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.

Zadanie: Nowe ścieżki

Pewna Firma transportowa opatentowała niesamowity napęd samochodowy. Dzięki temu napędowi pojazdy mogą poruszać się z tak dużą prędkością, że czas przejazdu między dwoma punktami przestał być dla firmy problemem. Niestety takie pojazdy mają jedną znaczącą wadą, są okropnie szerokie. Spowodowało to, iż znane do tej pory, najkrótsze ścieżki na trasach obsługiwanych przez firmę zaczęły być bardzo uciążliwe, gdyż nowe samochody nie mieściły się na niektórych drogach.

Część I – najszerze ścieżki (1p)

Dyrektorzy firmy postanowili zatrudnić grupę studentów, aby Ci pomogli im zaprojektować nowe połączenia w taki sposób, który znajduje najszersze możliwe ścieżki. Najszerszą ścieżką jest taka ścieżka, dla której najwęższa część trasy jest największa; innymi słowy, chcemy zmaksymalizować najwęższe miejsce na trasie. Jako wejście, studentom został przekazany graf skierowany, w którym wagi krawędzi oznaczają szerokość trasy (trasa z punktu a do punktu b może mieć inną szerokość, niż z punktu b do punktu a), punkt początkowy i punkt końcowy. Jako wynik powinna zostać zwrócona trasa między punktem początkowym a końcowym, która maksymalizuje szerokość najwęższego połączenia. Jeśli taka ścieżka nie istnieje, należy zwrócić pustą listę.

Część II – epidemiczne najszerze ścieżki (1.5p)

Niestety po jakimś czasie z powodu epidemii, której nie udało się opanować mimo doskonałych algorytmów identyfikujących potencjalne jej źródła, władze krajowe zaczęły ustawiać bramki na wjazdach do miast. Na bramkach tych każdy pojazd musi odczekać pewien czas. Dyrektorzy, mimo początkowego zadowolenia z pierwszej części, zaczęli dostrzegać, że w nowej rzeczywistości znalezione wcześniej ścieżki są dalekie od optymalnych. Wyznaczyli więc nowe zadanie, polegające na znalezieniu kompromisu między szerokością a czasem oczekiwania: należy zmaksymalizować różnicę między najwęższą krawędzią znalezionej ścieżki, a sumą czasów oczekiwania na bramkach, przez które przechodzi ta ścieżka. Jako wejście został przekazany, podobnie jak poprzednio, graf skierowany, w którym wagi krawędzi oznaczają szerokość trasy, punkt początkowy, punkt końcowy, lista wag wierzchołków, które oznaczają długość stania na bramce w danym wierzchołku oraz wartość najszerszej krawędzi w całym grafie. Jako wynik powinna zostać zwrócona trasa między punktem początkowym a końcowym, dla której zmaksymalizowana będzie różnica między najwęższą krawędzią w ścieżce a sumą wag wierzchołków, przez które przechodzi ta ścieżka.

Uwagi i wskazówki

- W obu częściach możemy przyjąć, że wagi są nieujemnymi liczbami całkowitymi
- W części I oczekiwana złożoność to O (m + n log n), gdzie m jest liczbą krawędzi, a n liczbą wierzchołków grafu zadanego na wejściu
- W części II oczekujemy złożoności O (k * (m + n log n)), gdzie m i n są analogiczne jak w pierwszej części, a k jest liczbą różnych szerokości krawędzi w grafie
- Kara za niedotrzymanie wymagań odnośnie złożoności to 0.5p za każdą z części
- Jednym z pomysłów na rozwiązanie drugiej części jest sprawdzanie odpowiednio zdefiniowanych najkrótszych ścieżek dla podgrafów wejściowego grafu, które zawierają odpowiednie krawędzie