Politechnika Warszawska
Wydział Matematyki i Nauk Informacyjnych
Kierunek Matematyka i analiza danych

Park Krajobrazowy - Projekt AiSD

Autorzy:
Mateusz Kwiatkowski,
Jan Wojtas

Spis treści

1 Wstęp

1.1 Treść projektu

W Górach Bajdockich otwarto niedawno Park Krajobrazowy, w którym utworzono n tarasów widokowych (ponumerowanych od 1 do n), z których można podziwiać piękno malowniczych gór Bajtocji. Po Parku Krajobrazowym można poruszać się jedynie wytyczonymi m dwu-kierunkowymi ścieżkami, które łączą tarasy widokowe. Zwiedzanie parku zaczyna się od tarasu numer 1 położonego w pobliżu wejścia. Z każdego tarasu widokowego można dotrzeć do wszystkich pozostałych ścieżką bezpośrednią lub pośrednią (odwiedzając inne tarasy widokowe). Wstęp na każdy taras jest płatny i kosztuje tyle bajtalarów, jaki jest numer tarasu. Najbliższy weekend, o ile dopisze pogoda, Bajtazar postanowił spędzić w Parku Krajobrazowym i dotrzeć do tarasu numer t uznawanego za szczególnie urokliwy. Niestety jego możliwości finansowe są obecnie ograniczone, stąd koszt trasy do wybranego tarasu nie może przekroczyć kwoty K bajtalarów. Bajtazar zwrócił się do Was o pomoc. Czy jego plan weekendowy uda się zrealizować? A jeśli tak, jaką trasę (kolejność odwiedzania tarasów widokowych) polecacie? To właśnie jest tematem Waszego zadania projektowego.

1.2 Problem w języku grafów

Na pierwszy rzut oka widać, że problem przedstawiony w treści można wyrazić w języku grafów, w którym tarasy są wierzchołkami, a ścieżki są krawędziami. Wiemy, że z każdego tarasu można dojść do wszystkich pozostałych, więc graf jest spójny. Zauważmy jednak, że koszt przejścia z tarasu i do tarasu j jest inny niż koszt przejścia z j do i. Zważając na to, postanowiliśmy zmienić dany graf na digraf ważony o $2m$ krawędziach, w każda krawędź postaci (i, j) ma wagę j. Naszym celem jest znalezienie najkrótszej (najtańszej) drogi z wierzchołka 1 do wierzchołka t w otrzymanym digrafie.

2 Generowanie danych

2.1 Dane wejściowe

Nasz program prosi użytkownika o podanie czterech liczb całkowitych: n - liczby tarasów, m - liczby ścieżek, t - numeru tarasu, do którego chcemy dojść i K - liczby bajtalarów. Następnie sprawdzamy, czy podane wartości znajdują się w odpowiednich przedziałach: $1 \leqslant n \leqslant 100, 1 \leqslant m \leqslant min(300, \frac{n(n-1)}{2}), 1 \leqslant t \leqslant n, K > 0$.

Kolejnym krokiem jest wylosowanie m różnych ścieżek. Każda ścieżka łączy dwa tarasy, które reprezentowane są przez liczby ze zbioru $1, 2, ..., n$

2.2 Generowanie macierzy i list

Żeby móc działać na grafie reprezentującym park musimy zapisać dane w formie macierzy incydencji M, gdzie $M[i][j]$ to koszt bezpośredniego przejścia z tarasu i do tarasu j (jeśli ścieżka między tymi tarasami nie istnieje to przyjmujemy $M[i][j] = 2^{10}$). Tak jak zaznaczyliśmy wcześniej, chcemy utworzyć digraf ważony, dlatego macierz M nie będzie symetryczna.
Z warunków zadania wynika, że jeśli istnieje ścieżka między tarasami i, j to $M[i][j] = j$.
Przez NS oznaczymy zbiór wierzchołków jeszcze nieodwiedzonych, na początku $NS = \{2, 3, ..., n\}$.
W dalszej części kodu będziemy również korzystali z tablicy pomocniczej D, w której będziemy przechowywali najmniejsze wyznaczone odległości ze źródła do kolejnych wierzchołków i informacje, przez który wierzchołek musimy przejść. Jeśli jest trasa bezpośrednia z 1 do i to $D[i+1][0] = i$ a jeśli nie to na początku wpisujemy 2^{10}.

