadaniem jest napisanie programu, który dla zadanych parametrów (r_1, d_1, x_1, y_1, r_2, d_2, x_2, y_2) stwierdzi, czy dla któregoś gracza istnieje strategia wygrywająca.

Wejście. W pierwszej linii pliku wejściowego znajduje się liczba naturalna d ($1 \leqslant d \leqslant 10$), określająca liczbę zestawów danych, których opisy umieszczone są kolejno po sobie w następnych liniach pliku. Opis pojedynczego zestawu jest następujący:

W pierwszej i jedynej linii znajduje się osiem liczb całkowitych: r_1, d_1, x_1, y_1, r_2, d_2, x_2, y_2 ($1 \leqslant r_1, d_1, r_2, d_2 \leqslant 100$, $-1000 \leqslant x_1, y_1, x_2, y_2 \leqslant 1000$).

Wyjście. Każdemu zestawowi w pliku wejściowym powinna odpowiadać jedna linia pliku wyjściowego. Linia ta powinna zawierać jedną liczbę całkowitą określającą numer gracza, dla którego istnieje strategia wygrywająca

(tj. 1 lub 2) lub słowo `NIE`, jeśli dla żadnego z nich takowa strategia nie istnieje.

Przykład

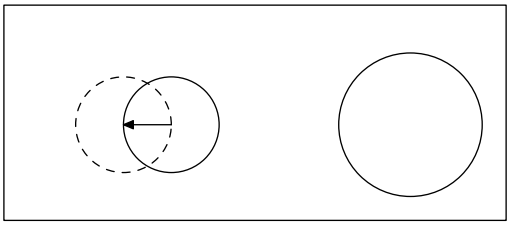
Wejście – `A.IN`

```
3
2 20 0 0 2 15 100 100
2 20 0 0 2 5 100 100
2 2 0 0 3 5 0 10
```

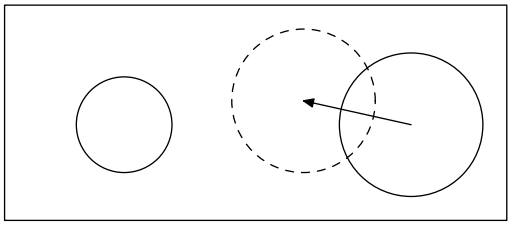
Wyjście – `A.OUT`

```
NIE
1
2
```

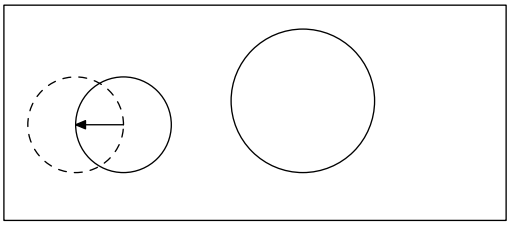
Możliwy przebieg gry dla sytuacji z ostatniego zestawu przedstawiają poniższe rysunki:



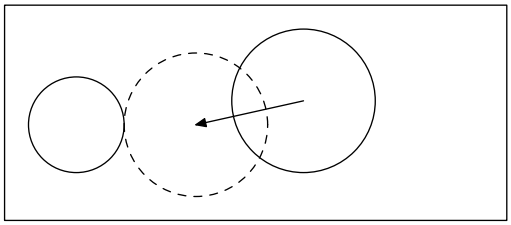
pierwszy ruch — gracz 1



drugi ruch — gracz 2



trzeci ruch — gracz 1



czwarty ruch — gracz 2 wygrywa

Zadanie B – Obchodzenie szachownicy

Wejście:	`B.IN`
Wyjście:	`B.OUT`
Program:	`B.PAS`, `B.C` lub `B.CPP`

Mamy daną szachownicę o wymiarach $n \times m$ (n kolumn, m wierszy). Naszym zadaniem jest, startując z lewego górnego rogu szachownicy, obejść wszystkie pola szachownicy dokładnie raz, po czym powrócić na pole startowe. Z każdego pola szachownicy można wejść na jedno z sąsiednich pól w pionie lub w poziomie (przechodzenia na ukos są zabronione).

