Matematyka dyskretna

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

sem. zimowy 2025/26

inf.ug.edu.pl/~amb/MaD

Podsumowanie wykładów semestru zimowego

Teoria zbiorów

- zbiory, operacje na zbiorach: suma, przekrój, różnica, różnica symetryczna
- analogia z operacjami logicznymi
  - $x \in A \cap B \Leftrightarrow (x \in A \wedge x \in B)$
  - $x \in A \oplus B \Leftrightarrow (a \in A \wedge x \notin B \vee x \notin A \wedge x \in B)$
    inne oznaczenia na różnicę symetryczną $A \div B$, $A \triangle B$
  - $A \subseteq B \Leftrightarrow (x \in A \Rightarrow x \in B)$
- zbiór „uniwersalny" U, rozpatrujemy tylko podzbiory ustalonego zbioru: $\mathcal{P}(U)$
  - dopełnienie zbioru $\neg X = U \backslash X$ dla $X \in \mathcal{P}(U)$
    nie ma sensu bez kontekstu zbioru uniwersalnego
  - inne oznaczenie: $X^c, \bar{X}, \complement X$ – wszystkie zakładają niejawnie, że znany jest zbiór uniwersalny
- prawa de Morgana (dla zbiorów): $(A \cap B)^c = (A^c \cup B^c)$

Funkcje charakterystyczne i operacje bitowe

- dla $A \in \mathcal{P}(U)$, $\chi_A : U \to 2$ jest zdefiniowana $\chi_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$
- dla $f : U \to 2$, definiujemy $\operatorname{supp} f = \{x \in U \mid f(x) = 1\}$
- operacje na zbiorach przenoszą się na operacje bitowe na funkcjach (ciągach bitów)
  - $\chi_{A \cup B} = \chi_A \vee \chi_B$
  - $\chi_{\neg A} = \neg \chi_A$
  - $\chi_U = 1$ czyli funkcja $U \to 2$ zawsze równa 1
  - $A \subseteq B \Leftrightarrow \chi_A \leqslant \chi_B$
- zbiór potęgowy $\mathcal{P}(U)$ czasem oznaczany 2^U (jako zbiór funkcji charakterystycznych)

Iloczyn kartezjański (grafy, relacje)

- $A \times B = \{(a, b) \mid a \in A, b \in B\}$
- elementy w parze mają ustaloną kolejność
  (a, b) i (b, a) są różne, chyba że $a = b$
- iloczyn kartezjański to nie samo co zbiór dwuelementowych podzbiorów $\mathcal{P}_2(A \cup B)$
- graf skierowany: zbiór wierzchołków V i krawędzi (łuków)
  $A \subseteq \{(v, w) \in V \times V \mid v \neq w\}$, łuki mają kierunek
  graf nieskierowany: zbiór wierzchołków i krawędzi $E \subseteq \mathcal{P}_2(V)$, krawędzie nie mają kierunku
  graf dwudzielny: $V = V_1 \cup V_2$, $E \subseteq V_1 \times V_2$, krawędzie tylko pomiędzy V_1 a V_2
- ścieżki i cykle w grafach nieskierowanych i skierowanych
- domknięcie tranzytywne relacji – czy istnieje ścieżka pomiędzy wierzchołkami
- drzewo: graf spójny bez cykli (nieskierownych)

Drzewa

- binarne: grafy skierowane, każdy węzeł ma lewego i/lub prawego potomka
- węzły można numerować słowami $w \in \{0, 1\}^\star$,
  słowa określają jednoznacznie węzły w drzewie pełnym
- wysokość drzewa = długość najdłuższego słowa
- drzewo pełne wysokości n ma 2^n liści i $2^n - 1$ węzłów wewnętrznych, razem $2^{n+1} - 1$ węzłów
- drzewa binarnych wyszukiwań: etykieta węzła o kodzie $w0$ poprzedza etykietę węzła o kodzie w, a ta poprzedza etykietę węzła o kodzie $w1$
- wstawianie i wyszukiwanie wymaga przejrzenia jednej gałęzi (ale usuwanie nie jest łatwe)
- analiza algorytmów: drzewo z możliwymi opcjami w każdym ruchu, np. ważenie: trzy możliwe wyniki porównań

