Wojciech Kordecki
Anna Łyczkowska-Hańćkowiak

MATEMATYKA DYSKRETNA
DLA INFORMATYKÓW

Helion

Redaktor prowadzący: Małgorzata Kulik

Projekt okładki: Studio Gravite / Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Grafika na okładce została wykorzystana za zgodą Shutterstock.com

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/madyin
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-4011-4

Printed in Poland.

Spis treści

8.2. Minimalne drzewa spinające i minimalne drogi

Niech $G = (V, E)$ będzie grafem nieskierowanym bez pętli o n wierzchołkach. Na zbiorze krawędzi E określamy funkcję o wartościach nieujemnych, $w : E \to \mathbb{R}^+$. Wartość $w(e)$ nazywamy wagą krawędzi e. Jeśli $e = \{u, v\}$, to piszemy po prostu $w(u, v)$. Graf G z określoną funkcją w nazywa się grafem ważonym.

Minimalne drzewo spinające

Minimalnym drzewem spinającym spójnego grafu G nazywamy drzewo spinające T takie, że jego waga

$$w(T) = \sum_{e \in T} w(e) \tag{8.2.1}$$

jest najmniejsza. Gdy G nie jest spójny, to analogicznie minimalnym lasem spinającym T grafu G nazywamy las spinający T_G taki, że waga $w(G)$ określona wzorem (8.2.1) jest najmniejsza.

Droga minimalna

Minimalną drogą łączącą wierzchołki u oraz v w grafie G, $u \neq v$, nazywamy drogę $P(u, v)$ taką, że jej waga

$$w(P(u, v)) = \sum_{e \in P(u,v)} w(e) \tag{8.2.2}$$

jest najmniejsza. Gdy wierzchołki u oraz v nie są połączone drogą (należą do różnych składowych grafu G), to przyjmujemy $P(u, v) = \infty$.

Dla sieci komunikacyjnych lub przepływowych wagi mogą reprezentować pewne wielkości fizyczne, takie jak koszt, odległość, efektywność, pojemność lub niezawodność, przypisane odpowiednim krawędziom. Prosty graf ważony może reprezentować jego macierz wag $W = [w_{ij}]$, gdzie w_{ij} jest wagą krawędzi e_{ij} między wierzchołkiem v_i oraz wierzchołkiem v_j. Wagom nieistniejących krawędzi nadaje się w zależności od zastosowań wartość ∞ lub 0.

Omówimy dalej dwa algorytmy służące do rozwiązywania problemu minimalnego drzewa spinającego. Są nimi algorytm Kruskala i algorytm Prima. Oba algorytmy są przykładami zachłannych algoryt-

mów optymalizacyjnych, które zostaną przedstawione bardziej ogólnie w rozdziale 11. W każdym kroku takiego algorytmu musimy dokonać jednego z wielu możliwych wyborów. Stosując strategię zachłanną, dokonujemy wyboru, który jest w danej chwili najlepszy.

Algorytm Kruskala

Algorytm Kruskala[37] wyznaczający minimalne drzewo spinające ma następującą postać. Niech G będzie grafem o n wierzchołkach. Wtedy następujący algorytm daje rozwiązanie problemu minimalnego drzewa lub lasu spinającego:

Algorytm 8.2.1

1. *Sortujemy krawędzie niemalejąco według wag.*
2. *Wybieramy krawędź e_1 o najmniejszej wadze.*
3. *Dodajemy krawędzie $e_2, e_3, \ldots, e_{n-1}$, wybierając za każdym razem nową krawędź o najmniejszej możliwej wadze, która nie tworzy cyklu z dotychczas wybranymi krawędziami e_i.*

Podgraf T grafu G, którego krawędziami są $e_1, e_2, \ldots, e_{n-1}$, jest szukanym drzewem spinającym. Jeśli G jest grafem niespójnym, to algorytm Kruskala wyznacza minimalny las.

Algorytm Prima

Metodą wyznaczania minimalnego drzewa spinającego grafu, która nie wymaga ani sortowania, ani sprawdzania istnienia cykli w każdym kroku, jest *algorytm Prima*[57], zwany też *algorytmem najbliższego sąsiada*. Dla grafu spójnego przebiega on w następujący sposób.

Algorytm 8.2.2

1. *Niech $T = \emptyset$.*
2. *Wybieramy dowolny wierzchołek początkowy $v_1 \in T$ i dołączamy go do T.*
3. *Wybieramy dowolną krawędź incydentną z v_1 o najniższej wadze i drugi koniec tej krawędzi dołączamy do zbioru T.*
4. *Kolejno dla $i = 2, \ldots, n-1$ wybieramy dowolną krawędź e_i incydentną z dowolnym wierzchołkiem ze zbioru T o najniższej wadze taką, że jej drugi koniec nie należy do T, i dołączamy ten koniec do zbioru T.*
5. *Algorytm kończy się, gdy $T = V$.*

Powyższa procedura wyznacza minimalne drzewo spinające zawierające krawędzie $e_1, e_2, \ldots, e_{n-1}$. Najgorszy przypadek dla powyższego algorytmu pojawia się, gdy graf G jest grafem pełnym. W takiej

sytuacji w każdym kroku algorytmu musimy przeprowadzić maksymalną liczbę porównań, aby znaleźć najbliższy sąsiedni wierzchołek. Jeśli G jest grafem niespójnym, to dla wyznaczenia minimalnego lasu należy algorytm Prima zastosować dla każdej składowej oddzielnie.

