Jak Dojade?

autor:/author:
Krzysztof Ocetkiewicz

Dana jest prostokątna mapa podzielona na kwadraty. Każde pole mapy może być puste (nieprzejezdne), być fragmentem drogi (przejezdnej w każdą stronę) lub miastem (przejezdnym tak jak droga). Nazwa każdego z miast jest podana na mapie i każda litera zajmuje jedno pole (pole z literą jest nieprzejezdne). Przebycie jednego pola zajmuje minutę. Poruszać się możemy tylko pomiędzy polami sąsiadującymi bokami. Dodatkowo wybrane miasta połączone są jednokierunkowymi połączeniami lotniczymi.
Nazwa miasta będzie sąsiadowała (bokiem lub rogiem) z miastem pierwszą lub ostatnią literą. Nazwa będzie jednoznacznie przypisana do miasta. Pole przed i za nazwą miasta (o ile będzie takie pole) nie pojawi się cyfra lub litera.

Uwagi implementacyjne:

- Nie można używać klasy string ani kontenerów z biblioteki STL.
- Można używać elementów języka C++ takich jak klasy, referencje, itp.
- Można zaimplementować własną klasę string.
- Czytelność kodu jest ważna, a komentarze (które są oczywiście dozwolone) niekoniecznie ją zapewniają.
- Smartpointer'y z biblioteki standardowej są zabronione, ale nadal można zaimplementować własną klasę inteligentnych wskaźników.
- Proszę pamiętać o odzyskiwaniu pamięci podczas usuwania elementów implementowanych struktur danych.

A rectangular map divided into squares is given. Each field on the map can be empty (impassable), be part of a road (passable in both directions), or be a city (passable like a road). The name of each city is given on the map, and each letter occupies one field (the field with the letter is impassable). Moving through one field takes one minute. We can only move between fields adjacent by sides. Additionally, selected cities are connected by one-way air connections. The name of a city will be adjacent (by side or corner) to the city with the first or last letter of the name.
The name will be unambiguously assigned to the city. There will be no number or letter before or after the name of the city (if there is such a field).

Implementation notes:

- Standard string class and other standard library containers cannot be used.
- You can use C++ elements such as classes, references, etc.
- One can implement their own string class.
- Code readability is important, and comments (which are of course allowed) do not necessarily ensure it.
- Smartpointers from standard library are forbidden, but you can still implement your own smartpointer class.
- Please remember to free memory when deleting elements of the implemented data structures.

Przykłady:/Examples:

Tak może być:/**This can be**:

```
*GDANSK....OPOLE....RUMIA
...*GDYNIA......*....*...
*SOPOT.............*....*
...PUCK*.*KRAKOW.REDA.HEL
```

Tak nie będzie:/**This is incorrect**:

```
**GDANSK..........OPOLE.
..........*GDYNIA*......
*SOPOTHEL..............
........*..REDA.*PUCK*..
```

Z A można dotrzeć do B:/**It is possible to arrive from point A to B**:

```
..................
A*####...#####*B..
.....#.C.#........
.....##*##........
......
.A**B.
......
```

Z A nie można dotrzeć do B:/**It is not possible to arrive from point A to B**:

```
....##*B
A*##....
....##*B
A*#*C...
....*....
A*##C##*B
```

Wejście:/Input:

W pierwszej linii wejścia pojawią się wymiary mapy: szerokość *w* i wysokość *h*. W następnych *h* liniach (każda po *w* znaków) podany będzie opis mapy. Każdy znak opisu to kropka (.) oznaczająca puste pole, krzyżyk (#) reprezentujący drogę, gwiazdka (*) symbolizujący miasto lub litera bądź cyfra, będąca częścią nazwy miasta.
W kolejnej linii pojawi się jedna liczba *k* – liczba połączeń lotniczych. Następne *k* linii to opis połączeń w postaci *źródło cel czas*, gdzie *źródło* to nazwa miasta startowego, *cel* to nazwa miasta docelowego a *czas* jest czasem przelotu w minutach. W kolejnej linii pojawi się jedna liczba *q* będąca liczbą zapytań. Każde zapytanie pojawi się w osobnej linii i będzie miało format: *źródło cel typ*. Jest to zapytanie o najkrótszy czas przejazdu od miasta *źródło* do miasta *cel*. Typ równy zero oznacza zapytanie tylko o czas. Gdy zapytanie ma typ równy jeden, należy także podać trasę przejazdu.

The first line of input will contain the dimensions of the map: width w and height h. The next h lines (each containing w characters) will describe the map. Each character in the description is a dot (.) representing an empty field, a cross (#) representing a road, an asterisk (*) representing a city, or a letter or number that is part of the city name.
The next line will contain a single integer k - the number of flight connections. The next k lines will describe the connections in the format source destination time, where source is the name of the starting city, destination is the name of the destination city, and time is the flight time in minutes. The next line will contain a single integer q - the number of queries. Each query will appear on a separate line and will have the format source destination type. This is a query for the

shortest travel time from the source city to the destination city. If the type is zero, the query is only for the time. If the type is one, the route should also be provided.

Wyjście:/Output:

Na wyjściu należy wypisać dla każdego zapytania jedną linię. Na początku linii powinna pojawić się liczba, będąca najkrótszym czasem podróży pomiędzy miastami. Jeżeli zapytanie ma typ równy jeden, należy także wypisać, po spacji, wszystkie miasta pośrednie (bez startowego i końcowego) w kolejności ich odwiedzania.

For each query, output one line. At the beginning of the line should be the number representing the shortest travel time between the cities. If the query has a type of one, all intermediate cities (excluding the starting and ending cities) should also be listed in the order they were visited, separated by spaces.

Testy:

Tutaj
Opis testów:
1 - przykład
2 - minimalny
3 - poprawność czytania etykiet
4 - poprawność
5 - prosty test na kracie
6 - mały wachlarz
7 - mała ścieżka
8 - mały graf pełny
9 - duża mapa, mało miast
10 - duża mapa, mało miast
11 - wachlarz, implementacja z tablicą raczej nie zmieści się w czasie
12 - spirala, rekurencja implementacja przeszukiwania przepełni stos
13 - długa ścieżka, implementacja z tablicą raczej nie zmieści się w czasie
14 - dużo krawędzi

Tests:

Here
Opis testów:
1 - example
2 - minimum
3 - label reading correctness
4 - correctness
5 - simple test on a grid
6 - small fan
7 - small path
8 - small complete graph
9 - large map, few cities
10 - large map, few cities
11 - fan, array implementation may not fit in time

12 - spiral, recursion implementation of search overflows the stack
13 - long path, array implementation may not fit in time
14 - many edges

Przykład:/Example:

Wejście:/Input:

```
20 20
.........GDANSK.....
........*...........
........#...........
........#...........
*###################.
#SZCZECIN.........#.
#.................#.
##................#.
.#############*#####.
.#...WARSZAWA.......
.#..................
.##############......
.#...........#......
.#..WROCLAW.##......
.#..*.......*.......
.####.......#KIELCE.
......*##.#########.
.OPOLE..#.*.......#.
........#.KRAKOW..#.
........###########.
0
3
KIELCE KRAKOW 0
KRAKOW GDANSK 0
KRAKOW GDANSK 1
```

Wyjście:/Output:

```
5
40
40 KIELCE SZCZECIN
```