Relacje i funkcje

- relacja: jakiś zbiór par $R \subseteq A \times B$, związek wiele do wiele
  - zwrotność, przechodniość, symetria i antysymetria
  - relacje równoważności i porządku
  - domknięcie relacji do relacji zwrotnej, przechodniej
  - transpozycja $R^T = \{(b, a) \mid (a, b) \in R\}$
  - rzut $\pi_1(R) = \{a \in A \mid \exists\, b \in B.\ a\ R\ b\}$
  - złączenie relacji $R \bowtie S = \{(a, b, c) \mid a\ R\ b, b\ S\ c\}$
- złożenie relacji $S \circ R = \{(a, c) \mid \exists\, b \in B.\ a\ R\ b, b\ S\ c\}$, czyli złączenie $R \bowtie S$ i rzut na dwie współrzędne
- funkcja: relacja o warunku funkcyjności (jednoznaczności)
  jeśli $a\ R\ b$ i $a\ R\ b'$, to $b = b'$ (związek wiele do 1)
- funkcja różnowartościowa: związek 1 do 1
- warunki całkowitości i „na”: dla każdego a istnieje b, takie że $a\ R\ b$ i na odwrót

Arytmetyka

- dany wielomian $f(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_0$
- schemat Hornera: $((a_n \cdot x + a_{n-1}) \cdot x + a_{n-2}) \cdot x \ldots + a_0$
- możemy wyliczać dowolne wartości, $f(0) = a_0$, $f(1) = \Sigma_{j=0}^{n}\, a_j$, $f(10) = \Sigma_{j=0}^{n}\, a_j \cdot 10^j$
- dla liczby naturalnej a jest dokładnie jeden ciąg $\{a_j\}$ t.ż.
  - $a = ((a_n \cdot 10 + a_{n-1}) \cdot 10 + a_{n-2}) \cdot 10 \ldots + a_0$
  - $0 \leqslant a_0, a_1, \ldots, a_n < 10$ i $a_n > 0$

  mianowicie wielokrotnie dzielimy przez 10 z resztą
- jest to prawdą i dla innych liczb całkowitych $|b| \geqslant 2$
  czyli układ liczenia o podstawie b (w zasadzie zawsze używamy $b \geqslant 2$)
- $b^n \leqslant a < b^{n+1}$, liczba cyfr a jest $\lceil \log_b a \rceil$
- najczęstsze układy liczenia: dwójkowy $b = 2$, cyfry $0, 1$,
  ósemkowy $b = 8$, cyfry $0, 1, 2, 3, 4, 5, 6, 7$,
  dziesiętny $b = 10$, cyfry $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$,
  szestnastkowy $b = 16$, cyfry $0, 1, 2, 3, 4, 5, 6, 7, 9, A, B, C, D, E, F$

Układy liczenia

- zamiana podstaw liczenia
  - wielokrotne dzielenie z resztą
  - układ dwójkowy – bity, układ ósemkowy – trójki bitów, układ szesnastkowy – czwórki bitów
    można zamieniać te układy grupując bity
- zamiana układu dla ułamków: mnożenie i część całkowita
- podstawa i/lub cyfry mogą być ujemne
  - nie używamy tych możliwości na co dzień
- działania arytmetyczne bezpośrednio w układach liczenia
  - dodawanie: dwa ciągi cyfr, obliczenia od prawej do lewej, możliwe przeniesienie 1
  - mnożenie: obliczenia od prawej do lewej, konieczność tabliczki mnożenia, dowolne przeniesienia
    są szybsze algorytmy dla b. dużych liczb
  - dzielenie: b. łatwe dla układu dwójkowego
  - liczba cyfr wyniku $\approx$ maksimum, suma, różnica

