# ROZDZIAŁ 3

# Najkrótsze drogi i drzewa

## 3.1 Elementy teorii grafów

Na początku przypomnimy podstawowe pojęcia i proste fakty.

1. Parę $G = (V, E)$ nazywamy *grafem (nieskierowanym)*, gdzie $V = V(G) = \{v_1, ..., v_n\}$ jest zbiorem *wierzchołków*, $E = E(G) = \{e_1, ..., e_m\}$ jest zbiorem *krawędzi*, jeśli każdej krawędzi $e \in E$ przyporządkowana jest para (nieuporządkowana) wierzchołków $v_1, v_2 \in V$ – *końców* tej krawędzi; mówimy, że krawędź jest *związana* z jej końcami, a końce te są względem siebie wierzchołkami *sąsiednimi*.

2. Krawędzie grafu $G$ nazywamy krawędziami *równoległymi* jeśli mają te same końce.

3. *Pętlą* w grafie $G$ nazywamy jego krawędź o jednakowych końcach.

4. Grafem *prostym* nazywamy graf bez krawędzi równoległych i pętli.

5. Dla grafu $G = (V, E)$ przez $e = v_1v_2$ oznaczamy krawędź o końcach $v_1, v_2 \in V$ (zakładamy, że graf ten jest bez krawędzi równoległych); symbole $v_1v_2$ i $v_2v_1$ oznaczają tę samą krawędź.

6. Grafem *pełnym* nazywamy graf prosty, dla którego każda para różnych wierzchołków jest połączona krawędzią.

7. Mówimy, że graf $G' = (V', E')$ jest *podgrafem* grafu $G = (V, E)$, jeśli $V' \subset V$, $E' \subset E$.

8. Dla $W \subset V$ przez $G[W]$ oznaczamy podgraf $G' = (W, A)$ grafu $G$, gdzie $A$ jest zbiorem tych krawędzi grafu $G$, których oba końce należą do $W$.

9. Dla $A \subset E$ przez $G \diagdown A$ oznaczamy podgraf $G' = (V, E \diagdown A)$; dla $e \in E$ przez $G \diagdown e$ oznaczamy podgraf $G \diagdown \{e\}$.

10. Dla $W \subset V$ przez $G \setminus W$ (lub przez $G[V \diagdown W]$) oznaczamy podgraf grafu $G$ powstały przez usunięcie wierzchołków $W$ i związanych z nimi krawędzi; dla $v \in V$ przez $G \diagdown v$ oznaczamy podgraf $G \diagdown \{v\}$.

11. Dla grafu $G = (V, E)$ i dla nietrywialnego podzbioru $W \subset V$ (tzn. $W \neq \varnothing$ i $W \neq V$) przez $\delta(W)$ oznaczamy zbiór tych krawędzi grafu $G$ mających jeden koniec w $W$ zaś drugi w $V \diagdown W$; zbiór postaci $\delta(W)$ dla pewnego $W \subset V$ nazywamy *cięciem* grafu $G$.

12. Dla grafu $G = (V, E)$ i dla $W \subset V$ przez $\gamma(W)$ oznaczamy zbiór tych krawędzi grafu $G$ mających oba końce w $W$.

13. Podgraf $G' = (V', E')$ grafu $G = (V, E)$ jest grafem *rozpinającym* graf $G$ jeśli $V' = V$.

14. *Drogą* $P$ w grafie $G$ łączącą wierzchołki $v_0$ i $v_k$ (lub $(v_0, v_k)$-*drogą*) nazywamy ciąg krawędzi $e_0, e_1, ..., e_k$ tego grafu, gdzie $e_i = v_{i-1}v_i$, $i = 1, ..., k$, (dwie kolejne krawędzie mają wspólny wierzchołek); drogę $P$ w grafie $G$ możemy utożsamiać z podgrafem

$$G' = (\{v_0, ..., v_k\}, \{e_1, ..., e_k\}),$$

w którym wyróżniono dwa końce $v_0$ i $v_k$ i w którym zachodzi opisany wyżej związek między wierzchołkami i krawędziami. Drogę tę zapisujemy również w postaci $P = (e_0, e_1, ..., e_k)$ albo $P = (v_0v_1...v_k)$. W szczególności $k$ może być równe 0. Wówczas droga $P$ jest grafem złożonym z jednego wierzchołka $v_0$ bez krawędzi. Mówimy, że jest to droga łącząca wierzchołek $v_0$ ze sobą.

15. Dla grafu $G = (V, E)$ i dla wierzchołka $v \in V$ przez $C_v$ oznaczamy zbiór wszystkich wierzchołków $w$ grafu $G$, dla których istnieje $(v, w)$-droga; zbiory $C_v$ są klasami abstrakcji dla relacji typu równoważności

$$v \sim w \text{ jeśli istnieje } (v, w)\text{-droga w grafie } G$$

określonej na $V$.

16. Podgraf grafu $G = (V, E)$ postaci $G[C_v]$ dla pewnego $v \in V$ nazywamy *składową* grafu $G$.

17. Graf $G$ nazywamy grafem *spójnym*, jeśli dowolne dwa jego wierzchołki można połączyć drogą.

18. Jeśli $G = (V, E)$ jest grafem spójnym, to $\#E \geq \#V - 1$.

19. Graf $G$ jest spójny wtedy i tylko wtedy jest swoją (jedyną) składową.

20. Droga $P$ łącząca wierzchołki $v_0$ i $v_k$ grafu $G$ jest *zamknięta* jeśli $v_0 = v_k$.

21. Droga $P = (v_0v_1, ..., v_{k-1}v_k)$ jest *prosta* jeśli wierzchołki $v_0, ..., v_k$ są różne.

22. Jeśli istnieje droga łącząca dwa wierzchołki, to istnieje łącząca je droga prosta.

23. Droga $P$ nazywa się *cyklem* jeśli jest zamknięta i prosta.

24. Krawędź $e = uv$ grafu $G$ należy do cyklu wtedy i tylko wtedy, gdy istnieje droga zawarta w $G \diagdown e$ łącząca wierzchołki $u$ i $v$.

25. Jeśli graf spójny $G = (V, E)$ zawiera cykl, to $\#E \geq \#V$.

26. Jeśli graf spójny $G = (V, E)$ nie zawiera cyklu, to $\#E = \#V - 1$.

27. Dla grafu $G$ *cyklem Hamiltona* nazywamy cykl łączący wszystkie wierzchołki grafu $G$.

28. Wierzchołek $v$ grafu spójnego $G$ nazywamy wierzchołkiem *tnącym*, jeśli graf $G \setminus v$ nie jest spójny.

29. *Lasem* nazywamy graf nie zawierający cykli.

30. *Drzewem* nazywamy las spójny.

31. *Dendrytem* nazywamy drzewo rozpinające.

32. *Wagą* (*długością*, *kosztem*) $c(e)$ krawędzi $e \in E$ nazywamy wartość funkcji $c : E \to \mathbb{R}$ zwanej *funkcją wag* (długości, kosztów) dla $e \in E$, wagę tę będziemy oznaczać również symbolem $c_e$.

33. *Siecią* nazywamy graf $G = (V, E)$ z określoną funkcją wag $c : E \to \mathbb{R}$.

W dalszej części rozważamy grafy proste.

**Lemat 3.1.1** *Krawędź $e = vv'$ grafu $G$ jest krawędzią cyklu wtedy i tylko wtedy, gdy w grafie $G \setminus e$ istnieje droga łącząca $v$ i $v'$.*

**Dowód**.

$\Rightarrow$ Niech $C = (v_0v_1, ..., v_{k-1}v_k)$ będzie cyklem w grafie $G$ i niech $e = vv'$ będzie krawędzią tego cyklu. Oczywiście $e = e_i = v_{i-1}v_i$ dla pewnego $i$, $i = 1, ..., k$. Jest jasne, że droga

$$P_i = (v_iv_{i+1}, ..., v_{k-1}v_0, v_0v_1, ..., v_{i-2}v_{i-1})$$

jest podgrafem grafu $G \setminus e$ jest drogą łączącą $v$ i $v'$.