2.3 Pseudokod

void stwórz_macierz_incydencji(int rozmiar, list lista_krawędzi)
begin
for i:= 0 to rozmiar **do**:
  for j:=0 to rozmiar **do**:
    $M[i][j] := 2^{10}$
for $v \in$ lista_krawędzi **do**:
  $M[v[0]-1][v[1]-1] := v[1]$
  $M[v[1]-1][v[0]-1] := v[0]$
return M
end

l = stwórz_macierz_incydencji(liczba_tarasów, wybrane_ścieżki)
źródło := 1
n:= liczba_tarasów

D - tablica $n \times 1$ (elementami tablicy są tablice wymiaru 2×1)
D[źródło-1][1] := źródło

  for $i := 0$ to n **do**:
    D[i][0] = l[źródło - 1][i]
    if l[źródło-1][i] $< 2^{10}$ **then** $D[i][1] :=$ źródło **fi od**

NS - tablica wymiaru $n-1$
for $i := 0$ to n **do**:
  $NS[i] := i + 2$ **od**

3 Algorytm Dijkstry

3.1 Opis algorytmu

Wcześniej zaznaczyliśmy, że skupimy się na poszukiwaniu najkrótszej drogi w zadanym grafie. Oczywiście wykorzystamy do tego, poznany na wykładzie, algorytm Dijkstry.
W tym algorytmie będziemy modyfikowali tablicę D, by znalezione przez nas ścieżki do kolejnych wierchołków rzeczywiście były najtańsze. Każdy obrót pętli znajduje wierzchołek, do którego droga jest najtańsza, oznacza ten wierzchołek jako odwiedzony (czyli wyrzucamy go z NS), a następnie sprawdza czy nie można zminimalizować kosztów podróży do każdego z pozostałych tarasów, idąc przez znaleziony wierzchołek. Procedurę powtarzamy dopóki nie odwiedzimy wszystkich wierzchołków. Działanie algorytmu przedstawiliśmy przy pomocy pseudokodu.

3.2 Pseudokod

$i := 1$
while NS **do**:
    $mindist = 2^{10}$
    for $w \in NS$ **do**:
        if $D[w-1][0] < mindist$ **then**:
            $v := w$
            $mindist := D[w-1][0]$ **fi od**
    if $v = numer\_wybranego\_tarasu$ **then**:
        break **fi**
    NS.remove(v)
    $i += 1$
    for $w \in NS$ **do**:
        if $D[w-1][0] > mindist + l[v-1][w-1]$ **then**:
            $D[w-1][0] = mindist + l[v-1][w-1]$
            $D[w-1][1] = v$ **fi od**
od

for $i := 0$ to $n-1$ **do**:
    D[i][0] += 1 **od**

4 Najkrótsza ścieżka do wybranego tarasu, koszt trasy

4.1 Znajdowanie najkrótszej ścieżki

Tablica D, którą otrzymaliśmy w wyniku algorytmu Dijkstry mówi jaki jest koszt dotarcia do danego tarasu ze źródła oraz jaki taras musimy odwiedzić bezpośrednio przed dotarciem do celu. Najtańszą trasę zapiszemy w liście zaczynając od końca. Na końcu chcemy się znaleźć na tarasie t, więc od niego zaczniemy. Z D możemy odczytać jaki wierzchołek musieliśmy odwiedzić bezpośrednio przed t: $punkt_trasy := D[t-1][1]$. Punkt trasy zapisujemy do tabeli i powtarzamy zadaną procedurę ($punkt_trasy := D[punkt_trasy - 1][1]$) do momentu gdy $punkt_trasy == 1$.