Reprezentacja liczb w komputerze

- liczby całkowite bez znaku: 32 lub 64 bity w układzie dwójkowym
- liczby całkowite ze znakiem: reprezentacja uzupełnieniowa
  liczby $0 \ldots 2^{31}-1$ są reprezentowane dosłownie, liczby $x = -2^{31} \ldots -1$ są reprezentowane przez $x + 2^{32}$
  czyli 32 bity zaczynające się od 1
  np. $-1 = 11111111111111111111111111111111$
- liczby reprezentowana są modulo 2^{32} lub 2^{64}
- liczby mogą być traktowane jako ciągi bitów, można je uważać za funkcje charakterystyczne podzbiorów 32 lub 64, można na nich wykonywać operacje bitowe, np. $-3 \,\&\, -6 = -8$, $-3 \,|\, -6 = -1$, $-3 \,\widehat{}\, -6 = 7$ (xor)
- liczby rzeczywiste $s \cdot m \cdot 2^e$ zapisuje się jako bit znaku, bity mantysy m i bity cechy e

Kombinatoryka, zasada Dirichleta, zasada włączania–wyłączania

- liczenie = bijekcja ze zbiorem $n = \{0, \ldots, n-1\}$
- zasada szufladkowa Dirichleta: jeśli umieszczamy n przedmiotów w $k < n$ szufladach, to w jakiejś muszą być dwa
- np. każdy algorytm kompresji musi wydłużać niektóre pliki
- zasada włączania–wyłączania: liczymy $A \cup B \cup C \cup \ldots$ naprzemiennie dodając i odejmując podwójne/potrójne itd. przekroje
- $|A \times B| = |A| \cdot |B|$, bijekcja jest stosowana do przeliczania indeksów w tablicach wielowymiarowych, $f(i,j) = i \cdot m + j$ gdzie $m = |B|$, moc zbioru A nie musi być znana

Ciągi, permutacje, podzbiory

- ciągi i funkcje: $|A \to B| = |B|^{|A|}$
  $|A|$-krotnie wybieramy jeden z elementów z B
  ciągi długości n to funkcje $n \to B$ (powtórzenia dopuszczalne)
- B^k – ciągi długości k, 2^A – funkcje charakterystyczne
- ciągi bez powtórzeń – elementy *nie mogą* się powtarzać, wybieramy je kolejno na n, $n-1$, $n-2$, ...sposobów
  razem $n \cdot (n-1) \cdot \ldots \cdot (n-k+1) = \frac{n!}{(n-k)!}$, o ile $n \geqslant k$
- permutacje – wszystkie elementy występują w ciągu bez powtórzeń, $n = k$, liczba permutacji jest $n!$
- permutacje to bijekcje $\pi : A \to A$, można je składać, czy obliczać odwrotne
- podzbiory k-elementowe – nie zwracamy uwagi na kolejność elementów w ciągu definiującym zbiór
- symbol Newtona: $\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}$

Kombinatoryka, inne konstrukcje

- permutacje z powtórzeniami: przestawiamy litery w słowie MATEMATYKA, różnych słów jest $\frac{10!}{2! \cdot 3! \cdot 2!}$
- podzbiór = funkcja charakterystyczna, czyli przestawienia słowa 0000...11111, jest ich $\frac{n!}{k! \cdot (n-k)!}$
- rozmieszczenia: n identycznych przedmiotów w k (rozróżnialnych) pudełkach, $\binom{n+k-1}{k-1}$
- kombinacje z powtórzeniami: k-krotny wybór ze zbioru n elementów ze zwracaniem, $\binom{n+k-1}{k}$
- podział nieuporządkowany, np. podział $3 \cdot n$ osób na drużyny trzyosobowe: $\frac{(3n)!}{(3!)^n} \cdot \frac{1}{n!}$: można przestawiać osoby w każdej drużynie, można też przestawiać całe drużyny

Generowanie obiektów kombinatorycznych

$\mathcal{P}(n)$: traktujemy jako zapis bitowy liczb od 0 do $2^n - 1$
$\varnothing, 0, 1, 10, 2, 20, 21, 210, 3, 30, 31, 310, 32, 320, 321, 3210$

$\mathcal{P}_k(n)$: najpierw wybór największego elementu, potem generowanie podzbiorów złożonych z mniejszych liczb np. ($n = 6$, $k = 4$):
3210, 4210, 4310, 4320, 4321, 5210, 5310, 5320, 5321,
5410, 5420, 5421, 5430, 5431, 5432.