$\Leftarrow$ Załóżmy teraz, że dla pewnej krawędzi $e = vv'$ grafu $G$, w grafie $G \setminus e$ istnieje droga $P$ łącząca $v$ i $v'$. Zatem istnieje droga prosta łącząca $v$ i $v'$ Dla pewnych wierzchołków $w_0, ..., w_k$ grafu $G \setminus e$ zachodzi więc

$$P = (w_0w_1, ..., w_{k-1}w_k),$$

gdzie $w_0 = v$ i $w_k = v'$. Ponieważ $e = w_kw_0$, więc

$$C = (w_0w_1, ..., w_{k-1}w_k, w_kw_0)$$

jest podgrafem grafu $G$. Ponadto jest on oczywiście cyklem. ■

**Lemat 3.1.2** *Graf $G = (V, E)$ jest spójny wtedy i tylko wtedy, gdy dla dowolnego nietrywialnego podzbioru wierzchołków $W \subset V$ zbiór $\delta(W)$ jest niepusty.*

**Dowód**.

$\Rightarrow$ Przypuśćmy, że graf $G = (V, E)$ jest spójny i niech $W \neq \varnothing$ będzie dowolnym istotnym podzbiorem wierzchołków tego grafu. Niech $v \in W$ i niech $v' \in V \setminus W$. Ponieważ $G$ jest grafem spójnym, więc istnieje droga prosta $P = (v_0v_1, ..., v_{k-1}v_k)$ taka, że $v = v_0$ i $v' = v_k$. Niech $i_W = \max\{i : 0 \leq i \leq k, v_i \in W\}$. Oczywiście $i_W < k$ oraz $v_{i_W+1} \in V \setminus W$. Zatem $e = v_{i_W}v_{i_W+1} \in \delta(W)$.

$\Leftarrow$ Przypuśćmy teraz, że graf $G$ nie jest spójny, tzn. dla pewnych wierzchołków $v, v' \in V$ nie istnieje droga łącząca $v$ i $v'$. Zatem $C_v \cap C_{v'} = \varnothing$. Dla nietrywialnego podzbioru krawędzi $W = C_v$ cięcie $\delta(W)$ jest więc zbiorem pustym. ■

**Lemat 3.1.3** *Niech graf $G = (V, E)$ będzie spójny. Wówczas następujące warunki są równoważne:*

(i) *$G$ jest drzewem,*

(ii) $\#E = \#V - 1$,

(iii) *każdą parę wierzchołków grafu $G$ można połączyć dokładnie jedną drogą,*

(iv) *po dodaniu jednej krawędzi do zbioru $E$ powstały graf $G'$ zawiera dokładnie jeden cykl,*

(v) *po usunięciu dowolnej krawędzi ze zbioru $E$ powstały graf $G'' = G \setminus e$ traci spójność.*

**Dowód**. (i)⇒(ii) Niech $G$ będzie drzewem. Ponieważ $G$ jest grafem spójnym nie zawierającym cyklu więc $\#E = \#V - 1$ (patrz własność 26).

(ii)⇒(iii) Niech $\#E = \#V - 1$. Przypuśćmy, że pewne dwa wierzchołki $v$ i $v'$ grafu $G$ można połączyć dwiema różnymi drogami $P$ i $P'$. Bez szkody dla ogólności możemy założyć, że wierzchołki w $P$ i $P'$ poza początkiem i końcem są różne. Wówczas $(P, P')$ jest cyklem w grafie spójnym $G$ i w konsekwencji $\#E \geq \#V$ (patrz własność 25), co stoi w sprzeczności z założeniem.

(iii)⇒(iv) Niech $e = vv' \notin E$ dla pewnych wierzchołków $v, v'$ grafu $G$ i niech $G' = (V, E \cup \{e\})$. Niech $P$ będzie drogą prostą w grafie $G$ łączącą wierzchołki $v$ i $v'$. Wówczas $(P, e)$ jest cyklem w grafie $G'$. Jest to jedyny cykl w tym grafie. Gdyby powiem istniał inny cykl $(P', e)$, to $P'$ byłaby inną niż $P$ drogą w grafie $G$ łączącą wierzchołki $v$ i $v'$.

(iv)⇒(v) Niech $e = vv' \in E$. Przypuśćmy, że graf $G'' = G \setminus e$ jest spójny. Niech $P$ będzie drogą prostą w grafie $G''$ łączącą wierzchołki $v$ i $v'$. Oczywiście $(P, e)$ jest cyklem w grafie $G$. Stoi to w sprzeczności z założeniem, gdyż po dodaniu pewnej krawędzi $e' \notin E$ do grafu $G$ powstały graf $G'$ będzie zawierał co najmniej dwa cykle.

(v)⇒(i) Przypuśćmy, że graf $G$ nie jest drzewem. Ponieważ, zgodnie z założeniem jest on grafem spójnym, więc $G$ zawiera cykl. Z lematu 3.1.1 wynika, że po usunięciu dowolnej krawędzi tworzącej ten cykl graf ten jest nadal spójny. Stoi to w sprzeczności z założeniem. ■

**Wniosek 3.1.4** *Spójny podgraf rozpinający $H = (V, F)$ grafu $G = (V, E)$ jest drzewem wtedy i tylko wtedy, gdy $\#F = \#V - 1$.*

**Wniosek 3.1.5** *Niech $H = (V, F)$ będzie drzewem rozpinającym grafu $G = (V, E)$. Jeśli $e = uv \in E$ ale $e \notin F$ i jeśli $f \in F$ jest elementem drogi łączącej $u$ i $v$, to podgraf $H' = (V, F \setminus \{f\} \cup \{e\})$ jest drzewem rozpinającym graf $G$.*

**Dowód**. Wystarczy zauważyć, że graf $H'$ jest spójny i ma tę samą liczbę krawędzi, co graf $H$. ■

## 3.2 Najkrótsze drzewo rozpinające

**Definicja 3.2.1** Dany jest graf nieskierowany $G = (V, E)$ i funkcja wag $c : E \to \mathbb{R}$. *Długością* (lub *wagą*) grafu $G$ nazywamy liczbę $c(E) = \sum_{e \in E} c(e)$.

Rozważmy następujące zadanie zwane problemem *najkrótszego drzewa rozpinającego* (NDR) (ang. *minimum spanning tree*):

> Dany jest graf spójny $G = (V, E)$ i funkcja kosztów $c : E \to \mathbb{R}$. Znaleźć drzewo rozpinające $H = (V, F)$ grafu $G$ takie, że dla dowolnego innego drzewa $H' = (V, F')$ rozpinającego ten graf zachodzi nierówność $c(F) \leq c(F')$.

**Definicja 3.2.2** Mówimy podzbiór $F$ krawędzi grafu $G = (V, E)$ jest *rozszerzalny* do najkrótszego drzewa rozpinającego jeśli $F$ jest zawarty w zbiorze krawędzi pewnego najkrótszego drzewa rozpinającego graf $G$.

Podamy teraz pewien warunek wystarczający na to, aby po dołączeniu krawędzi do zbioru $F$ nie stracił on własności rozszerzalności do najkrótszego drzewa rozpinającego.

**Twierdzenie 3.2.3** *Jeśli podzbiór $F \subset E$ jest rozszerzalny do najkrótszego drzewa rozpinającego graf spójny $G = (V, E)$ i $e$ jest najkrótszą krawędzią pewnego cięcia $D$ rozłącznego ze zbiorem $F$, to $F \cup \{e\}$ jest również rozszerzalny do najkrótszego drzewa rozpinającego graf $G$.*

**Dowód**. Niech podzbiór krawędzi $F \subset T$ będzie rozszerzalny do NDR $H = (V, T)$ i niech $e$ będzie najkrótszą krawędzią pewnego cięcia $D$ rozłącznego z $F$. Jeśli $e \in T$, to oczywiście $F \cup \{e\}$ jest rozszerzalny do NDR. Niech więc $e = uv \notin T$. Niech $P$ będzie $(u, v)$-drogą w grafie $H$. Na mocy lematu 3.1.3 istnieje dokładnie jedna taka droga. Ponieważ $e = uv \in D$ i $D$ jest cięciem, więc w grafie $G \setminus D$ nie ma $(u, v)$-drogi. W konsekwencji pewna krawędź $f \in P \cap D$. Ponieważ $e$ jest najkrótszą krawędzią w $D$, więc $c(e) \leq c(f)$. Stąd i na mocy wniosku 3.1.5 graf $H' = (V, T \setminus \{f\} \cup \{e\})$ jest również NDR. Zauważmy teraz, że $f \notin F$ ponieważ $f \in D$ i $F \cap D = \emptyset$. Widzimy więc, że $F \cup \{e\} \subset T \setminus \{f\} \cup \{e\}$, czyli $F \cup \{e\}$ jest rozszerzalny do NDR $H'$. ■

W kolejnych punktach podamy dwa algorytmy pozwalające skonstruować najkrótsze drzewa rozpinające. Obydwa mają tzw. własność *zachłanności*, tzn. w każdej iteracji wybór jest lokalnie najlepszy.