4.2 Sprawdzanie kosztu trasy

Koszty najtańszej trasy zapisywaliśmy w algorytmie Dijkstry w tablicy D. Musimy zatem tylko odczytać wartość $D[t-1][0]$. W odpowiedzi mamy stwierdzić, czy Bajtazara stać na dotarcie do tarasu t, więc musimy sprawdzić, czy zachodzi nierówność $D[t-1][0] \leqslant K$.

4.3 Pseudokod

Trasa - tablica wymiaru n
$Trasa[0] := numer\_wybranego\_tarasu$
$punkt\_trasy := D[numer\_wybranego\_tarasu - 1][1]$
$Trasa[1] := punkt\_trasy$
$licznik := 1$

while $punkt\_trasy \neq 1$ **do**:
  $punkt\_trasy := D[punkt\_trasy - 1][1]$
  $Trasa[licznik + 1] := punkt\_trasy$
  $licznik += 1$ **od**

Trasa_od_startu - tablica wymiaru $licznik + 1$

for $i := 0$ to $licznik + 1$ **do**:
  $Trasa\_od\_startu[i] := Trasa[licznik - i]$
od

$Koszt\_trasy := D[numer\_wybranego\_tarasu - 1][0]$

5 Złożoność

5.1 Złożoność pamięciowa

Wyniki algorytmu Dijkstry zapisujemy w tablicy D o długości n (liczba tarasów). Przy tworzeniu tablicy D używamy macierzy incydencji grafu, co oznacza, że nasz program ma kwadratową złożoność pamięciową.

5.2 Złożoność obliczeniowa

Prześledźmy działanie algorytmu Dijkstry:

- Pętla *while* przechodzi po tablicy NS, więc wykona ona maksymalnie $n - 1$ obrotów
- Pierwsza pętla for, znajdująca się wewnątrz powyżej opisanej pętli $while$, przechodzi po wszystkich elementach, które znajdują się aktualnie w tablicy NS. Dla każdego z nich wykonuje maksymalnie 3 działania, więc liczbę operacji tego $for'a$ możemy oszacować od góry przez $3i$ (gdzie i to liczba elementów w NS)
- Następnie usuwamy jeden element z NS
- Druga pętla for również przechodzi po NS i wykonuje maksymalnie 3 działania dla każdego elementu, zatem liczba wykonywanych przez nią operacji to maksymalnie $3(i-1)$
- Jeśli szybciej trafimy na numer wybranego tarasu to pętla *while* przerwie swoje działanie, co pozwoli zaoszczędzić sporo obliczeń, jednak w pesymistycznym wariancie będzie ona musiała przejść przez całą tablicę NS
- Na końcu musimy dodać jedynkę do wszystkich elementów z tablicy kosztów, ponieważ jest to cena wejścia na pierwszy taras.

Możemy teraz podsumować liczbę operacji wykonanych przez algorytm Dijkstry (w najbardziej pesymistycznym wariancie):
$T(n) = \sum_{i=1}^{n-1}[3i + 3(i-1) + 3] + n = \sum_{i=1}^{n-1} 6i + n = 6\sum_{i=1}^{n-1} i + n = \frac{6(n-1)n}{2} + n = 3n^2 - 2n$
Wszystkie pozostałe, wykonywane przez nas działania, takie jak tworzenie macierzy incydencji lub odczytywanie ścieżki z tablicy, mają złożoność liniową, więc nie zwiększą one złożoności całego kodu. Stąd $T(n) = O(n^2)$.

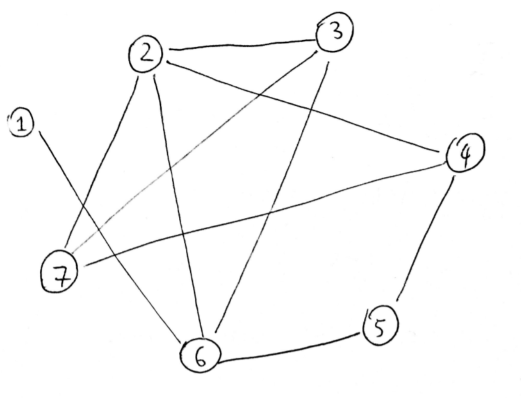
6 Przykłady

Sekcja poświęcona przykładom, pokazującym poprawność działania algorytmu.