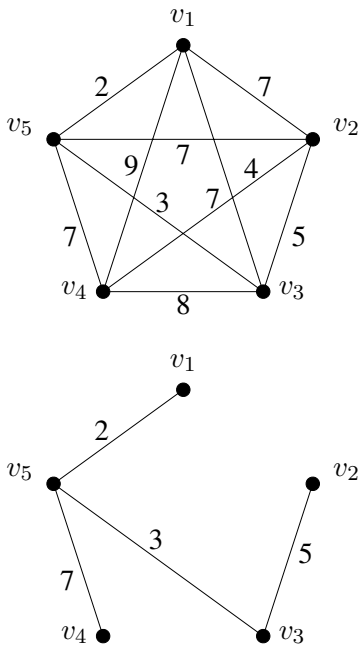
Algorytm zachłanny generujący minimalne drzewo spinające zawdzięczamy J. B. Kruskalowi (1956). Algorytm wyznaczający minimalne drzewo spinające metodą najbliższego sąsiada został odkryty w 1930 roku przez czeskiego matematyka V. Jarníka[32], a następnie ponownie odkryty w 1957 roku przez R. C. Prima oraz niezależnie w 1959 roku przez holenderskiego informatyka E. Dijkstrę[11].

Przykład 8.2.1

Na rysunku 8.4 przedstawiony jest graf $G = K_5$ z wagami krawędzi. Oznaczmy $e_{ij} = (v_i, v_j)$. Krawędzie porządkujemy według rosnących wag:

$$e_{15}, e_{35}, e_{13}, e_{23}, e_{12}, e_{25}, e_{24}, e_{45}, e_{34}, e_{14}\,.$$

Zgodnie z algorytmem Kruskala wybieramy kolejne krawędzie, które nie tworzą cykli z poprzednio wybranymi. Są to krawędzie $e_{15}, e_{35}, e_{23}, e_{45}$. Waga tego drzewa wynosi $w\,(T) = 2+3+5+7 = 17$ i jest to minimalne drzewo spinające.



Rysunek 8.4. Graf do przykładu 8.2.1 i jego minimalne drzewo spinające

Stosując algorytm Prima, zaczynamy od dowolnego wierzchołka, na przykład od v_2. Krawędź o najmniejszej wadze do pozostałych wierzchołków to e_{23}. Od zbioru wierzchołków $\{v_2, v_3\}$ do pozostałych prowadzi krawędź e_{35}, od zbioru $\{v_2, v_3, v_5\}$ do pozostałych prowadzi krawędź e_{15}, a w końcu wybieramy krawędź e_{45}. Dostaliśmy to samo drzewo co otrzymane za pomocą algorytmu Kruskala.

Drzewa minimalne mogą być różne w zależności od użytego algorytmu, sposobu posortowania przy algorytmie Kruskala (gdy wagi niektórych krawędzi są identyczne), wierzchołka startowego i kolejnych wyborów przy algorytmie Prima. Niemniej wagi tak otrzymanych drzew minimalnych są oczywiście takie same.

Długością ważoną drogi $(u_0, u_1, \ldots, u_n)$ w grafie $G(V, E)$ o nieujemnych wagach krawędzi (łuków) $e \in E$ jest suma wag krawędzi tej drogi. Do znajdowania najkrótszej drogi ważonej z ustalonego wierzchołka u w grafie $G(V, E)$ o nieujemnych wagach krawędzi $e \in E$ służy algorytm opracowany przez E. Dijkstrę.

Algorytm Dijkstry

Algorytm ten znajduje w grafie wszystkie najkrótsze drogi pomiędzy wyróżnionym wierzchołkiem u a wszystkimi pozostałymi, dodatkowo wyliczając długość każdej z tych dróg. Algorytm Dijkstry jest również przykładem algorytmu zachłannego (zobacz na przykład [7], [13]). Sformułujemy go dla grafów nieskierowanych.

Algorytm 8.2.3

Tworzymy dwa zbiory wierzchołków Q i S. W trakcie algorytmu będziemy tworzyć funkcję $d(v) \geqslant 0$ określoną na V i funkcję $p(v)$ określoną na $V \setminus u$ o wartościach w V. Wierzchołek u jest jedynym wierzchołkiem początkowym.

1. *Niech $Q = V$, $S = \emptyset$.*
2. *Dla każdego $v \in V \setminus u$ przyjmujemy $d(v) = \infty$ oraz $d(u) = 0$.*
3. *Funkcja $p(v)$ jest niezdefiniowana dla wszystkich $v \in V$.*
4. *Powtarzamy poniższe kroki algorytmu, aż $Q = \emptyset$.*
   (a) *Wybieramy w Q wierzchołek o najmniejszym $d(v)$, usuwamy go z Q i dodajemy do S.*
   (b) *Dla każdego $t \in \Gamma(v) \cap Q$, jeśli $d(t) > d(v) + w(v, t)$, to $d(t) = d(v) + w(v, t)$.*
   (c) *Określamy $p(t) = v$.*

W wyniku działania tego algorytmu ciąg wierzchołków $v, v_1 = p(v)$, $v_2 = p(v_1), \ldots, u = p(v_m)$ jest najkrótszą drogą z wierzchołka v do początkowego wierzchołka u.