### 3.2.1 Algorytm Kruskala

W algorytmie Kruskala punktem startowym jest las rozpinający $H = (V, \emptyset)$ grafu $G$ o pustym zbiorze krawędzi. W każdej iteracji algorytmu do zbioru krawędzi tego lasu dokładamy najkrótszą krawędź taką, aby powstały graf był nadal lasem. Zauważmy, że aby własność ta była zachowana, oba końce dokładanej krawędzi muszą należeć do różnych składowych (drzew) lasu.

**Algorytm 3.2.4 (Kruskal)**

**Wejście**: Graf spójny $G = (V, E)$ i funkcja wag $c : E \to \mathbb{R}$

**Wyjście**: Najkrótsze drzewo rozpinające $H = (V, F)$ grafu $G$

**Krok 0** (*inicjalizacja*)

(a) Uporządkować krawędzie grafu $G$ w ciąg $e_1, e_2, ..., e_m$ zgodnie z ich rosnącymi długościami, czyli tak, aby
$$c(e_1) \leq c(e_2) \leq ... \leq c(e_m).$$

(b) Położyć $F = \varnothing$ i $H = (V, F)$.

(c) Położyć $k = 1$ (*licznik iteracji*).

**Krok 1** (*kryterium zatrzymania*)

(a) Jeśli $k = m + 1$, to algorytm zatrzymuje się ($H$ jest NDR).

(b) Jeśli $H$ jest grafem spójnym, to algorytm zatrzymuje się ($H$ jest NDR).

**Krok 2** (*aktualizacja zbioru krawędzi lasu*)

(a) Sprawdzić, czy oba końce krawędzi $e_k$ należą do różnych składowych grafu $H$; jeśli tak, to położyć $F := F \cup \{e_k\}$.

(b) Położyć $k := k+ 1$.

(c) Przejść do kroku 1.

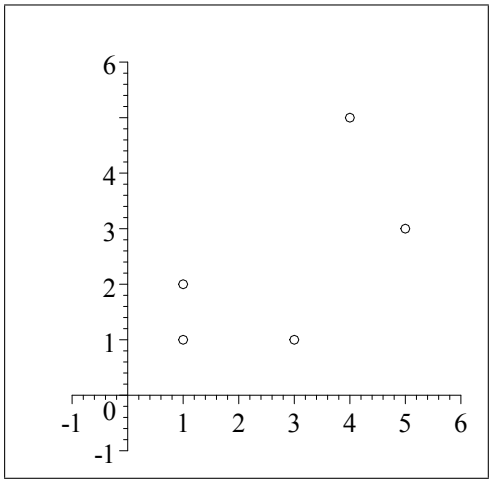
**Twierdzenie 3.2.5** *Dla dowolnego grafu spójnego $G = (V, E)$ i dowolnej funkcji długości $c : E \to \mathbb{R}$ graf $H = (V, F)$ skonstruowany przy pomocy algorytmu Kruskala jest najkrótszym drzewem rozpinającym graf $G$.*

**Dowód**. Niech $H = (V, F)$ będzie lasem rozpatrywanym w bieżącej iteracji algorytmu Kruskala. Jeśli w iteracji tej krawędź $e_k$ zostaje dołączona do zbioru $F$, to powstały graf jest nadal lasem, gdyż oba końce krawędzi $e_k$ leżą w różnych składowych grafu $H$. Zauważmy, że graf $H$ skonstruowany przy pomocy algorytmu Kruskala jest spójny. Przypuśćmy bowiem, że nie jest on spójny i niech $S_1$ i $S_2$ będą dwiema jego różnymi składowymi i niech krawędź $e_j \in \delta(S_1)$ dla pewnego $j$. Krawędź taka istnieje, bo graf $G$ jest spójny. Oczywiście $e_j \notin F$. Z postaci algorytmu wynika, że w $j$-tej iteracji algorytmu krawędź $e_j$ zostaje dołączona do zbioru krawędzi grafu $H$ i, że w kolejnych iteracjach pozostaje nadal jego krawędzią, czyli $e_j \in F$. Otrzymana sprzeczność dowodzi spójności grafu $H$. W konsekwencji $H$ jest drzewem rozpinającym, ponieważ jest on lasem rozpinającym. Ponieważ $\emptyset$ jest rozszerzalny do NDR więc w celu pokazania, że $H$ jest NDR wystarczy zauważyć – korzystając z twierdzenia 3.2.3 – że w wyniku każdej iteracji algorytmu Kruskala zostaje zachowana własność rozszerzalności do NDR. ■

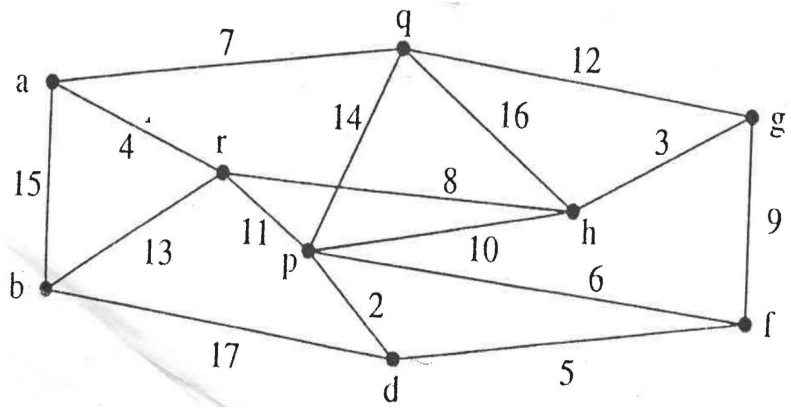
Liczba operacji wykonywanych w algorytmie Kruskala jest zdominowana przez złożoność obliczeniową sortowania, które trzeba wykonać w kroku 0., a ta wynosi jak wiadomo $O(m \log m)$. Wobec tego złożoność obliczeniowa algorytmu Kruskala jest wielomianowa.

**Przykład 3.2.6** Dana jest sieć nieskierowana pełna $(V, E, c)$, której wierzchołki $u, v, w, x, y$ umieszczone są na płaszczyźnie $\mathbb{R}^2$ i mają współrzędne $u = (5, 3)$, $v = (1, 2)$, $w = (3, 1)$, $x = (1, 1)$, $y = (4, 5)$. Na zbiorze krawędzi $E$ określona jest funkcja wag $c : E \to \mathbb{R}$ następująco: $c(e)$ jest równa odległości euklidesowej między wierzchołkami związanymi krawędzią $e$, $e \in E$.

Metodą Kruskala wyznaczyć najkrótsze drzewo rozpinające ten graf: podać kolejne krawędzie wyznaczane tą metodą i zaznaczyć te krawędzie na rysunku




**Przykład 3.2.7** Dla sieci podanej na poniższym rysunku metodą Kruskala wyznaczyć najkrótsze drzewo rozpinające.




### 3.2.2 Algorytm Prima

W algorytmie Prima punktem startowym jest drzewo $H = (\{r\}, \emptyset)$, gdzie $r$ jest dowolnym wierzchołkiem grafu $G$. W każdej iteracji algorytmu do zbioru krawędzi grafu $H$ dodajemy najkrótszą krawędź z cięcia grafu $G$ odpowiadającego zbiorowi wierzchołków grafu $H$. W wyniku tej operacji graf $H$ jest nadal drzewem. Po wykonaniu $n$ iteracji, gdzie $n$ jest liczbą wierzchołków grafu $G$ skonstruowane w opisany sposób drzewo jest drzewem rozpinającym. Sformułujemy teraz dokładnie algorytm Prima, a następnie pokażemy, że otrzymane drzewo jest najkrótszym drzewem rozpinającym graf $G$.

