Matematyka dyskretna

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

sem. zimowy 2025/26

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

Wielomiany, notacja asymptotyczna

Sortowanie bąbelkowe

- `pl.wikipedia.org/wiki/Sortowanie_b%C4%85belkowe`
- dany ciąg liczb $a_1, \ldots, a_n$, cel: porządek nierosnący
- wielokrotnie powtarzamy pętlę
  `for` $(i = 1; i < n; i{+}{+})$
  `if` $a_i < a_{i+1}$ `then zamień(`a_i`,` a_{i+1}`)`
- w pętli jest $n - 1$ porównań
  po każdym wykonaniu kolejny najmniejszy element znajdzie swoje miejsce
- wystarczy $n - 1$ pętli, $(n{-}1) \cdot (n{-}1)$ porównań
- można skracać kolejne pętle: $n{-}1 + n{-}2 + \ldots + 1 = \frac{1}{2}n \cdot (n-1)$
- można jeszcze bardziej skracać pętle, notując miejsce ostatniej zamiany
  – w najgorszym przypadku nic nie zyskamy, ale jeśli ciąg był już uporządkowany, to po $n - 1$ porównaniach dowiemy się o tym

Notacja asymptotyczna

- złożoność sortowania bąbelkowego w zależności od długości:
  $n^2 - 2n + 1$
  w lepszej wersji $\frac{n(n-1)}{2}$, a może się zdarzyć, że jeszcze mniej
- dane dwa ciągi liczb nieujemnych $b, c : \mathbb{N} \to \mathbb{R}_+$
- $b_n = O(c_n)$ (b jest „O duże od" c) jeśli
  $\exists\, C.\ \exists\, n_0.\ \forall\, n.\ n > n_0 \Rightarrow b_n < C \cdot c_n$
  „prawie wszystkie b są mniejsze niż c, stała jest nieważna"
- $b_n = o(c_n)$ (b jest „o małe od" c) jeśli
  $\forall\, C.\ \exists\, n_0.\ \forall\, n.\ n > n_0 \Rightarrow b_n < C \cdot c_n$
  „prawie wszystkie b są dowolnie mniejsze niż c"
- $b_n = \Theta(c_n)$ (b jest „Θ od" c) jeśli
  $\exists\, C_1.\ \exists\, C_2.\ \exists\, n_0.\ \forall\, n.\ n > n_0 \Rightarrow C_1 \cdot c_n < b_n < C_2 \cdot c_n$
  „prawie wszystkie b są mniej więcej równe c"

Notacja asymptotyczna c.d.

- złożoność sortowania bąbelkowego jest „O duże od" n^2
- jeśli b jest „o małe od" c, to jest również „O duże"
- z kolei c na pewno nie jest „O duże od" b
- „Θ od" oznacza „O duże" w obie strony
- jeśli istnieje granica ilorazu $\frac{b_n}{c_n}$ skończona, niezerowa, to b jest Θ od c, ale bez tej granicy też może być
- relacja „O duże" jest zwrotna i przechodnia, ale brak słabej antysymetrii, relacja „Theta" jest relacją równoważności
- relacja „o małe" jest porządkiem (przeciwzwrotna i przechodnia)
- $\log n$ jest „o małe od" n, n^k jest „o małe od" n^ℓ, $n < \ell$
- n^ℓ jest „o małe od" 2^n dla dowolnej potęgi ℓ
- $100 \cdot n$ jest „o małe od" $n \cdot \log_2 n$, ale dla $n < 2^{100}$ $100 \cdot n > n \cdot \log_2 n$