łatwo dopuścić powtórzenia (przykład $n = 4$, $k = 3$)
000, 100, 110, 111, 200, 210, 211, 220, 221, 222,
300, 310, 311, 320, 321, 322, 330, 331, 332, 333.

permutacje: zapisujemy jako liczby, zaczynając od najmniejszej
0123, 0132, 0213, 0231, 0312, 0321,
1023, 1032, 1203, 1230, 1302, 1320,
2013, 2031, 2103, 2130, 2301, 2310,
3012, 3021, 3102, 3120, 3201, 3210.

Algebry Boole'a

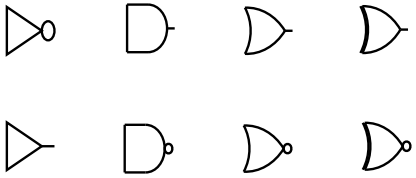
- algebra Boole'a: główny przykład, zbiór $\{0, 1\}$ i operacje logiczne: $\neg, \wedge, \vee, \Rightarrow, \oplus$
  również algebra zbiorów: $\neg, \cap, \cup, \subseteq, \div$
  również ciągi bitów i działania po współrzędnych
- zbiory utożsamiamy z funkcjami charakterystycznymi, dowolne algebry Boole'a są izomorficzne z jakąś algebrą zbiorów
- prawa algebry Boole'a: łączność, przemienność, rozdzielność, prawa de Morgana, pochłanianie i idempotentność
- wyrażenia boolowskie: budowane są ze spójników boolowskich, można interpretować w dowolnej algebrze Boole'a

Funkcje boolowskie

- funkcja boolowska, tabelka wartości na B^n, $B = \{0, 1\}$
- tabelki dla podstawowych operacji i dla dodatkowych (nand, nor)
- równość dwóch wyrażeń – sprawdzenie równości w tabelce funkcji tych wyrażeń
- postaci normalne funkcji: dyzjunkcyjna i koniunkcyjna
  wypisanie wierszy z wartością 1 (DNF) lub 0 (CNF)
- funkcje boolowskie w językach programowania
  - logiczne: napis niepusty i liczba niezerowa są traktowane jako prawda, wynikiem jest 0 lub 1
  - bitowe: liczby traktowane są jako ciągi bitów, wynikiem działań są też ciągi bitów

Funkcje boolowskie c.d.

- zapis graficzny wyrażeń boolowskich (standardowe oznaczenia negacji, koniunkcji, dyzjunkcji, xor i ich negacji)



- przykład półsumatora i sumatora
- przykłady zastosowań funkcji boolowskich: operacje na wektorach, szyfr jednorazowy, podział sekretu, odciski zabezpieczające

Arytmetyka modularna, pierścienie $\mathbb{Z}_m$

- def.: $a = b \pmod m \Leftrightarrow m|(a-b)$ czyli $a = b + m \cdot q$ dla pewnego q nazwa: a przystaje do b modulo m
- $[x]_m = \{y \in \mathbb{Z} \mid x = y \pmod m\} = \{x + q \cdot m \mid q \in \mathbb{Z}\}$
- fakt: $x = y \pmod m \Leftrightarrow y \in [x]_m \Leftrightarrow [x]_m = [y]_m$
- def.: pierścień $\mathbb{Z}_m$
  - elementy: klasy abstrakcji relacji kongruencji modulo m
  - zero: $[0]_m = \{q \cdot m \mid q \in \mathbb{Z}\}$
  - jedynka: $[1]_m = \{1 + q \cdot m \mid q \in \mathbb{Z}\}$
  - dodawanie, odejmowanie, mnożenie: poprzez reprezentantów, np. $[x]_m \cdot [y]_m = [x \cdot y]_m$
- tw.: $\mathbb{Z}_m$ ma zero, jedynkę, istnieją elementy przeciwne, działania są przemienne, łączne, mnożenie jest rozdzielne względem dodawania