**Algorytm 3.2.8 (Prim)**

**Wejście**: Graf spójny $G = (V, E)$ i funkcja wag $c : E \to \mathbb{R}$

**Wyjście**: Najkrótsze drzewo rozpinające $H = (V, F)$ grafu $G$

**Krok 0** (*inicjalizacja*)

(a) Wybrać dowolny wierzchołek $r \in V$ i położyć $W = \{r\}$

(b) położyć $F = \emptyset$ i $H = (W, F)$.

**Krok 1** (*kryterium zatrzymania*)

Jeśli $\#W = \#V$, to algorytm zatrzymuje się ($H$ jest drzewem rozpinającym).

**Krok 2** (*aktualizacja zbioru krawędzi drzewa*)

(a) Wyznaczyć najkrótszą krawędź $e$ ze zbioru $\delta(W)$

(b) Położyć $F := F \cup \{e\}$

(c) Położyć $W := W \cup \{w\}$, gdzie $w$ jest końcem krawędzi $e$ nie należącym do $W$

(d) Przejść do kroku 1.

**Twierdzenie 3.2.9** *Dla dowolnego grafu spójnego $G = (V, E)$ i dowolnej funkcji długości $c : E \to \mathbb{R}$ graf skonstruowany przy pomocy algorytmu Prima jest najkrótszym drzewem rozpinającym graf $G$.*

**Dowód**. Załóżmy, że podgraf $H = (W, F)$ grafu $G$ jest drzewem i niech $f = uv \in \delta(W)$. Pokażemy, że wówczas podgraf $H^+ = H \cup \{f\} = (W \cup \{u, v\}, F \cup \{f\})$ jest również drzewem. W tym celu zauważmy, że podgraf $H^+$ jest spójny, gdyż $f \in \delta(W)$, czyli jeden z końców krawędzi $f$ należy do zbioru $W$. Ponadto podgraf $H^+$ nie zawiera cyklu. Gdyby bowiem było inaczej, to zgodnie z lematem 3.1.1 w grafie $H^+ \setminus \{f\} = (W \cup \{u, v\}, F)$ istniałaby droga łącząca wierzchołki $u$ i $v$. Jednakże jeden z tych wierzchołków nie należy do $W$, więc takiej drogi nie ma. Skoro podgraf $H^+$ jest spójny i nie zawiera cyklu, więc jest drzewem.

Zauważmy dalej, że cięcie $\delta(W)$ jest niepuste, o ile $H$ nie jest drzewem rozpinającym (czyli $W \neq V$), gdyż zgodnie z założeniem graf $G$ jest spójny.

Ponieważ zgodnie z algorytmem Prima w każdej iteracji dodajemy do zbioru wierzchołków grafu $H$ jeden nowy wierzchołek, więc po $n$ iteracjach tego algorytmu otrzymamy drzewo rozpinające $H^* = (V, F^*)$.

Zbiór $\emptyset$ jest oczywiście rozszerzalny do NDR. Ponadto wybór krawędzi dokonywany w każdej iteracji algorytmu Prima spełnia założenia twierdzenia 3.2.3 ($f$ jest najkrótszą krawędzią cięcia $\delta(W)$ rozłącznego z $F$). Zatem zgodnie z tym twierdzeniem zbiór $F \cup \{f\}$ jest rozszerzalny do NDR. W konsekwencji $H^*$ jest NDR. ■

Można pokazać, że złożoność obliczeniowa algorytmu Prima wynosi $O(n^2)$, czyli podobnie jak dla algorytmu Kruskala jest ona wielomianowa.

**Przykład 3.2.10** Dla sieci podanej w przykładach 3.2.6 i 3.2.7 metodą Prima wyznaczyć najkrótsze drzewa rozpinające.

**Przykład 3.2.11** Wybrane miasta w lubuskiem należy połączyć szybkim światłowodem. Posługując się poniższą tabelą odległości metodami Kruskala i Prima wyznaczyć graf połączeń, dla którego koszty przedsięwzięcia byłyby najniższe. Wyznaczyć również łączne najniższe koszty

przedsięwzięcia Przyjmujemy, że koszt budowy jednego kilometra światłowodu łączącego dwa miasta wynosi 1 mln zł.

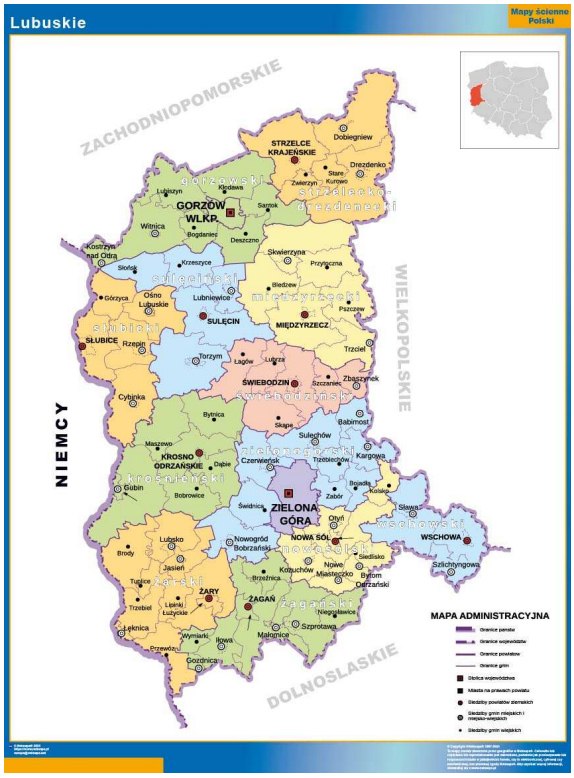
| Odległość (kilometry) | Gorzów Wlkp. | Zielona Góra | Nowa Sól | Żary | Międzyrzecz | Kostrzyn nad Odrą | Słubice |
|---|---|---|---|---|---|---|---|
| Gorzów Wlkp. | | | | | | | |
| Zielona Góra | 111 | | | | | | |
| Nowa Sól | 129 | 27 | | | | | |
| Żary | 159 | 47 | 51 | | | | |
| Międzyrzecz | 46 | 67 | 87 | 115 | | | |
| Kostrzyn nad Odrą | 45 | 136 | 156 | 144 | 83 | | |
| Słubice | 80 | 123 | 142 | 114 | 89 | 33 | |
| Gubin | 144 | 58 | 87 | 53 | 100 | 91 | 60 |

W celu skorzystania z algorytmu Kruskala uporządkujmy odległości między miastami w sposób rosnący

| Zielona Góra | Kostrzyn nad Odrą | Gorzów Wlkp. | Gorzów Wlkp. | Zielona Góra | Nowa Sól | Żary | Zielona Góra | Słubice | Zielona Góra | Gorzów Wlkp. | Międzyrzecz | Nowa Sól | Nowa Sól |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Nowa Sól | Słubice | Kostrzyn nad Odrą | Międzyrzecz | Żary | Żary | Gubin | Gubin | Gubin | Międzyrzecz | Słubice | Kostrzyn nad Odrą | Międzyrzecz | Gubin |
| 27 | 33 | 45 | 46 | 47 | 51 | 53 | 58 | 60 | 67 | 80 | 83 | 87 | 87 |

| Międzyrzecz | Kostrzyn nad Odrą | Międzyrzecz | Gorzów Wlkp. | Żary | Żary | Zielona Góra | Gorzów Wlkp. | Zielona Góra | Nowa Sól | Gorzów Wlkp. | Żary | Nowa Sól | Gorzów Wlkp. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Słubice | Gubin | Gubin | Zielona Góra | Słubice | Międzyrzecz | Słubice | Nowa Sól | Kostrzyn nad Odrą | Słubice | Gubin | Kostrzyn nad Odrą | Kostrzyn nad Odrą | Żary |
| 89 | 91 | 100 | 111 | 114 | 115 | 123 | 129 | 136 | 142 | 144 | 144 | 156 | 159 |

Zgodnie z algorytmem Kruskala otrzymamy następujące najkrótsze drzewo rozpinające



### 3.2.3 Sformułowanie w postaci zadania programowania liniowego

Wprowadzimy najpierw pewne oznaczenia.