Twoim zadaniem jest napisanie programu, który dla danych wymiarów $n \times m$ szachownicy stwierdzi, czy „obchód" spełniający powyższe wymagania w ogóle istnieje. Jeśli tak, to powinien dodatkowo wypisać jedną z takich tras. Lista odwiedzonych pól powinna się zaczynać i kończyć lewym górnym rogiem szachownicy i mieć dokładnie $nm+1$ elementów. Każde pole szachownicy jest opisywane za pomocą pary punktów (x, y), gdzie x jest numerem kolumny ($1 \leqslant x \leqslant n$), a y – numerem wiersza ($1 \leqslant y \leqslant m$). Lewy górny róg szachownicy ma współrzędne $(1, 1)$.

Wejście. W pierwszej linii pliku wejściowego znajduje się liczba naturalna d ($1 \leqslant d \leqslant 10$), określająca liczbę zestawów danych, których opisy umieszczone są kolejno po sobie w następnych liniach pliku. Opis pojedynczego zestawu jest następujący:

W pierwszej i jedynej linii znajdują się dwie liczby naturalne: n i m ($2 \leqslant n, m \leqslant 500$), określające, odpowiednio, liczbę kolumn i wierszy szachownicy.

Wyjście. Każdemu zestawowi w pliku wejściowym powinna odpowiadać jedna linia pliku wyjściowego. Zawartość tej linii zależy od tego, czy dla określonej szachownicy szukana trasa istnieje, czy nie.

Jeśli trasa taka istnieje, linia powinna zawierać $2(nm + 1)$ liczb całkowitych x_1, y_1, x_2, y_2, $\ldots$, x_{nm+1}, y_{nm+1}, oddzielonych odstępami, takich że (x_i, y_i) jest i-tym polem trasy. Oczywiście $x_1 = y_1 = x_{nm+1} = y_{nm+1} = 1$.

Jeśli natomiast trasa taka nie istnieje, wówczas linia ta powinna zawierać pojedyncze słowo `NIE`.

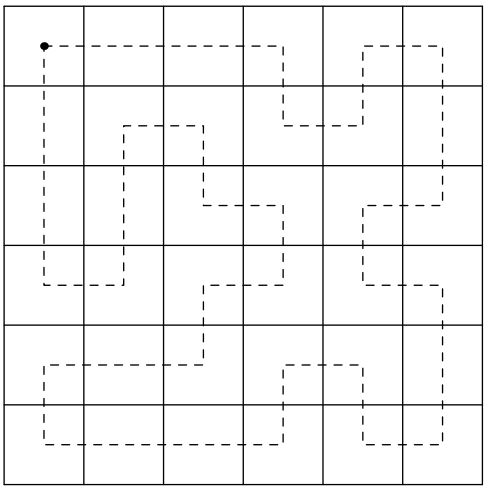
Przykład

Wejście – `B.IN`

```
3
2 3
3 3
6 6
```

Wyjście – `B.OUT`

```
1 1 1 2 1 3 2 3 2 2 2 1 1 1
NIE
1 1 2 1 3 1 4 1 4 2 5 2 5 1 6 1
6 2 6 3 5 3 5 4 6 4 6 5 6 6 6 5
5 5 4 5 4 6 3 6 2 6 1 6 1 5 2 5
3 5 3 4 4 4 4 3 3 3 3 2 2 2 2 3
2 4 1 4 1 3 1 2 1 1
```

W rzeczywistości plik wynikowy zawiera tylko 3 linie. Ostatnia, trzecia linia została, z powodu jej długości, wydrukowana w kilku wierszach. Należy również pamiętać, że jest to tylko przykładowy plik wyjściowy, istnieją inne (zawierające inne drogi), które są poprawne. Program powinien wygenerować dowolny z nich.

Przykładową drogę dla ostatniego zestawu prezentuje poniższy rysunek: