Labirynt Algorytmusa

MISJA: URATUJ LUDZKOŚĆ PRZED ZAGŁADĄ

Starożytny Algorytmus władał niegdyś wszystkimi algorytmami świata. Aby jednak upewnić się, że algorytmy nie wpadną w niepowołane ręce, stworzył sieć nieskończenie długich i zawiłych korytarzy i umieścił je na ich końcu.

Przez długie wieki nikt nie odważył się wejść do labiryntu. Pewnego jednak dnia zły czarownik Hackerus opanował labirynt, przejął algorytmy i zaczął je wykorzystywać przeciwko ludzkości. Labirynt zapełnił siłami zła a same algorytmy otoczył wielką siecią drzwi i teleportów. Świat popadł w chaos.

Aby uratować świat przed zgubnym wpływem algorytmów w rękach Hackerusa musisz przejść przez sieć labiryntów omijając siły zła, szukając kluczy i teleportacji. Zadanie wymagać będzie nie lada wysiłku ale w grę wchodzi przyszłość ludzkości!

Zadanie 1: Uczymy się chodzić

OPIS ZADANIA

Twoje zadanie polegać będzie na zbudowaniu algorytmu który będzie w stanie przechodzić przez kolejne poziomy labiryntu. Aby móc przechodzić labirynty musimy najpierw nauczyć się chodzić, rozróżniać korytarze od ścian oraz wiedzieć skąd dokąd mamy dojść.
Każdy labirynt to tablica pól. Każde pole jest odpowiednio oznaczone. W pierwszych labiryntach występują następujące typy pól:

0 puste pole, korytarz, którym można iść

1 ściana przez którą nie przejdziemy

2 punkt startowy z którego zaczynamy naszą wędrówkę po określonym poziomie labiryntu

3 punkt końcowy do którego musimy dojść aby przejść na niższy poziom labiryntu

Podstawowe zasady dotyczące labiryntów oraz chodzenia po labiryntach to:

- Labirynty mogą mieć różną długość i szerokość.
- Każdy labirynt powinniśmy przejść korzystając z możliwie najkrótszej drogi.
- Dozwolone są cztery ruchy: lewo, prawo, góra, dół.
- Nie ma możliwości chodzenia na ukos.
- Tablica nie zawsze jest kwadratem (mogą się zdarzyć tablice o wymiarach np. 1x10).
- Punkt startowy traktujemy jak **0 (puste pole)** w przypadku gdybyśmy chcieli ponownie przez nie przejść.
- Jeżeli występują dwa najkrótsze rozwiązania podajemy którekolwiek z nich.

Nasz algorytm powinien możliwość czytania znaków wejściowych z konsoli, a wynikiem działania algorytmu powinna być seria znaków wyrzucona na standardowy output.

Input zawiera następujące elementy:

{nr of labirynths}
{width of the 1st labirynth} {height of the 1st labirynth }
{fields of the 1st labirynth}
{width of the 2nd labirynth} {height of the 2nd labirynth }
{fields of the 2nd labirynth }
{width of the nth labirynth} {height of the nth labirynth }
{ fields of the nth labirynth }

PRZYKŁADY

Aby lepiej zrozumieć zasadę działania algorytmu zapoznajmy się z kilkoma przykładowymi inputami i outputami które powinny zostać zwrócone przez algorytm:

Przykład 1 : Prosty labirynt 3x3

INPUT

```
1
3 3
0 0 0
0 1 0
2 1 3
```

OUTPUT

```
(0,0) (0,1) (0,2) (1,2) (2,2) (2,1) (2,0)
```

WYJAŚNIENIE

W powyższym przykładzie musimy przejść jedynie 1 labirynt, o rozmiarze 3x3.
Zaczynamy w lewym dolnym rogu, a powinniśmy skończyć w prawym dolnym. Output pokazuje drogę którą musimy przejść, z uwzględnieniem punktu startu i punktu końca.

Przykład 2: Dwa labirynty 3x3 (trywialne wyjście) oraz 5x5 (wyjście typu „ślimak”)

INPUT

```
2
3 3
1 1 1
3 1 1
2 1 1
5 5
0 0 0 0 0
0 1 1 1 0
0 1 2 1 0
0 1 0 1 0
3 1 0 0 0
```

OUTPUT

```
(0,0) (0,1)
(2,2) (2,1) (2,0) (3,0) (4,0) (4,1) (4,2) (4,3) (4,4) (3,4) (2,4)
(1,4) (0,4) (0,3) (0,2) (0,1) (0,0)
```

WYJAŚNIENIE

W powyższym przykładzie mamy 2 labirynty: pierwszy o rozmiarze 3x3, drugi o rozmiarze 5x5. Pierwszy labirynt jest niezmiernie prosty, wystarczy przejść o jedno pole.
W drugim labiryncie droga jest nieco dłuższa. Po zakończeniu pierwszego labiryntu, output powinien zawierać przejście do nowej linii. Dzięki temu można w prosty sposób rozróżnić w którym miejscu rozpoczynamy przechodzenie kolejnego labiryntu.