Niech $A \subset \{1, ..., m\}$. Wektor $\chi_A = (\chi_1, ..., \chi_m) \in \mathbb{R}^m$ postaci

$$\chi_i = \begin{cases} 1 & \text{dla } i \in A \\ 0 & \text{dla } i \notin A \end{cases}$$

nazywamy *wektorem charakterystycznym* zbioru $A$.

Dalej, niech $x = (\xi_1, ...\xi_m) \in \mathbb{R}^m$. Oznaczmy $x(A) = \sum_{i \in A} \xi_i$. Mamy więc

$$x(A) = \chi_A^\top x.$$

Dany jest graf spójny $G = (V, E)$, i funkcja wag $c : E \to \mathbb{R}$. Rozpatrzmy teraz następujące zadanie programowania liniowego.

$$\begin{array}{ll} \text{minimalizować} & c^\top x \\ \text{względem} & x \in \mathbb{R}^m, \\ \text{przy ograniczeniach} & \chi_{\gamma(S)}^\top x \leq \#S - 1 \quad \forall\ S \subset V, S \neq \emptyset, S \neq V, \\ & e^\top x = \#V - 1, \\ & x \geq 0. \end{array} \tag{3.1}$$

Dla $A \subset E$ przez $s(A)$ oznaczamy liczbę składowych podgrafu $(V, A)$ grafu $G$. Zadanie (3.1) zapiszemy w następującej w postaci, której analiza będzie łatwiejsza:

$$\begin{array}{ll} \text{minimalizować} & c^\top x \\ \text{względem} & x \in \mathbb{R}^m, \\ \text{przy ograniczeniach} & \chi_A^\top x \leq \#V - s(A) \quad \forall\ A \subset E, A \neq E, \\ & e^\top x = \#V - 1, \\ & x \geq 0. \end{array} \tag{3.2}$$

**Lemat 3.2.12** *Zadania programowania liniowego* (3.1) *i* (3.2) *są sobie równoważne.*

**Dowód**. Obydwa zadania różnią się ograniczeniami. Przypuśćmy więc, że $x$ jest punktem dopuszczalnym zadania (3.2) i niech $S \subset V, S \neq \emptyset, S \neq V$. Dla $A = \gamma(S)$ mamy $s(A) = s(\gamma(S)) \geq \#(V \setminus S) + 1$. Zatem

$$\chi_{\gamma(S)}^\top x \leq \#V - s(\gamma(S)) \leq \#V - \#(V \setminus S) - 1 = \#S - 1,$$

czyli $x$ jest rozwiązaniem dopuszczalnym zadania (3.1). Niech teraz $x$ będzie rozwiązaniem dopuszczalnym zadania (3.1) i niech $A \subset E$. Dalej niech $S_1, ..., S_k$ będą wszystkimi składowymi podgrafu $(V, A)$. Oczywiście $k = s(A)$ i mamy wówczas

$$\begin{aligned} \chi_A^\top x &= (\sum_{i=1}^{k} \chi_{\gamma(S_i)})^\top x = \sum_{i=1}^{k} \chi_{\gamma(S_i)}^\top x \\ &\leq \sum_{i=1}^{k} (\#S_i - 1) = \#V - k = \#V - s(A). \end{aligned}$$

Widzimy więc, że $x$ jest punktem dopuszczalnym zadania (3.2). ■

**Twierdzenie 3.2.13** *Niech $x^*$ będzie wektorem charakterystycznym zbioru krawędzi najkrótszego drzewa $(V,T)$ rozpinającego graf $G=(V,E)$ dla funkcji kosztów $c$. Wówczas $x^*$ jest jest rozwiązaniem optymalnym zadania* (3.2).

**Dowód**. Zauważmy najpierw, że $x^*$ jako wektor charakterystyczny drzewa rozpinającego jest rozwiązaniem dopuszczalnym zadania (3.1). Fakt ten wynika stąd, że podgraf $(S,\gamma(S))$ drzewa $(V,T)$ jest lasem, zaś składowe lasu $(S_i,E_i), i=1,...,k$, są drzewami. Dla każdej takiej składowej $(S_i,E_i)$ zachodzi równość $\#E_i=\#S_i-1$. Po zsumowaniu tych równości dla wszystkich składowych otrzymamy

$$\begin{aligned}\chi^\top_{\gamma(S)}x &= x(\gamma(S))=\sum_{i=1}^{k}\#E_i \\ &= \sum_{i=1}^{k}(\#S_i-1)=\#S-k\leq \#S-1.\end{aligned} \tag{3.3}$$

Ograniczenie $e^\top x=\#V-1$ jest oczywiście spełnione, bo $(V,T)$ jest drzewem. W tej sytuacji wystarczy pokazać, że wektor charakterystyczny jakiegokolwiek najkrótszego drzewa rozpinającego jest rozwiązaniem optymalnym zadania (3.2). Pokażemy, że tak jest dla wektora charakterystycznego $x'$ NDR otrzymanego metodą Kruskala. W tym celu rozpatrzymy zadanie dualne do (3.2). Zauważmy, że ma ono postać

$$\begin{array}{rll} \text{minimalizować} & \sum_{A\subset E}(\#V-s(A))y_A & \\ \text{względem} & y\in\mathbb{R}^{2^m-1} & \\ \text{przy ograniczeniach} & \sum_{A:e\in A}y_A\geq -c_e & \forall e\in E, \\ & y_A\geq 0 & \forall A\subset E, A\neq E, \end{array} \tag{3.4}$$

gdzie $y_A$ oznacza zmienną dualną (współrzędną wektora $y$) odpowiadającą ograniczeniu $\chi^\top_A x\leq \#V-s(A)$. Zwróćmy uwagę na to, że zmienna $y_E$ nie musi być nieujemna, gdyż odpowiadające jej ograniczenie w zadaniu pierwotnym jest równościowe. W celu pokazania, że $x'$ jest rozwiązaniem optymalnym zadania (3.2) skorzystamy z twierdzenia o komplementarności. Zgodnie z nim para rozwiązań dopuszczalnych $(x,y)$ zadania pierwotnego i dualnego jest parą rozwiązań optymalnych wtedy i tylko wtedy gdy dla dodatnich współrzędnych $x_i$ wektora $x$ w odpowiadających im ograniczeniach zadania dualnego zachodzą równości i dla dodatnich współrzędnych $y_A$ wektora $y$ w odpowiadających im ograniczeniach zadania pierwotnego zachodzą równości. Mając dany wektor $x'$ (otrzymany metodą Kruskala) zdefiniujemy teraz rozwiązanie dopuszczalne $y$ zadania dualnego (3.4). Niech krawędzie $e_1,...,e_m$ grafu $G$ będzie ustawione w takim samym porządku, jak w algorytmie Kruskala. Oznaczmy $A_i=\{e_1,...,e_i\}, i=1,...,m$. Połóżmy $y_{A_i}=c_{e_{i+1}}-c_{e_i}$ dla $i=1,...,m-1$, $y_{A_m}=-c_{e_m}$ oraz $y_A=0$ dla $A\neq A_i$. Zauważmy, że tak określony wektor $y$ jest rozwiązaniem dopuszczalnym zadania dualnego (3.4). Nieujemność współrzędnych $y_A$ dla $A\neq E$ wynika z ich definicji oraz z nierówności $c_{e_{i+1}}\geq c_{e_i}, i=1,...,m-1$. Dalej mamy dla $e=e_j$

$$\sum_{A:e\in A}y_A=\sum_{i=j}^{m}y_{A_i}=\sum_{i=j}^{m-1}(c_{e_{i+1}}-c_{e_i})-c_{e_m}=-c_{e_j}=-c_e.$$