Uwaga. Dla wybrania z Q wierzchołka v o najmniejszym $d(v)$ najlepiej przyjąć, że Q jest uporządkowany tak jak przy przeszukiwaniu grafu metodą BFS (algorytm 8.1.1).

Przykład 8.2.2

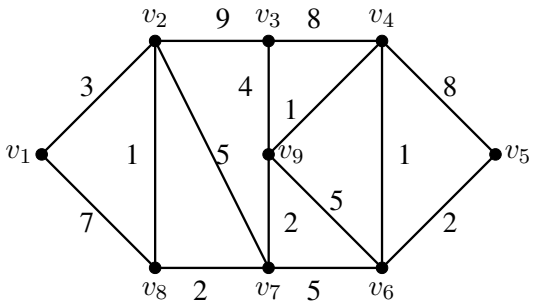
Dla grafu pokazanego na rysunku 8.5 najkrótszą drogą ważoną od v_1 do v_5 jest ciąg $(v_1, v_2, v_8, v_7, v_9, v_4, v_6, v_5)$. Ważona długość tej drogi jest równa 12, a długość nieważona jest równa 7. Długość zaś najkrótszej drogi nieważonej jest równa 4, a taką drogą jest na przykład ciąg v_1, v_2, v_3, v_4, v_5. Długość ważona tej drogi wynosi 28.

Aby zastosować algorytm Dijkstry, ustalamy najpierw algorytmem BFS zbiór (kolejkę) Q (wiersz pierwszy) i nadajemy wartości z kroku 2 w algorytmie (wiersz drugi).

W kolejnych krokach algorytmu, idąc z kolejnych wierzchołków v z kolejki Q, wierzchołki $t \in V^+(v)$ grafu otrzymują wartości $d(t)$.

	v_1	v_2	v_8	v_3	v_7	v_4	v_9	v_6	v_5
	0	∞	∞	∞	∞	∞	∞	∞	∞
v_1	0	3	7	∞	∞	∞	∞	∞	∞
v_2	0	3	4	12	8	∞	∞	∞	∞
v_8	0	3	4	12	6	∞	∞	∞	∞
v_3	0	3	4	12	6	20	16	∞	∞
v_7	0	3	4	12	6	20	8	11	∞
v_9	0	3	4	12	6	9	8	11	∞
v_4	0	3	4	12	6	9	8	10	17
v_6	0	3	4	12	6	7	6	10	**12**
		v_1	v_3	v_2	v_8	v_9	v_7	v_4	v_6

Wartości $p(v)$ dla $v \in V \setminus v_1$ są podane w ostatnim wierszu.



Rysunek 8.5. Graf do przykładu 8.2.2

11. Systemy i algorytmy zachłanne

11.1. Systemy zachłanne

Około 1980 roku J. Korte[35] i L. Lovász[42] wprowadzili greedoidy, będące dalszym uogólnieniem matroidów, jako narzędzie do opisywania i badania algorytmów zachłannych*.

Niech E będzie zbiorem skończonym. Systemem zachłannym (greedoidem) nazywamy parę $M = (E, \mathcal{F})$ taką, że niepusta rodzina $\mathcal{F}$ podzbiorów zbioru E spełnia następujące postulaty:

(g_1) $\emptyset \in \mathcal{F}$,
(g_2) jeśli $F_1, F_2 \in \mathcal{F}$, $|F_1| < |F_2|$, to istnieje $e \in F_2 \setminus F_1$ taki, że $F_1 \cup e \in \mathcal{F}$.

Zbiory wykonalne i osiągalne

Zbiory należące do $\mathcal{F}$ nazywa się *wykonalnymi* (ang. *feasible*). Rodzinę $\mathcal{F}$ nazywa się *osiągalną* (ang. *accesible*), gdy spełniony jest warunek (g_1) oraz

(a_1) jeśli $F \in \mathcal{F} \setminus \emptyset$, to istnieje $e \in F$ takie, że $F \setminus e \in \mathcal{F}$.

Parę $(E, \mathcal{F})$, gdzie $\mathcal{F}$ jest rodziną osiągalną, nazywamy systemem osiągalnym. Greedoidy są systemami osiągalnymi. Łatwo też zauważyć, że każdy matroid jest systemem zachłannym, gdzie zbiorami wykonalnymi są zbiory niezależne matroidu.

Baza

Bazą systemu zachłannego nazywa się każdy maksymalny zbiór wykonalny. Z postulatu (g_2) wynika, że każda baza ma tę samą liczbę elementów.

Rząd

Rzędem $\rho(F)$ zbioru $F \subseteq E$ nazywa się liczbę elementów maksymalnego zbioru wykonalnego $F \subseteq A$. Zbiór $F \in \mathcal{F}$ jest wykonalny

*B. Korte, L Lovász, R. Schrader, *Greedoids*, Springer-Verlag, Berlin 1991. Patrz również B. Korte, J. Vygen, *Combinatorial Optimization, Theory and Algorithms*, Springer-Verlag, Berlin 2012.

wtedy i tylko wtedy, gdy $\rho(F) = |F|$. Każdy maksymalny wykonalny podzbiór F zbioru A ma tę samą liczbę elementów $\rho(A)$.