Wielomiany

- wielomian – funkcja zmiennej liczbowej $f(x) = a_n \cdot x^n + \ldots + a_1 \cdot x + a_0$
- stopień wielomianu – największe n, że $a_n \neq 0$ $(st(0) = -1)$
- pierwiastek wielomianu (niezerowego) – $f(a) = 0$
- dodawanie – (wielomian niższego stopnia trzeba uzupełnić zerami) $(a_n \cdot x^n + \ldots + a_0) + (b_n \cdot x^n + \ldots + b_0) = (a_n + b_n) \cdot x^n + \ldots + (a_0 + b_0)$
- mnożenie – $(a_n \cdot x^n + \ldots + a_0) \cdot (b_k \cdot x^k + \ldots + b_0) = c_{n+k} \cdot x^{n+k} + \ldots + c_0$ gdzie $c_i = \Sigma_{j=0}^{i} a_j \cdot b_{i-j}$, a dalsze a_j i b_j są zerami dw.: $c_i \cdot x^i = \Sigma_{j=0}^{i} a_j \cdot x^j \cdot b_{i-j} \cdot x^{i-j}$
- stopień sumy niewiększy niż większy ze stopni stopień iloczynu równy sumie stopni (dla niezerowych)

Wielomiany, pierwiastki, tw. Bézouta

- dzielenie: dane $U(x), V(x) = v_k \cdot x^k + \ldots, v_k \neq 0$ szukamy $Q(x), R(x)$ takich, że $U = V \cdot Q + R$ i $st(R) < k$
- najpierw $Q = 0, R = U$ dopóki $st(R) \geqslant k$ dzielimy najwyższy wyraz w R przez $v_k \cdot x^k$, dodajemy do Q i od R odejmujemy odpowiednio
- dzielenie przez $x - a$: $U(x) = (x - a) \cdot Q(x) + R$ gdzie R jest stałą
- wniosek (twierdzenie Bézouta): U jest podzielny przez $x - a$ wtt a jest pierwiastkiem dw.: podstawiając $x = a$ widać $R = 0$
- liczba pierwiastków wielomianu nie przekracza jego stopnia może być mniej, bo mogą być wielokrotne $U(x) = (x - a_1) \cdot (x - a_2) \ldots (x - a_k)$

Schemat Hornera

- naiwne obliczenie wartości wielomianu $f(x) = \Sigma_{j=0}^{n} a_j \cdot x^j$ $a_n x^n$ wymaga n mnożeń potem $n-1$ mnożeń itd. razem $O(n^2)$
- schemat Hornera: $f(x) = ((a_n \cdot x + a_{n-1}) \cdot x + a_{n-2}) \cdot x \ldots + a_0$
- najpierw $wynik = 0$ w pętli $wynik = wynik * x + a_i$, $i = n, n-1, \ldots, 0$

```
wynik=0;
for (j=n;j>-1;j--)
        wynik=wynik*x+a[j];
```

czyli $O(n)$ mnożeń (dodawania zwyczajowo nie rozpatrujemy, w tym przypadku jest ich drugie tyle)

Arytmetyka

Ciąg geometryczny

- niech $b > 1$, wówczas $\Sigma_{j=0}^{n-1}\ b^j = \frac{b^n-1}{b-1}$.
  dw.: Po pomnożeniu przez mianownik mamy
  $\Sigma_{j=1}^{n}\ b^j - \Sigma_{j=0}^{n-1}\ b^j = b^n - 1$
- dla $b = 1$ suma ma sens: $\Sigma_{j=0}^{n-1}\ b^j = n$, choć ułamek jest nieokreślony
- dla $b < 1$, $b \neq 0$ twierdzenie również jest prawdziwe, z tym samym dowodem
- dla $b = 0$ ułamek jest równy 1, suma zawiera wyraz nieokreślony 0^0, ale rozsądnie jest przyjąć $0^0 = 1$, oczywiście $0^i = 0$ dla $i > 0$
  czyli równość również zachodzi

Układy liczenia

- dany jest wielomian $f(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_0$
- możemy wyliczać dowolne wartości, np. $f(0) = a_0$, $f(1) = \Sigma_{j=0}^{n}\ a_j$, $f(10) = \Sigma_{j=0}^{n}\ a_j \cdot 10^j$
- dla dowolnej liczby naturalnej a można znaleźć wielomian f taki, że
  1. $a = f(10)$
  2. $\forall\, j \in \{0, \ldots, n\}\,.\ a_j \in \{0, 1, \ldots, 9\}$

  dw.: stosujemy schemat Hornera,
  $a = ((a_n \cdot 10 + a_{n-1}) \cdot 10 + a_{n-2}) \cdot 10 \ldots + a_0$,
  czyli a_0 jest resztą z dzielenia a przez 10
  a_1 jest resztą z dzielenia $\frac{a-a_0}{10}$ przez 10, itd.
  dzielimy coraz mniejsze liczby, musimy dojść do zera
- oczywiście jest to prawdą i dla innych liczb $\geqslant 2$