Zatem $y$ jest rozwiązaniem dopuszczalnym zadania dualnego (3.4). Ponieważ w ograniczeniach tego zadania zachodzą równości, więc spełniony jest pierwszy warunek w twierdzeniu o

komplementarności. Sprawdzimy teraz, że spełniony jest również drugi warunek. Niech więc zmienna dualna $y_A > 0$. Jest jasne, że $A = A_i$ dla pewnego $i$, $i = 1, ..., m$, gdyż dla innych zbiorów $A$ zachodzi równość $y_A = 0$. Pokażemy, że dla $x = x'$ w odpowiadającym tej zmiennej ograniczeniu zachodzi równość, czyli $\chi_{A_i}^\top x' = \#V - s(A_i)$. Ponieważ $x'$ jest wektorem charakterystycznym NDR $(V, T)$, więc $\chi_{A_i}^\top x' = \#(T \cap A_i)$. Zauważmy ponadto, że zgodnie z algorytmem Kruskala $s(A_i) = s(T \cap A_i)$. Oba zbiory $A_i$ oraz $T \cap A_i$ różnią się bowiem tylko krawędziami niedołączonymi do zbioru krawędzi grafu $H$ w trakcie realizacji algorytmu Kruskala. Jeśli natomiast krawędź $e_j \in A_i$ nie została dołączona do tego zbioru, to mogło to nastąpić tylko z tego powodu, że oba końce tej krawędzi należą do jednej składowej tego grafu. Tak więc oba zbiory $A_i$ i $T \cap A_i$ mają tę samą liczbę składowych. Ponieważ $T \cap A_i$ jest lasem, więc podobnie jak w (3.3) zachodzi równość $\#(T \cap A_i) = \#V - s(T \cap A_i)$. Pokazaliśmy więc, że oba warunki twierdzenia o komplementarności są spełnione. Zatem para $(x', y)$ jest parą rozwiązań optymalnych. ■

**Uwaga 3.2.14** Dowód powyższego twierdzenia może służyć jednocześnie jako dowód tego, że algorytm Kruskala generuje najkrótsze drzewo rozpinające.

## 3.3 Najkrótsze drogi

Przypuśćmy, że chcemy znaleźć najkrótszą (najtańszą, najszybszą) drogę z Zielonej Góry do Białegostoku, przy czym możemy poruszać się wyłącznie drogami krajowymi przestrzegając przepisów kodeksu drogowego (ograniczenia prędkości, drogi jednokierunkowe, zakazy ruchu, itp.). Sieć dróg możemy przedstawić w postaci pewnego grafu, którego wierzchołki są skrzyżowaniami, zaś krawędzie są odcinkami dróg łączącymi sąsiednie skrzyżowania. Zgodnie z przepisami ruchu drogowego jest zakaz poruszania się pod prąd na jednokierunkowych odcinkach dróg. Ponadto koszt (czas) przejazdu na danym odcinku może się różnić w zależności od kierunku jazdy. Prowadzi to do tego, że nasze dalsze rozważania będziemy musieli prowadzić dla *grafu skierowanego*, czyli grafu $G = (V, E)$ w którym każdej krawędzi $e = uv$ (będziemy ja nazywać *łukiem*) wyróżniono jej początek $u$ i koniec $v$. O ile dla grafu nieskierowanego para wierzchołków $u, v$ jest nieuporządkowana (czyli krawędzie $uv$ i $vu$ są jednakowe), o tyle dla grafu skierowanego para wierzchołków $u, v$ (będziemy je nazywać *węzłami*) jest uporządkowana, czyli $uv \neq vu$. Poza pewnymi szczegółami, które będą omówione poniżej, pojęcia i oznaczenia zdefiniowane dla grafów nieskierowanych dotyczą również grafów skierowanych.

Dla grafu skierowanego $G = (V, E)$:

1. Przez zapis $e = uv \in E$ rozumiemy, że łuk $e$ ma *początek* $u \in V$ i *koniec* $v \in V$.

2. Łuki grafu $G$ nazywamy łukami *równoległymi* jeśli mają te same początki i te same końce.

3. Istnieje graf nieskierowany o zbiorze wierzchołków $V$ i zbiorze krawędzi $E' = \{uv : uv \in E$ lub $vu \in E\}$.

4. $G$ może być prosty jako graf nieskierowany nie będąc jednocześnie prostym jako graf skierowany.

5. *drogą* (*skierowaną*) $P = (e_1, ..., e_k)$ w grafie $G = (V, E)$ o początku w węźle $v_0 \in V$ i końcu w węźle $v_k \in V$ (lub $(v_0, v_k)$-*drogą*) nazywamy ciąg łuków $e_1 = v_0v_1, ..., e_k = v_{k-1}v_k$ tego grafu (koniec $i$-tego łuku jest początkiem następnego, $i = 1, ..., k-1$).

6. Drogę (skierowaną) $P$ w grafie skierowanym $G$ możemy utożsamiać z podgrafem skierowanym $G' = (\{v_0, ..., v_k\}, \{e_1, ..., e_k\})$, w którym wyróżniono początek $v_0$ i koniec $v_k$ i, w którym zachodzi opisany wyżej związek między węzłami i łukami.

7. Droga $P = (v_0v_1, ..., v_{k-1}v_k)$ nazywa się *prosta*, jeśli węzły $v_0, ..., v_k$ są różne.

8. Droga $P = (v_0v_1, ..., v_{k-1}v_k)$ nazywa się *zamknięta*, jeśli $v_0 = v_k$.

9. Jeśli w grafie $G$ istnieje $(u, v)$-droga, to nie musi w nim istnieć $(v, u)$-droga (w konsekwencji relacja: $u \sim v \Leftrightarrow$ *istnieje* $(u, v)$-*droga*, nie jest symetryczna, czyli nie jest relacją typu równoważności).

10. $G$ nazywamy *spójnym w sensie mocnym* jeśli dla dowolnych $u, v \in V$ istnieje $(u, v)$-droga.

11. $G$ nazywamy *spójnym w sensie słabym* jeśli dla dowolnych $u, v \in V$ istnieje $(u, v)$-droga lub $(v, u)$-droga.

12. *Cyklem* (*skierowanym*) nazywamy drogę skierowaną zamkniętą i prostą.

13. *$G$ nazywa się drzewem* (*skierowanym*, o *korzeniu* $r$) jeśli dla każdego $v \in V$ istnieje dokładnie jedna $(r, v)$-droga skierowana.

14. Drzewo skierowane $H = (V, T)$ nazywamy *skierowanym drzewem rozpinającym* (graf $G$).

15. *Siecią skierowaną* nazywamy układ $(V, E, c)$, gdzie $(V, E)$ jest grafem skierowanym zaś $c : E \to \mathbb{R}$ – funkcją długości.

Rozważmy następujące zadanie zwane problemem najkrótszej drogi:

> Dana jest sieć skierowana $(V, E, c)$ i węzeł $r \in V$. Dla dowolnego węzła $v \in V$ należy znaleźć najkrótszą $(r, v)$-drogę (o ile taka droga w ogóle istnieje).

W dalszej części tego ustępu będziemy zakładać, że dla dowolnego $v \in V$ istnieje $(r, v)$-droga. Założenie to nie ogranicza ogólności rozważań, bo w razie potrzeby można zbiór łuków uzupełnić o łuki $e = rv \notin E$ o wspólnej długości $d$ na tyle dużej, że jeśli taka dołączony łuk byłby najkrótszą $(r, v)$-drogą, to w wyjściowym grafie nie istniałaby $(r, v)$-droga. (Ile powinna wynosić długość $d$?) W ten sposób metoda wyznaczania najkrótszej drogi stwierdzałaby jednocześnie, czy taka droga w ogóle istnieje.

### 3.3.1 Potencjały dopuszczalne

Dana jest sieć skierowana $(V, E, c)$ i węzeł $r \in V$. Przypuśćmy, że dla $v \in V$ wyznaczyliśmy $(r, v)$-drogę o długości $y_v$. Zauważmy, że jeśli $(r, w)$-droga jest najkrótszą drogą w tym grafie łączącą korzeń $r$ z węzłem $w$, to spełnione są nierówności

$$y_v + c_{vw} \geq y_w \text{ dla każdego łuku } vw \in E.$$

Gdyby bowiem dla pewnego łuku $vw \in E$ zaszła nierówność przeciwna, to uzupełniając $(r, v)$-drogę o długości $y_v$ o ten łuk, której długość wynosi $c_{vw}$ otrzymalibyśmy $(r, w)$-drogę o długości $y_v + c_{vw}$ mniejszej niż $y_w$. Zatem dla każdego rozwiązania problemu najkrótszej drogi muszą być spełnione warunki

$$\begin{aligned} y_v + c_{vw} &\geq y_w \text{ dla każdego łuku } vw \in E && \text{(3.5a)} \\ \text{oraz } y_r &= 0. && \text{(3.5b)} \end{aligned}$$

Wektor $y = (y_v : v \in V)$ spełniający te warunki nazywa się *potencjałem dopuszczalnym*. Dla drogi $P = (e_1, ..., e_k)$ i dla funkcji długości $c : E \to \mathbb{R}$ długość tej drogi wyraża się wzorem $c(P) = \chi_P^\top c = \sum_{i=1}^k c_{e_i}$. Potencjały dopuszczalne posiadają następującą własność.