Przykład 11.1.1

Niech $G = (V, E)$ będzie grafem nieskierowanym z wyróżnionym wierzchołkiem $r \in V$. Zbiory F krawędzi drzew (niekoniecznie spinających) zawierających wierzchołek r tworzą rodzinę $\mathcal{F}$ taką, że $(E, \mathcal{F})$ jest greedoidem.

Przykład 11.1.2

Niech $T = (V, E, r)$ będzie drzewem o ustalonym korzeniu r. Określmy rodzinę $\mathcal{F}$ jako rodzinę zbiorów krawędzi poddrzew drzewa T o tym samym korzeniu r. Jeśli $F \in \mathcal{F}$ oraz $F \neq \emptyset$, to usuwając krawędź e incydentną z wierzchołkiem stopnia 1, otrzymujemy znowu poddrzewo o korzeniu r lub zbiór pusty. Jedyną bazą takiego greedoidu jest E (patrz zadanie 11.1).

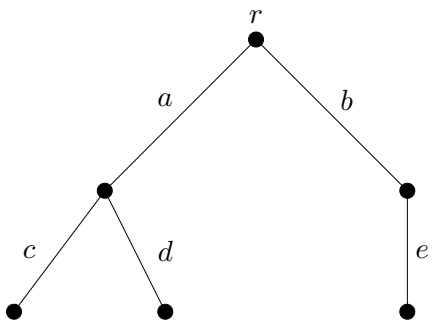
Niech T będzie drzewem przedstawionym na rysunku 11.1. Wówczas $A \in \mathcal{F}$ tylko wtedy, gdy zajdzie jeden z przypadków:

a) $\{a, b\} \subseteq A$,

b) $a \in A,\ A \subseteq E \setminus \{b, e\}$,

c) $b \in A,\ A \subseteq E \setminus \{a, c, d\}$,

d) $a = \emptyset$.



Rysunek 11.1. Drzewo T z korzeniem r

Wtedy też:

$$\rho(A) = \begin{cases} |A|, & \text{jeśli } A \in \mathcal{F}, \\ 0, & \text{jeśli } A \notin \mathcal{F}. \end{cases}$$

W matroidzie każdy podzbiór bazy jest zbiorem niezależnym. Jeśli więc jedyną bazą jest cały zbiór E, to każdy zbiór $A \subseteq E$ jest niezależny, czyli jest to matroid wolny. Tutaj nie każdy podzbiór E należy do $\mathcal{F}$. Oznacza to (patrz zadanie 11.1), że tak skonstruowana para $(E, \mathcal{F})$ jest greedoidem, ale nie jest matroidem.

Wróćmy do warunków (s_1) – (s_3) z twierdzenia 10.1.5 na str. 184 i zastąpmy warunek (s_4) warunkiem (s'_4):

(s_1) $A \subseteq \sigma(A)$,
(s_2) jeśli $A \subseteq B$, to $\sigma(A) \subseteq \sigma(B)$,
(s_3) $\sigma(\sigma(A)) = \sigma(A)$,
(s'_4) jeśli $f \notin \sigma(A)$, $f \in \sigma(A \cup e)$, $f \neq e$ to $e \notin \sigma(A \cup f)$.

Antymatroid Parę (E, σ) spełniającą warunki (s_1) – (s'_4) nazywamy antymatroidem. Antymatroid jest greedoidem, choć nie jest matroidem.

Przykład 11.1.3

Niech $\sigma(A)$ będzie minimalnym poddrzewem drzewa z przykładu 11.1.2, zawierającym krawędzie zbioru A. Wtedy $\sigma(A)$ spełnia warunki (s_1) – (s'_4), czyli greedoid z tego przykładu jest antymatroidem.

Wprowadźmy trzy własności greedoidów o rodzinie wykonalnej $\mathcal{F}$.

(v_0) Własność przedziałowa: jeśli $F, G, H \in \mathcal{F}$, $F \subseteq G \subseteq H$ oraz dla każdego $e \notin H$ zachodzi $F \cup e \in \mathcal{F}$, $H \cup e \in \mathcal{F}$, to wtedy $G \cup e \in \mathcal{F}$.

(v_G) Własność przedziałowa bez ograniczenia górnego: jeśli $F, G \in \mathcal{F}$, $F \subseteq G$ oraz dla każdego $e \notin G$ zachodzi $F \cup e \in \mathcal{F}$, to wtedy $G \cup e \in \mathcal{F}$.

(v_D) Własność przedziałowa bez ograniczenia dolnego: jeśli $G, H \in \mathcal{G}$, $F \subseteq H$ oraz $H \cup e \in \mathcal{F}$, to wtedy $G \cup e \in \mathcal{F}$.

Uzasadnieniem wprowadzenia tych pojęć są następujące trzy własności.

Własność 11.1.1

Greedoid mający własność przedziałową jest greedoidem, dla którego suma dwóch dowolnych zbiorów wykonalnych jest wykonalna, o ile zawiera inny zbiór wykonalny.

