Najkrótsze ścieżki w grafach

Biblioteka Graphs zawiera statyczną klasę Paths, która udostępnia podstawowe algorytmy znajdujące najkrótsze ścieżki w grafach:

- BellmanFord – wyznacza najkrótsze ścieżki z zadanego wierzchołka algorytmem Forda-Bellmana,
- NegativeCycle – znajduje ujemny cykl posługując się modyfikacją algorytmu Bellmana-Forda,
- Dijkstra – wyznacza najkrótsze ścieżki z zadanego wierzchołka algorytmem Dijkstry,
- FloydWarshall – wyznacza najkrótsze ścieżki algorytmem Floyda-Warshalla,
- Johnson – wyznacza najkrótsze ścieżki algorytmem Johnsona.

Powyższe metody zgłaszają wyjątek, jeżeli algorytm się nie powiedzie z powodu niespełnienia założeń (np. istnienie ujemnych cykli lub krawędzi o ujemnej wadze).

Wynikiem działania algorytmów jest obiekt typu PathsInfo, która zawiera następujące metody

- Reachable – sprawdza, czy istnieje ścieżka między dwoma zadanymi wierzchołkami,
- GetDistance – zwraca odległość między zadanymi wierzchołkami; jeśli ścieżka nie istnieje, zgłaszany jest wyjątek,
- GetPath – znajduje najkrótszą ścieżkę między zadanymi dwoma wierzchołkami; jeśli ścieżka nie istnieje, metoda zwraca null.

Zadanie: labirynt

Dany jest labirynt reprezentowany przez dwuwymiarową tablicę typu char - na przykład:

XSOOOOO
XXXXXOX
OOOOOOO
OXXXXXX
OOOEXOO

- S – punkt startowy
- E – punkt końcowy
- X – ściany
- O – droga

Wersja I - Rozgrzewka (0.5 pkt.)

Wyznaczyć najkrótszy czas, w jakim można dotrzeć z punktu S do E poruszając się tylko po drodze. Poruszanie możliwe jest tylko o jedno pole w dół, w górę, w lewo lub w prawo. Przejście pomiędzy sąsiednimi polami zajmuje 1 jednostkę czasu. Jeśli dojście do punktu końcowego nie jest możliwe zwróć -1.

Wersja II - Wybuchowa (0.5 pkt.)

Mamy dynamit (potencjalnie nieskończoną jego ilość), za pomocą którego możemy burzyć ściany. Zburzenie ściany i przejście na pole, na którym się znajdowała zajmuje t jednostek czasu. Wejście na drogę z pola po zburzonej ścianie zajmuje 1 jednostkę czasu. Wyznaczyć najkrótszy czas w jakim można przejść labirynt z użyciem dynamitów.

Wersja III - 1 dynamit (1 pkt.)

Do dyspozycji mamy tylko 1 sztukę dynamitu. Pozostałe założenia jak w poprzednich wersjach. Wyznaczyć najkrótszy czas w jakim można przejść labirynt z użyciem co najwyżej 1 dynamitu.

Wersja IV - k dynamitów (1 pkt.)

Do dyspozycji mamy dokładnie k sztuk dynamitu. Pozostałe założenia jak w poprzednich wersjach. Wyznaczyć najkrótszy czas w jakim można przejść labirynt z użyciem co najwyżej k sztuk dynamitu.

Oczekiwana złożoność wersji I, II, III: $O(n \log n)$, gdzie n to liczba pól labiryntu. Oczekiwana złożoność wersji IV: $O(nk \log(nk))$, gdzie n to liczba pól labiryntu, a k to liczba dostępnych dynamitów.

Wskazówki:

- Główny element zadania to stworzenie z labiryntu odpowiedniego grafu
- Można korzystać z bibliotecznych metod wyznaczania najkrótszych ścieżek.
- (W wersjach III i IV) Stworzony graf powinien składać się z k+1 „warstw", kolejne warstwy odpowiadają kolejnym użyciom dynamitu

Oprócz realizacji powyższych poleceń należy dodać wyznaczanie znalezionej ścieżki w postaci napisu (typ string) składającego się z kolejnych kierunków świata (N - północ, S - południe, E - wschód, W - zachód), w jakich należy podążać od punktu startowego do punktu końcowego, aby osiągnąć wartość zwróconego rozwiązania.

Przykład: Dla labiryntu z treści zadania bez dynamitów należy zwrócić: EEEESSWWWWWSSEEE

Aby przetestować poprawność zwracanej ścieżki w Mainie zmień wartość parametru checkPath na true.