**Twierdzenie 3.3.1** *Niech $y$ będzie potencjałem dopuszczalnym dla sieci skierowanej $(V, E, c)$, niech $v \in V$ i niech $P$ będzie $(r, v)$-drogą skierowaną. Wówczas $c(P) \geq y_v$.*

**Dowód**. Niech $P = (e_1, ..., e_k)$, gdzie $e_i = v_{i-1}v_i, i = 1, ..., k$ i $v_0 = r$ oraz $v_k = v$. Na mocy (3.5) mamy

$$c(P) = \sum_{i=1}^{k} c_{e_i} \geq \sum_{i=1}^{k} (y_{v_i} - y_{v_{i-1}}) = y_{v_k} - y_{v_0} = y_v$$

■

**Uwaga 3.3.2** Jeśli $P = (v_0v_1, ..., v_{k-1}v_k)$, gdzie $v_0 = r$, $v_k = v$, jest najkrótszą $(r, v)$-drogą, to $P' = (v_0v_1, ..., v_{k-2}v_{k-1})$ jest najkrótszą $(r, v_{k-1})$-drogą. Dla każdego węzła $v$ możemy więc wskazać węzeł $v'$ bezpośrednio poprzedzający go w najkrótszej $(r, v)$-drodze. Takich węzłów może być wprawdzie więcej (bo najkrótszych dróg może być więcej), w takiej sytuacji przypuśćmy jednak, że wybraliśmy jeden z nich (oznaczmy go symbolem $p(v)$) traktując go jako bezpośredniego poprzednika węzła $v$. Węzły $v' = p(v)$ i $v$ wyznaczają łuk $v'v$ zaś wszystkie takie łuki dla $v \in V$ określają skierowane drzewo rozpinające $H = (V, T)$ o korzeniu $r$.

### 3.3.2 Algorytm Forda

Zauważmy, że dysponując potencjałem dopuszczalnym $y$ twierdzenie 3.3.1 daje nam możliwość stwierdzenia, czy dana $(r, v)$-droga skierowana $P$ jest najkrótsza. Jeśli bowiem $c(P) = y_v$, to na mocy twierdzenia 3.3.1 droga ta jest najkrótszą $(r, v)$-drogą. Wystarczy więc skonstruować potencjał dopuszczalny oraz określić bezpośredniego poprzednika $p(u)$ takiego, że łuk $p(u)u \in E$ dla każdego węzła $u \in V \setminus \{r\}$. Najkrótszą $(r, v)$-drogę można wtedy skonstruować od końca do początku. Zwróćmy uwagę na to, że w ten sposób skonstruujemy najkrótsze $(r, v)$-drogi dla wszystkich $v \in V$.

**Wniosek 3.3.3** *Istnienie potencjału dopuszczalnego dla sieci skierowanej $(V, E, c)$ i węzła $r \in V$ jest równoważne istnieniu dla tej sieci najkrótszych $(r, v)$-dróg dla wszystkich $v \in V$.*

Opiszemy teraz, jak skonstruować potencjał dopuszczalny. Przypuśćmy, że mamy dany pewien wektor $y \in \mathbb{R}^n$. Może nim być na przykład wektor o współrzędnych $y_v$ określających długości $(r, v)$-dróg dla $v \in V$ o łukach należących do pewnego skierowanego drzewa rozpinającego $H = (V, T)$ o korzeniu $r$. Każdy węzeł $v$ takiego drzewa różny od $r$ ma swojego bezpośredniego poprzednika $p(v)$ takiego, że $p(v)v \in T$. Można również przyjąć $y_r = 0$ oraz $y_v = +\infty$ i $p(v) = -1 \notin V$ dla dowolnego $v \neq r$. Przyjęcie $y_v = +\infty$ oznacza, że na razie nie znamy $(r, v)$-drogi, zaś przyjęcie $p(v) = -1$ oznacza, że węzeł $v$ nie ma jeszcze swojego poprzednika. Jeśli wektor $y$ spełnia warunki (3.5), to jest on potencjałem dopuszczalnym. Jeśli tylko (3.5a) jest spełniony, to po odjęciu $y_r$ od dowolnej współrzędnej wektora $y$ otrzymamy oczywiście potencjał dopuszczalny. Przypuśćmy więc, że wektor $y$ jest taki, że $y_v + c_{vw} < y_w$ dla pewnego łuku $vw \in E$. Łuk taki nazywamy *niepoprawnym*. Wówczas zastępując $y_w$ przez $y_v + c_{vw}$ otrzymamy nowy wektor $y$. Czynność tę nazywamy *poprawką*. Po jej dokonaniu przyjmujemy $v = p(w)$, czyli węzeł $v$ określamy jako bezpośredniego poprzednika węzła $w$. Dla poprawionego wektora $y$ sprawdzamy znowu, czy spełnia on warunek (3.5a) dokonując ewentualnej poprawki, itd. W ten sposób opisaliśmy działanie algorytmu Forda wyznaczenia najkrótszych dróg dla sieci skierowanej $(V, E, c)$.

**Algorytm 3.3.4 (Ford)**

**Wejście**: Sieć skierowana $(V, E, c)$ i węzeł $r \in V$ takie, że dla każdego $v \in V$ istnieje $(r, v)$-droga.

**Wyjście**: Skierowane drzewo rozpinające $H = (V, T)$ o korzeniu $r$ i potencjał dopuszczalny $y$.

**Krok 0** (*inicjalizacja*)

a) Określić skierowane drzewo rozpinające $H = (V, T)$ o korzeniu $r$,

b) określić wektor $y$, którego współrzędne $y_v$ określają długości $(r, v)$-dróg zawartych w drzewie $H$, $v \in V$,

c) dla każdego $v \in V \setminus \{r\}$ określić bezpośredniego poprzednika $p(v) \in V$ takiego, że łuk $p(v)v \in T$,

d) przyjąć $p(r) = 0 \notin V$.

**Krok 1** (*kryterium zatrzymania*) Jeśli wektor $y$ jest potencjałem dopuszczalnym, to algorytm zatrzymuje się.

**Krok 2** (*poprawka*)

a) Dla niepoprawnego łuku $vw$ dokonać poprawki:

– przyjąć $y_w = y_v + c_{vw}$,

– przyjąć $p(w) = v$,

b) przejść do kroku 1.

**Przykład 3.3.5** Zastosujemy algorytm Forda wyznaczenia najkrótszych dróg dla sieci skierowanej$(V, E, c$ i węzła $a \in V$, gdzie $V = \{a, b, c, d, e, f, g, h, i\}$, zbiór łuków $E$ i funkcja długości $c : E \to \mathbb{R}$ podane są w postaci następującej tablicy

| | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ | $g$ | $h$ | $i$ |
|---|---|---|---|---|---|---|---|---|---|
| $a$ | | 3 | 1 | | | | | | |
| $b$ | 3 | | 1 | 3 | 4 | | | | |
| $c$ | 1 | 1 | | | 1 | 2 | | | |
| $d$ | | 3 | | | 5 | | 4 | | |
| $e$ | | 4 | 1 | 5 | | 3 | 7 | 1 | |
| $f$ | | | 2 | | 3 | | | 5 | |
| $g$ | | | | 4 | 7 | | | 2 | 2 |
| $h$ | | | | | 1 | 5 | 2 | | 1 |
| $i$ | | | | | | | 2 | 1 | |

Element tej tablicy jest liczbą wtedy i tylko wtedy, gdy odpowiadające mu węzły są połączone łukiem i wówczas liczba ta wskazuje długość tego łuku. Po zastosowaniu algorytmu Forda otrzymamy następujący wektor potencjałów $y = (y_a, ..., y_i) = (0, 2, 1, 5, 2, 3, 5, 3, 4)$ i wektor bezpośrednich poprzedników $p = (p(a), ..., p(i)) = (0, c, a, b, c, c, h, e, h)$. Wektor $p$ określa w sieci $(V, E, c)$ najkrótsze drogi łączące węzeł $a$ z pozostałymi węzłami natomiast wektor $y$ określa długości tych dróg.