Własność 11.1.2

Greedoid mający własność przedziałową bez ograniczenia górnego jest antymatroidem.

Własność 11.1.3

Greedoid mający własność przedziałową bez ograniczenia dolnego jest matroidem.

Przykład 11.1.4

Niech $G = (V, E)$ będzie grafem prostym z wyróżnionym wierzchołkiem $r \in V$. Niech $\mathcal{F}$ będzie rodziną zbiorów wierzchołków F takich, że $r \in F$ oraz F tworzy spójny podgraf grafu G. Wtedy $(V, \mathcal{F})$ jest antymatroidem. Ten greedoid jest zwany greedoidem przeszukiwań wierzchołków (ang. *vertex search greedoid*).

Zauważmy, że greedoid z przykładu 11.1.3 można uważać za szczególny przypadek tak określonego greedoidu. Wystarczy do drzewa T z przykładu 11.1.3 dołączyć wierzchołek r_0 i dodać krawędź $\{r_0, r\}$, otrzymując drzewo T_0, a następnie utworzyć graf krawędziowy $G = L(T_0)$ z wierzchołkiem $r = L(\{r_0, r\})$.

Przykład 11.1.5

Niech $G = (V, E)$ będzie digrafem prostym z wyróżnionym wierzchołkiem $r \in V$. Niech $\mathcal{F}$ będzie rodziną łuków poddrzew skierowanych zawierających łuki skierowane zgodnie z drogami prowadzącymi od r do wierzchołków końcowych. $(E, \mathcal{F})$ jest greedoidem przedziałowym, ale nie jest ani matroidem, ani antymatroidem. Ten greedoid jest zwany greedoidem przeszukiwań łuków (ang. *line search greedoid*).

Zauważmy, że również tutaj greedoid z przykładu 11.1.3 można uważać za szczególny przypadek tak określonego greedoidu. Wystarczy, że w drzewie T z przykładu 11.1.3 zorientujemy krawędzie tak, aby łuki były skierowane zgodnie z drogami prowadzącymi od r do wierzchołków końcowych.

Greedoid można też określić za pomocą poniżej zdefiniowanego języka zachłannego.

Niech E^* będzie zbiorem słów nad alfabetem E, to znaczy zbiorem ciągów o elementach (literach) z E. Konkatenacją $\alpha\beta$ słów $\alpha = x_1 \dots x_i$ oraz $\beta = y_1 \dots y_j$ jest słowo $x_1 \dots x_i y_1 \dots y_j$. Nośnikiem $\tilde{\alpha}$ słowa α jest zbiór różnych liter słowa α. Słowo jest *proste*, gdy każda litera występuje w słowie co najwyżej jeden raz.

Językiem nazywamy skończoną i niepustą rodzinę $\mathcal{L} \subseteq E^*$. Język jest prosty, gdy wszystkie jego słowa są proste. Nośnikiem $\tilde{\mathcal{L}}$ języka $\mathcal{L}$ jest:

$$\tilde{\mathcal{L}} = \{\tilde{\alpha} : \alpha \in \mathcal{L}\}. \tag{11.1.1}$$

Język zachłanny

Językiem zachłannym nad skończonym zbiorem E jest para $(E, \mathcal{L})$, gdzie $\mathcal{L}$ jest językiem prostym takim, że:

(l_1) jeśli $\alpha = \beta\gamma$ i $\alpha \in \mathcal{L}$, to $\beta \in \mathcal{L}$,
(l_2) jeśli $\alpha, \beta \in \mathcal{L}$ oraz $|\alpha| > |\beta|$, to α zawiera literę x taką, że $\beta x \in \mathcal{L}$.

Warunek (l_1) jest warunkiem (lewej) dziedziczności. Warunek (l_2) jest aksjomatem wymiany.

Twierdzenie 11.1.1

Systemy zachłanne i języki zachłanne są równoważne w następującym sensie.

(i) *Jeśli* $(E, \mathcal{L})$ *jest językiem zachłannym, to* $\left(E, \tilde{\mathcal{L}}\right)$ *jest systemem zachłannym.*

(ii) *Jeśli* $(E, \mathcal{F})$ *jest systemem zachłannym, to*

$$\mathcal{L}(\mathcal{F}) = \{x_1 \ldots x_k \in E^* : \{x_1, \ldots, x_i\} \in \mathcal{F} \text{ dla } 1 \leqslant i \leqslant k\}$$

jest językiem zachłannym.

(iii) $\mathcal{L}\left(\tilde{\mathcal{L}}\right) = \mathcal{L}$ *oraz* $\tilde{\mathcal{L}}(\mathcal{F}) = \mathcal{F}$.

Przykład 11.1.6

Niech zbiorem liter będą krawędzie drzewa z przykładu 11.1.2. Określmy słowa języka $\mathcal{L}$ jako ciągi symboli $\alpha = (x_1, \ldots, x_k)$ takie, że zbiór $\{x_1, \ldots, x_k\}$ jest drzewem o korzeniu r oraz jeśli $\alpha = \beta\gamma, \gamma = (y_1, \ldots, y_l)$, to również $\{y_1, \ldots, y_l\}$ jest drzewem o korzeniu r. Spełnione są więc warunki (l_1) oraz (l_2).