Algorytm Euklidesa

- $NWD(a,b) = c$ jeśli $c|a$, $c|b$ oraz c jest największe j.w.
  – wspólnym dzielnikiem jest zawsze 1
  – jeśli $a \neq 0$, $c|a$ to $c \leqslant |a|$
- def: a i b są względnie pierwsze, jeśli $NWD(a,b) = 1$
- Niech $a = \beta \cdot b + r$, wówczas dzielniki (a,b) oraz (b,r) są te same
  – w szczególności $NWD(a,b) = NWD(b,r)$
  – zaczynając od $a \geqslant b \geqslant 0$ otrzymujemy algorytm Euklidesa, koniec jest dla $NWD(a,0) = a$
- np. w języku python

```
def nwd(a,b):
while b:
a,b=b,a%b
return a
```


Algorytm Euklidesa, rozszerzony

- tw.: jeśli $NWD(a,b) = c$, to $\exists \alpha, \beta. \alpha \cdot a + \beta \cdot b = c$

```
#Tadeusz Andrzej Kadlubowski
a,b=map(int,file("dane.txt").readlines()[:2])
p,q,r,s = 1,0,0,1
while b: a,b,p,q,r,s = b,a%b,r,s,p-a/b*r,q-a/b*s
file("nwd.txt","w").write("%d\n%d\n%d\n"%(a,p,q))
```

- wniosek: współczynniki pozwalają obliczać odwrotności w arytmetyce modularnej;
  jeśli $NWD(a,b) = 1 = \alpha \cdot a + \beta \cdot b$, to $\alpha = a^{-1} \mod b$

Chińskie twierdzenie o resztach (CRT)

- tw.: Jeśli $NWD(k, m) = 1$, to układ równań

$$\begin{cases} x = a \mod k \\ x = b \mod m \end{cases}$$

  ma dokładnie jedno rozwiązanie $\mod k \cdot m$
- wniosek: jeśli dany jest układ równań dla zmiennej x

$$\begin{cases} x = a_1 \mod k_1 \\ \dots \\ x = a_\ell \mod k_\ell \end{cases}$$

  i wszystkie k_i są względnie siebie pierwsze, to istnieje jednoznaczne rozwiązanie $\mod k_1 \cdot \ldots \cdot k_\ell$

Twierdzenia Fermata i Eulera

- tw.: jeśli p jest liczbą pierwszą i nie dzieli a, to $a^{p-1} = 1 \pmod p$
- prawie zawsze $2^{n-1} = 1 \pmod n$ implikuje, że n jest liczbą pierwszą
  – 561 jest wyjątkiem, dowolne a spełnia $a^{n-1} = 1 \pmod n$
  – test pierwszości: szybkie potęgowanie w celu sprawdzenia czy $a^{n-1} = 1 \pmod n$, dla różnych wartości a zaczynając od $a = 2$
  – jeśli nie ma równości, to liczba nie jest pierwsza
- funkcja Eulera: $\varphi(n) =$ liczba liczb $a \in \{1, .., n-1\}$ takich że $NWD(a, n) = 1$
- tw.: Jeśli $NWD(a, n) = 1$, to $a^{\varphi(n)} = 1 \pmod n$
- tw. Fermata jest przypadkiem szczególnym
- na tym twierdzeniu oparty jest system kryptograficzny RSA, podnoszenie do jednej potęgi jest szyfrowaniem, podnoszenie do kolejnej wracamy do punktu wyjścia – osoby nieuprawnione nie znają tych potęg

Definicje rekurencyjne

- ciąg Fibonacci'ego: $a_0 = 1$, $a_1 = 1$, $a_{n+2} = a_{n+1} + a_n$
  rozwiązanie: kombinacja $(\frac{1+\sqrt{5}}{2})^n$ i $(\frac{1-\sqrt{5}}{2})^n$
- przykład algorytmu: sortowanie przez scalanie:
  - baza: jeden element – zadanie już jest wykonane
  - rekurencja: dzielimy tablicę na dwie części, sortujemy każdą z osobna, a potem scalamy obie części
- złożoność (i przykład funkcji):
  $T_1 = 0$, $T_n = 2 \cdot T_{n/2} + n - 1$
  rozwiązanie $T_n = O(n \cdot \log n)$
- $newton(n+1, k+1) = newton(n, k+1) + newton(n, k)$
  - dosłowna definicja prowadzi do wykładniczej złożoności
  - jeśli rozwiązania mniejszych zadań są zapisywane, złożoność jest radykalnie mniejsza