Do tej pory w sposób milczący przyjmowaliśmy, że długości wszystkich łuków są nieujemne. Zobaczmy jednak, co może się wydarzyć w trakcie realizacji algorytmu Forda, jeśli założenie to nie będzie spełnione.

**Przykład 3.3.6** Zmodyfikujmy poprzedni przykład przyjmując $c_{be} = -4$ i pozostawiając pozostałe wartości bez zmian. Zauważmy, że w tej sytuacji algorytm Forda nie zatrzyma się w skończonej liczbie iteracji, przy czym niektóre ze współrzędnych wektora $y$ będą dążyć do $-\infty$. Nie powinno to być niespodzianką, gdyż wędrowanie drogą zamkniętą $(cb, be, ec)$ zmniejsza jej długość o 2 jednostki po każdorazowym wykonaniu jednego cyklu. Ponieważ dla dowolnego węzła $v \in V$ istnieje $(a, v)$-droga zawierająca łuk $be$, więc dowolną $(a, v)$-drogę można poprawiać w nieskończoność tak, że jej długość będzie dążyć do $-\infty$.

Powyższy przykład wskazuje na to, że jeśli graf $G$ zawiera cykl o ujemnej długości, to dla dowolnego $r \in V$ istnieje węzeł $v \in V$, dla którego nie ma najkrótszej $(r, v)$-drogi. Okaże się, że brak takiego cyklu jest warunkiem wystarczającym na to, aby algorytm Forda wyznaczył najkrótsze drogi w grafie $G$. Nie musimy natomiast ograniczać się do nieujemnych funkcji długości, gdyż w wielu zastosowaniach funkcje te mogą przyjmować również ujemne wartości.

**Przykład 3.3.7** Dany jest koszyk $V$ składający się z $n$ walut, zbiór par uporządkowanych $E \subset V \times V$. Ponadto dla dowolnej pary walut $(u, w) \in E$ dany jest kurs wymiany $r_{uw} > 0$. Wielkość $r_{uw}$ wyraża ilość jednostek waluty $w$, które otrzymamy ze sprzedaży jednostki waluty $u$. Przypuśćmy, że dokonujemy ciągu $k$ wymian $(v_1v_2, v_2v_3, ..., v_kv_{k+1})$. Wówczas za jednostkę waluty $v_1$ otrzymamy $r_{v_1v_2} \cdot ... \cdot r_{v_kv_{k+1}}$ jednostek waluty $v_{k+1}$. Jeśli więc $v_{k+1} = v_1$, to rozsądnym jest założenie, że $r_{v_1v_2} \cdot ... \cdot r_{v_kv_1} \leq 1$. W przeciwnym wypadku otrzymalibyśmy bowiem w wyniku tych operacji $r_{v_1v_2} \cdot ... \cdot r_{v_kv_1} > 1$ jednostek waluty $v_1$ za jednostkę tej waluty. Byłby to niezły sposób na darmowe pomnażanie swojego majątku dopóty, dopóki banki nie zorientowałyby się, że należy zmienić kursy wymiany. Istnienie ciągu takich wymian walut (czy też ogólniej towarów) związane jest z pojęciem *arbitrażu* pochodzącym z ekonomii, gdzie zakłada się, że rynek jest bezarbitrażowy. W przypadku rynku wymiany walut oznacza to, że $r_{v_1v_2} \cdot ... \cdot r_{v_kv_1} \leq 1$ dla dowolnych walut $v_1, ..., v_k$ i dla dowolnego $k$. Przypuśćmy wobec tego, że dla danego bezarbitrażowego rynku walut i dla ustalonej waluty, powiedzmy $r$ poszukujemy najkorzystniejszego sposobu jej wymiany na inne waluty. Dla określonej waluty $v$ i dla określonego ciągu wymian $(v_1v_2, v_2v_3, ..., v_kv_{k+1})$, gdzie $v_1 = r$ i $v_{k+1} = v$, otrzymamy $r_{v_1v_2} \cdot ... \cdot r_{v_kv_{k+1}}$ jednostek waluty $v$ za jednostkę waluty $r$. Widzimy więc, że znalezienie jak najkorzystniejszej transakcji, prowadzi do zadania wyznaczenia w grafie $G = (V, E)$ drogi $(v_1v_2, v_2v_3, ..., v_kv_{k+1})$, dla której $r_{v_1v_2} \cdot ... \cdot r_{v_kv_{k+1}}$ przyjmuje największą wartość. Wprowadźmy funkcję $c : E \to \mathbb{R}$ określoną równością $c_{uw} = -\log r_{uw}$ . Wówczas korzystając z własności funkcji logarytmicznej otrzymujemy dla drogi $P = (v_1v_2, v_2v_3, ..., v_kv_{k+1})$ równości

$$c(P) = \sum_{i=1}^{k+1} c_{v_iv_{i+1}} = -\sum_{i=1}^{k+1} \log r_{v_iv_{i+1}} = -\log r_{v_1v_2} \cdot ...r_{v_kv_{k+1}}.$$

Nasze zadanie sprowadza się więc do wyznaczenia w sieci $(V, E, c)$ najkrótszej $(r, v)$-drogi. Widzimy ponadto, że założenie o bezarbitrażowym rynku walut jest równoważne założeniu o braku w grafie $G$ cyklu o ujemnej długości (mierzonej oczywiście funkcją $c$).

**Twierdzenie 3.3.8** *Dla sieci skierowanej $(V, E, c)$ i węzła $r \in V$ istnieje potencjał dopuszczalny wtedy i tylko wtedy, gdy graf $G = (V, E)$ nie zawiera cyklu o ujemnej długości.*

**Dowód**. $\Longrightarrow$ Przypuśćmy, że istnieje potencjał dopuszczalny $y$ i, że graf $G$ zawiera cykl $C = (v_1v_2, ...v_{k-1}v_k, v_kv_1)$ o ujemnej długości. Zgodnie z wnioskiem 3.3.3 dla węzła $v_1$ istnieje najkrótsza $(r, v_1)$-droga $P$. Dokładając do niej cykl $C$ otrzymamy $(r, v_1)$-drogę o długości krótszej niż wynosi długość drogi $P$. Otrzymaliśmy więc sprzeczność, co dowodzi konieczności warunku.

$\Longleftarrow$ Zauważmy, że jeśli $G$ nie zawiera cyklu o ujemnej długości, to poszukiwanie najkrótszych dróg wystarczy ograniczyć do dróg prostych, czyli dróg bez powtarzających się węzłów. Takich dróg jest skończenie wiele, czyli istnieją wśród nich drogi najkrótsze, co na mocy wniosku 3.3.3 jest równoważne istnieniu potencjału dopuszczalnego. ■

**Twierdzenie 3.3.9** *Niech $(V, E, c)$ będzie siecią skierowaną i niech $r \in V$ będzie węzłem takim, że dla każdego $v \in V$ istnieje $(r, v)$-droga. Jeśli graf $G = (V, E)$ nie zawiera cyklu o ujemnej długości, to algorytm Forda zatrzymuje się po wykonaniu skończenie wielu iteracji wyznaczając potencjał dopuszczalny $y$ i w konsekwencji najkrótszą $(r, v)$-drogę o długości $y_v$ dla każdego $v \in V$.*

**Dowód**. Jeśli sieć $G = (V, E, c)$ nie zawiera cyklu o ujemnej długości, to węzły $p(v)$ wyznaczane w algorytmie Forda nie powtarzają się. Oznacza to, że drogi wyznaczane przez ten algorytm są proste. Ponieważ dróg prostych jest skończenie wiele, a każda iteracja algorytmu Forda skraca przynajmniej jedną $(r, v)$-drogę, więc po wykonaniu skończenie wielu iteracji wektor $p$ bezpośrednich poprzedników wyznaczy drzewo skierowane o korzeniu $r$, najkrótsze $(r, v)$-drogi i potencjał dopuszczalny $y$. ■

**Twierdzenie 3.3.10** *Złożoność obliczeniowa algorytmu Forda wynosi* $O(n^2)$.

**Dowód** pomijamy.

### 3.3.3 Sformułowanie w postaci zadania programowania liniowego

### 3.3.4 Algorytm Dijkstry