11.2. Algorytmy zachłanne

Niech E będzie zbiorem skończonym oraz $w : E \to \mathbb{R}^+$. Wartość funkcji $w(e)$ nazywa się wagą elementu e. Niech $\mathcal{S}$ będzie pewną rodziną podzbiorów zbioru E. Liczbę:

$$w(A) = \sum_{e \in A} w(e)$$

Algorytm zachłanny

nazywa się wagą zbioru A. Rozważmy problem znalezienia zbioru $A \in \mathcal{S}$ o największej wadze. W tym celu sformułujemy następujący algorytm, zwany zachłannym.

Algorytm 11.2.1

1. *Sortujemy elementy* $e \in E$ *według nierosnących wag w ciąg* $e_1 \geqslant e_2 \geqslant \cdots \geqslant e_n$ *(lub według niemalejących wag* $e_1 \leqslant e_2 \leqslant \cdots \leqslant e_n$*).*
2. *Zaczynając od* $A = \emptyset$*, dla* $i = 1, \ldots, n$*, jeżeli* $A \cup e_i \in \mathcal{S}$*, to* $A \leftarrow A \cup e_i$.

Otrzymany w ten sposób zbiór A jest wynikiem działania algorytmu zachłannego.

Algorytm zachłanny nie musi być optymalny, to znaczy zbiór $A \in \mathcal{S}$ znaleziony za pomocą algorytmu zachłannego nie musi mieć największej (najmniejszej) wagi.

Przykład 11.2.1

Niech:

$$A = \begin{bmatrix} 7 & 5 & 1 \\ 3 & 4 & 3 \\ 2 & 3 & 1 \end{bmatrix}.$$

Suma trzech elementów macierzy A wybranych algorytmem 11.2.1 takich, że z każdej kolumny może być tylko jeden element, jest największa i jest równa 15. Jeśli natomiast wybrane elementy nie mogą być z tych samych kolumn i wierszy, to suma takich trzech elementów nie jest największa. Algorytm zachłanny daje sumę 12, natomiast największa suma jest równa 13.

Bardziej ogólne przykłady pozostawimy jako zadania 11.2 i 11.3.

R. Rado[59] i J. Edmonds[14] udowodnili poniższy wynik.

Twierdzenie 11.2.1

Jeżeli $M = (E, \mathcal{I})$ jest matroidem, to zbiór A wyznaczony przez algorytm zachłanny jest zbiorem niezależnym o największej (najmniejszej) wadze. Jeżeli $(E, \mathcal{I})$ nie jest matroidem, to istnieje waga $w : E \to \mathbb{R}^+$ taka, że A nie jest zbiorem o największej (najmniejszej) wadze.

Za pomocą algorytmu zachłannego i z wykorzystaniem twierdzenia 11.2.1 można wyznaczyć minimalne drzewo spinające grafu spójnego.

Jeżeli matroid jest grafowy, to algorytm Kruskala wyboru maksymalnego (minimalnego) drzewa spinającego w grafie G jest algorytmem zachłannym w matroidzie $M(G)$. Algorytm Prima nie jest zaś algorytmem zachłannym w matroidzie $M(G)$, gdyż w sformułowaniu tego algorytmu występują wierzchołki grafu. Jest on jednak algorytmem zachłannym w pewnym greedoidzie związanym z grafem G. Zagadnienie to omówimy w dalszej części tego punktu.

Przykład 11.2.2

W matroidzie Fano F_3 o elementach $\{1, 2, \ldots, 7\}$ (zobacz rysunek 9.3 na str. 176) określimy wagę elementu i wzorem $w(i) = \left(i^2 \mod 6\right) + 1$. Ponieważ różne elementy mogą mieć tę samą wagę, to elementy matroidu mogą być uporządkowane rosnąco na różne sposoby, na przykład:

1. $(6, 5, 7, 1, 3, 2, 4)$,
2. $(6, 5, 1, 7, 3, 2, 4)$,
3. $(6, 7, 1, 5, 3, 4, 2)$.

Wybieramy bazę o najmniejszej wadze. Przy pierwszym i drugim uporządkowaniu minimalną bazę tworzą trzy pierwsze elementy, czyli $B_1 = \{5, 6, 7\}$ oraz $B_2 = \{1, 5, 6\}$. Przy uporządkowaniu trzecim element 1 będący na trzeciej pozycji należy do $\sigma(6, 7)$, więc musi być pominięty i wzięty element 5. Stąd $B_3 = B_1$.

Algorytm Huffmana

Algorytm Huffmana[31] jest metodą bezstratnej kompresji danych, opracowaną w 1952 roku. Mimo że nie jest zbyt efektywny, stosuje się go ze względu na prostotę oraz brak ograniczeń patentowych. Jest przykładem algorytmu zachłannego.

Z symboli utworzony jest ciąg skończony (tekst) τ. Słowa kodowe o długości k bitów (znaków 0 i 1) każde wystarczają do zakodowania 2^k symboli. Na przykład powszechnie stosowany ośmiobitowy kod ASCII wystarcza na zakodowanie 128 symboli. Nie jest jednak konieczne, aby słowa kodowe były jednakowej długości.