CO MUSI ZAWIERAĆ ROZWIĄZANIE

Rozwiązaniem zadania powinna być aplikacja konsolowa napisana w języku Java.
Rozwiązanie musi zostać przesłane w postaci spakowanego kodu źródłowego. Preferowany projekt w Maven.

Następnie zostanie ono przepuszczone automatycznie przez 30 przykładowych labiryntów, a wyniki porównane z wynikami wzorcowymi. Jeżeli we wszystkich przypadkach wyniki będą poprawne zadanie zostanie zaakceptowane.

Maksymalny rozmiar labiryntu to 1000x1000. Oznacza to iż nie istnieje możliwość wystąpienia labiryntu o szerokości np. 1001 (1001x999).
Mimo iż labirynty zawierają poprawną strukturę, warto założyć podstawową walidację danych wejściowych.

Zadanie 2: Teleportacja

Przechodząc przez kolejne poziomy labiryntów Twoim oczom ukazała się dziwna świetlista brama. Nie zastanawiając się długo wszedłeś w nią.

Przez chwilę poczułeś dziwne wibracje i zawrót głowy a przed Twoimi oczami widziałeś tylko ciemność. Nagle ciemność znikła a Ty stałeś z powrotem na korytarzu. Tyle tylko, że był to zupełnie inny korytarz niż ten na którym jeszcze przed chwilą stałeś…

OPIS ZADANIA

Zadanie „Teleportacja" wprowadza dodatkowy element do labiryntów – teleportacje. Teleportacje oznaczone są liczbami od **7** do **99** i zawsze występują parami. Wchodząc do teleportu **7**, wyjdziesz w drugim teleporcie **7**.

7-99 Teleportacje (występujące parami)

Kilka podstawowych zasad związanych z teleportami:

- Wchodząc na pole teleportacji zawsze teleportujesz się do drugiego wyjścia. Nie ma możliwości przejścia przez teleportacje nie wchodząc w nią.
- Wychodząc z drugiej strony teleportu nie ma możliwości „automatycznego" wejścia z powrotem. Oznacza to, że aby z powrotem wejść do teleportu musi najpierw wykonać ruch o jedno pole, następnie wykonać kolejny ruch wchodząc ponownie na pole teleportu.
- Nie istnieje możliwość aby wyjście teleportu było ze wszystkich stron otoczone ścianami (nie dając możliwości ruchu, ani powrotu). Oczywiście można mimo wszystko dodatkowo zabezpieczyć program przez takimi sytuacjami wyrzucając odpowiedni komunikat o błędzie.
- Nie zawsze musimy wchodzić w jakikolwiek teleport żeby znaleźć najkrótszą ścieżkę.

PRZYKŁADY

INPUT

```
2
3 3
2 0 7
1 1 1
7 0 3
4 4
9 2 1 9
1 1 1 13
13 15 1 1
1 1 15 3
```

OUTPUT

```
(0,2) (1,2) (2,2) (0,0) (1,0) (2,0)
```

```
(1,3) (0,3) (3,3) (3,2) (0,1) (1,1) (2,0) (3,0)
```

WYJAŚNIENIE
Powyższy przykład zawiera 2 labirynty (3x3 oraz 4x4). W pierwszym labiryncie występuje tylko jedna para teleportów. W labiryncie drugim występują 3 pary teleportów.

CO MUSI ZAWIERAĆ ROZWIĄZANIE

Rozwiązaniem zadania powinna być aplikacja konsolowa napisana w języku Java.
Rozwiązanie musi zostać przesłane w postaci spakowanego kodu. Preferowany projekt w Maven.

Następnie zostanie ono przepuszczone automatycznie przez 30 przykładowych labiryntów, a wyniki porównane z wynikami wzorcowymi. Jeżeli we wszystkich przypadkach wyniki będą poprawne zadanie zostanie zaakceptowane.

Maksymalny rozmiar labiryntu to 1000x1000. Oznacza to iż nie istnieje możliwość wystąpienia labiryntu o szerokości np. 1001 (1001x999).
Mimo iż labirynty zawierają poprawną strukturę, warto założyć podstawową walidację danych wejściowych.
Maksymalna ilość wystąpień teleportacji w jednym labiryncie to (99-7)*2 = 184