Drzewa binarne

- drzewo ma jeden korzeń
  każdy węzeł ma lewego i prawego syna
  każdy z tych potomków może nie istnieć
- wypisywanie uporządkowanych etykiet drzewa T:
  - jeśli T jest puste, to nie ma co wypisywać
  - w p.p. wypisz uporządkowane etykiety lewego syna T
  - następnie etykietę $T(\varepsilon)$
  - i na końcu wypisz uporządkowane etykiety prawego syna T
- notacja polska prosta i odwrotna:
  - najpierw korzeń (operacja) potem synowie (argumenty)
  - najpierw synowie (argumenty funkcji) potem korzeń

Grafy

- def.: graf nieskierowany = zbiór wierzchołków V połączonych krawędziami $E \subseteq \mathcal{P}_2(V)$
- taka definicja wyklucza
  - podwójne krawędzie
  - pętelki wokół wierzchołków
  - odróżnienie krawędzi w jedną i przeciwną stronę
- graf skierowany $G = (V, A)$ gdzie $A \subseteq \{(v, w) \in V \times V \mid v \neq w\}$
  - tym razem krawędzie (łuki) mają kierunek „od v do w": początek i koniec krawędzi
  - mogą być dwie krawędzie w przeciwne strony

Tw Halla (wybór reprezentantów)

- skojarzenie: wybór krawędzi grafu dwudzielnego tak, by każdy wierzchołek miał najwyżej jednego sąsiada
- tw. Halla: istnieje skojarzenie i obejmie wszystkie wierzchołki V_1 wtedy i tylko wtedy, gdy każdy podzbiór $V \subseteq V_1$ ma co najmniej $|V|$ sąsiadów w V_2
- dana rodzina zbiorów $M_k \subseteq M, k = 1, \ldots, \ell$
  zadanie: wybrać różnych reprezentantów $m_k \in M_k$
  warunkiem koniecznym i wystarczającym jest by dla każdego $J \subseteq \{1, \ldots, \ell\}$ zachodził warunek $|J| \leqslant |\bigcup_{j \in J} M_j|$
- definiujemy graf: $V_1 = \{1, \ldots, \ell\}$, $V_2 = \bigcup_{k=1}^{\ell} M_k$,
  krawędź łączy j z m gdy $m \in M_j$,
  zbiór M_j staje się zbiorem sąsiadów j
  a zbiorem sąsiadów m jest $\{j \mid m \in M_j\}$

Reprezentacja grafów

graf $G = (V, E)$, $|V| = n$, $E \subseteq \mathcal{P}_2(V)$, $|E| = k$, na ogół $k >> n$

- macierz incydencji:
  macierz dwuwymiarowa $n \times k$: $G(i, j) = 1$ jeśli wierzchołek i jest końcem krawędzi j
- macierz sąsiedztwa: macierz dwuwymiarowa $n \times n$: $G(i, j) = 1$ jeśli wierzchołki i oraz j są połączone krawędzią
- dla grafów skierowanych też, być może $G(i, j) \neq G(j, i)$
- listy incydencji:
  dla każdego wierzchołka v lista $L(v)$ jego sąsiadów
- dla grafów nieskierowanych krawędzie są reprezentowane dwukrotnie, trzeba zachować ostrożność
- dla grafów skierowanych może być wygodne przechowywanie i następników i poprzedników (dwa rodzaje sąsiadów)

Algorytmy grafowe

- przeszukiwanie grafu wgłąb i wszerz
- budowa drzewa/lasu rozpinającego graf
  być może o minimalnej wadze, jeśli krawędzie mają wagi
- poszukiwanie ścieżki/cyklu Eulera/Hamiltona
- Eulera – łatwe, ale małe zastosowania
  Hamiltona – problem ważny i generalnie trudny
- szukanie najkrótszej ścieżki w grafie skierowanym (osobne przypadki dla grafów acyklicznych i dodatnich wag krawędzi)
- ogólna złożoność $O(n^3)$, przypadki szczególne mają złożoność $O(n^2)$