Załóżmy, że każdemu symbolowi przypisujemy $s \in S$ i przyporządkowujemy wagę $w(s)$. Niech $b(s)$ będzie liczbą bitów słowa kodującego symbol s. Kodowanie Huffmana polega na utworzeniu słów kodowych o niejednakowej długości w taki sposób, aby średnia ważona

$$W = \frac{\sum_{i=1}^{n} b(s_i)\, w(s_i)}{\sum_{i=1}^{n} w(s_i)} \tag{11.2.1}$$

była najmniejsza. Jeśli za wagę $w(s_i)$ przyjmiemy prawdopodobieństwo p_i występowania symbolu s_i w ciągu τ, to L określone wzorem (11.2.2) jest wartością oczekiwaną długości słowa kodowego:

$$L = \sum_{i=1}^{n} b(s_i)\, p(s_i), \tag{11.2.2}$$

gdyż mianownik prawej strony wzoru (11.2.1) jest równy 1 (patrz dodatek A.5).

Algorytm Huffmana:

1. Tworzymy listę drzew binarnych, które w wierzchołkach przechowują pary: (s_i, p_i). Na początku drzewa składają się wyłącznie z korzenia.

2. Jeśli na liście jest więcej niż jedno drzewo, powtarzamy: usuwamy z listy dwa drzewa o najmniejszym prawdopodobieństwie zapisanym w korzeniu. Wstawiamy nowe drzewo, w którego korzeniu jest suma prawdopodobieństw usuniętych drzew, natomiast one same stają się jego lewym i prawym poddrzewem. Korzeń drzewa nie przechowuje symbolu.
3. Drzewo, które pozostanie na liście, jest nazywane drzewem Huffmana – prawdopodobieństwo zapisane w korzeniu jest równe 1, natomiast w liściach drzewa zapisane są symbole.

Algorytm Huffmana nie określa, w jakiej kolejności wybierać drzewa z listy, jeśli mają takie samo prawdopodobieństwo. Nie jest również określone, które z usuwanych drzew ma stać się lewym bądź prawym poddrzewem. Jednak bez względu na przyjęte rozwiązanie wartość oczekiwana długości kodu pozostaje taka sama.

Na podstawie drzewa Huffmana tworzone są słowa kodowe według następującego algorytmu:

1. Każdej lewej krawędzi drzewa przypisujemy 0, prawej 1 (można oczywiście odwrotnie).
2. Przechodzimy w głąb drzewa od korzenia do każdego liścia (symbolu): jeśli idziemy w prawo, dopisujemy do kodu bit o wartości 1; jeśli idziemy w lewo, dopisujemy do kodu bit o wartości 0.

Długość słowa kodowego jest równa głębokości symbolu w drzewie.

Przykład 11.2.3

Cztery symbole a, b, c, d występujące z prawdopodobieństwami:

a	b	c	d
0.1	0.2	0.3	0.4

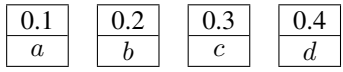
Proces budowy drzewa kodowego przedstawiony jest na rysunku 11.2. Kody znaków:

- $a = 000$,
- $b = 001$,
- $c = 01$,
- $d = 1$.

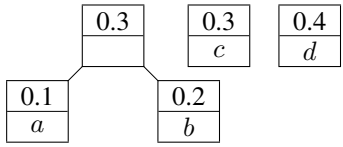
Wartość oczekiwana długości słowa kodowego wynosi:

$$3 \cdot 0.1 + 3 \cdot 0.2 + 2 \cdot 0.3 + 1 \cdot 0.4 = 1.9.$$

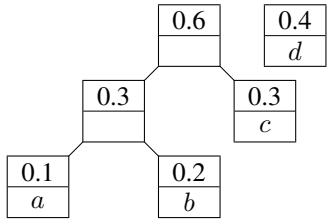
Cztery symbole – cztery drzewa składające się tylko z korzeni:



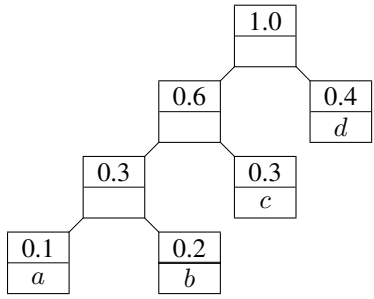
Łączymy korzenie z symbolami a oraz b:



Łączymy korzenie (a, b) oraz c:



Łączymy korzenie $((a, b), c)$ oraz d:



Rysunek 11.2. Budowa drzewa kodowego w algorytmie Huffmana

Więcej o algorytmie Huffmana i algorytmach na nim opartych można znaleźć w książkach [6] i [8].