6.1 Przykład nr 1

Dane wejściowe:
1) Liczba tarasów : 7
2) Ścieżki: [(5, 6), (2, 3), (2, 4), (1, 6), (2, 6), (4, 7), (2, 7), (3, 7), (3, 6), (4, 5)]
3) Numer tarasu, do którego Bajtazar chce dotrzeć: 7
4) Liczba Bajtalarów: 15

Reprezentacja graficzna:



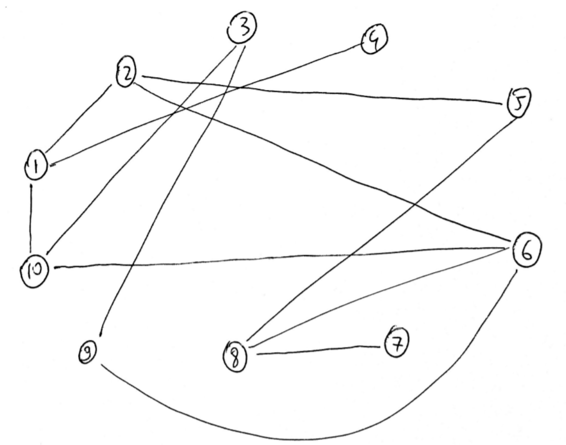
Rozwiązanie:

```
Lista reprezentująca koszt trasy ze źródła do konkretnego tarasu i numer poprzednika na ścieżce:
[[1, 1], [9, 6], [10, 6], [13, 2], [12, 6], [7, 1], [16, 2]]
Najtańsza droga do tarasu nr 7 to: [1, 6, 2, 7].
Jej koszt to 16 Bajtalarów
Czy Bajtazarowi wystarczy bajtalarów na trasę: nie
```

6.2 Przykład nr 2

Dane wejściowe:
1) Liczba tarasów : 10
2) Ścieżki: [(6, 10), (3, 10), (3, 9), (2, 5), (7, 8), (5, 8), (1, 2), (6, 8), (1, 4), (1, 10), (6, 9), (2, 6)]
3) Numer tarasu, do którego Bajtazar chce dotrzeć: 7
4) Liczba Bajtalarów: 20

Reprezentacja graficzna:



Rozwiązanie:

```
Lista reprezentująca koszt trasy ze źródła do konkretnego tarasu i numer poprzednika na ścieżce:
[[1, 1], [3, 1], [14, 10], [5, 1], [8, 2], [9, 2], [23, 8], [16, 5], [18, 6], [11, 1]]
Najtańsza droga do tarasu nr 7 to: [1, 2, 5, 8, 7].
Jej koszt to 23 Bajtalarów
Czy Bajtazarowi wystarczy bajtalarów na trasę: nie
```

6.3 Przykład nr 3

Dane wejściowe:

1) Liczba tarasów: 15
2) Liczba ścieżek: 40
3) Numer tarasu, do którego Bajtazar chce dotrzeć: 15
4) Liczba bajtalarów: 50

Rozwiązanie:

```
Lista reprezentująca koszt trasy ze źródła do konkretnego tarasu i numer poprzednika na ścieżce:
[[1, 1], [8, 5], [4, 1], [8, 3], [6, 1], [10, 3], [13, 5], [16, 2], [13, 3], [11, 1], [12, 1], [20, 2], [17, 3], [27, 7], [25, 6]]
Najtańsza droga do tarasu nr 15 to: [1, 3, 6, 15].
Jej koszt to 25 Bajtalarów
Czy Bajtazarowi wystarczy bajtalarów na trasę: tak
```