Pozostaje pytanie, kiedy algorytm zachłanny jest optymalny. Z twierdzenia 11.2.1 wiadomo, że dla dowolnych funkcji wagi jest tak dla matroidów. Rząd matroidu jest funkcją submodularną. Mocniejszą własnością jest modularność funkcji. Funkcja $c(A)$, $A \subseteq E$, jest modularna, gdy spełnia równość:

$$c(A \cup B) = c(A) + c(B) - c(A \cap B).$$

Mocna własność wymiany

Mocną własność wymiany sformułujemy następująco. *Dla każdego $A \in \mathcal{F}$, maksymalnego $B \in \mathcal{F}$, $A \subseteq B$, $x \in E \setminus B$, $A \cup x \in \mathcal{F}$ istnieje $x \in B \setminus A$ taki, że $A \cup y \in \mathcal{F}$ oraz $(B \setminus y) \cup x \in \mathcal{F}$.*

Własność 11.2.1

Greedoid z przykładu 11.1.1 ma mocną własność wymiany.

O optymalności algorytmu zachłannego mówi poniższe twierdzenie.

Twierdzenie 11.2.2

Niech $(E, \mathcal{F})$ będzie greedoidem. Algorytm zachłanny znajduje zbiór $F \subseteq \mathcal{F}$ o maksymalnej wadze dla dowolnej modularnej funkcji $w : \mathcal{F} \to \mathbb{R}^+$ wtedy i tylko wtedy, gdy greedoid ma mocną własność wymiany.

Waga zbioru krawędzi grafu nieskierowanego określona jako suma wag jego krawędzi jest funkcją modularną. Z twierdzenia 11.2.2 i własności 11.2.1 wynika więc optymalność algorytmu Prima.

11.3. Zadania

11.1. Pokazać, że para $(E, \mathcal{F})$ skonstruowana w przykładzie 11.1.2 jest greedoidem.

11.2. Niech A będzie macierzą o n wierszach i m kolumnach o nieujemnych elementach. Uzasadnić, że suma m elementów macierzy wybranych za pomocą algorytmu 11.2.1 takich, że z i-tej kolumny, $1 \leqslant i \leqslant m$, może być tylko jeden element, jest największa.

11.3. Niech A będzie macierzą kwadratową $n \times n$ o nieujemnych elementach. Wybrane elementy muszą się znajdować w różnych wierszach i różnych kolumnach. Uzasadnić, że algorytm zachłanny nie musi być optymalny.

11.4. Uzasadnić stwierdzenie z przykładu 11.1.3.

11.5. Zbudować drzewo kodowe Huffmana, gdy symbole a, b, c, d występują z prawdopodobieństwami:

a	b	c	d
0.1	0.1	0.1	0.7

,

i obliczyć wartość oczekiwaną L słowa kodowego.

11.6. Zbudować drzewo kodowe Huffmana dla pięciu symboli i prawdopodobieństw:

a	b	c	d	e
0.1	0.1	0.2	0.2	0.4

.

11.7. Zbudować drzewo kodowe Huffmana, gdy wszystkie z pięciu symboli mają to samo prawdopodobieństwo.

Skorowidz

Na zadane w ankiecie pytanie o definicję matematyki dyskretnej pewien student Politechniki Gdańskiej odpowiedział, że jest to dział matematyki, który „dyskretnie wciska się, gdzie się da". Choć prawdopodobnie nie o taką odpowiedź chodziło pytającemu, z pewnością jest w niej trochę prawdy. Z matematyką dyskretną mamy obecnie do czynienia dosłownie wszędzie, ponieważ wszędzie obecna jest informatyka, która wykorzystuje wiele pojęć i konstrukcji powstałych właśnie dzięki temu stosunkowo mało znanemu działowi królowej nauk.

Matematyka dyskretna to zbiorcza nazwa różnych działów matematyki zajmujących się badaniem struktur nieciągłych, a więc takich, które w naturalny sposób znajdują zastosowanie w informatyce. Kryptografia, teoria gier i teoria grafów — to tylko niektóre z działów matematyki dyskretnej praktycznie wykorzystywane przez wielu programistów w codziennej pracy. A jeśli doda się do nich takie zagadnienia jak rekurencja czy algorytmy zachłanne, potrzeba zrozumienia podstaw tego działu matematyki staje się chyba jasna dla wszystkich adeptów informatyki.

Ten podręcznik powstał na bazie doświadczeń autorów w prowadzeniu zajęć z matematyki dyskretnej, teorii grafów i algorytmów kombinatorycznych na Politechnice Wrocławskiej, na Wydziale Podstawowych Problemów Techniki, na Uniwersytecie Ekonomicznym w Poznaniu, na Wydziale Informatyki i Gospodarki Elektronicznej, oraz w Państwowej Wyższej Szkole Zawodowej im. Witelona w Legnicy, na Wydziale Nauk Technicznych i Ekonomicznych. Zajęcia były prowadzone dla studentów informatyki, a także dla tych z kierunku informatyka i ekonometria. I to przede wszystkim dla studentów kierunków informatycznych przeznaczona jest ta książka. Zawiera ona również wiadomości bardziej zaawansowane, przydatne dla doktorantów i zaawansowanych programistów, którym daje teoretyczne podstawy do studiowania algorytmów.

- **Indukcja i rekurencja**
- **Rozmieszczenia i permutacje**
- **Kombinacje i podziały**
- **Grafy i drzewa**
- **Algorytmy grafowe**
- **Struktury kombinatoryczne**
- **Systemy i algorytmy zachłanne**

Matematyka dyskretna bez tajemnic

ISBN 978-83-283-4011-4

Cena: 49,00 zł