Zamiana układu liczenia

- układ liczenia o podstawie $b \geqslant 2$ to przedstawianie liczby a jako wartości wielomianu $a = a_n \cdot b^n + a_{n-1} \cdot b^{n-1} + \ldots + a_0$
  $\forall\, j.\ 0 \leqslant a_j < b$
- współczynniki $a_n, a_{n-1}, \ldots, a_0$ – „cyfry" zapisu liczby a
- algorytm przedstawiania liczby w układzie o podstawie b jest odwrotnością schematu Hornera

```
zapisz=a; j=0;
while (zapisz>0) {
      a[j]=zapisz%b;
      zapisz=(zapisz-a[j])/b;
      j=j+1;
}
```

  uwaga: ostatnia wyprodukowana cyfra $a[j-1] \neq 0$
  dla $a = 0$ nic nie jest wyprodukowane, zamiast ε piszemy 0
- przyjmujemy ogólnie, że pierwsza cyfra jest niezerowa

Przykład zamiany

- zmiana $(148)_{10}$ i $(147)_{10}$ na układ dwójkowy

x	$= 2 \cdot q$	$+r$
148	74	0
74	37	0
37	18	1
18	9	0
9	4	1
4	2	0
2	1	0
1	0	1
0		

x	$= 2 \cdot q$	$+r$
147	73	1
73	36	1
36	18	0
18	9	0
9	4	1
4	2	0
2	1	0
1	0	1
0		

- $(148)_{10} = (10010100)_2$, $(147)_{10} = (10010011)_2$ wyżej są cyfry mniej znaczące (czyli bardziej z prawej)

Układ pozycyjny

- znaczenie cyfry zależy od jej pozycji
- niech $w \in \{0, \ldots, b-1\}^*$ – słowo złożone z cyfr
  $[[w]]$ – znaczenie słowa w jest zdefiniowane

$$[[w]] = \begin{cases} [[v]] \cdot b + c & w = vc \\ 0 & w = \varepsilon \end{cases}$$

  jest to schemat Hornera raz jeszcze
- zbiór cyfr może być dowolny, w zasadzie chcemy jednoznacznego zapisu
- dla $b = 1$ algorytm Hornera nie ma końca, ale $w \in \{1\}^*$ ma znaczenie, $[[w]] = |w|$ – liczba jedynek

Układy dwójkowy, ósemkowy, szesnastkowy

- są to najważniejsze dla nas układy liczenia
- dwójkowy = binarny, szesnastkowy = heksagonalny
- dla układu szesnastkowego potrzebujemy 16 cyfr –
  $0, 1, \ldots 9, A, B, C, D, E, F$
- zapis szesnastkowy – ciąg cyfr poprzedzony $0x$
- zamiana tych układów jest szczególnie prosta
  $\Sigma_{j=0}^{n}\, b_j \cdot 2^j =$
  $\Sigma_{k=0}^{\lfloor n/4 \rfloor}\, (b_{4k+3} \cdot 2^3 + b_{4k+2} \cdot 2^2 + b_{4k+1} \cdot 2 + b_{4k}) \cdot 2^{4k}$
- czyli paczki 4 bitów stanowią jedną cyfrę heksagonalną
  zapis binarny uzupełniamy z przodu zerami do pełnej czwórki
- na odwrót, każda cyfra układu szesnastkowego ma zapis jako 4 bity
  np. $0x0 = 0000$, $0x7 = 0111$, $0xA = 1010$, $0xF = 1111$

Dzielenie z resztą

- Twierdzenie (użyte w algorytmie zamiany układu liczenia)
  dana liczba całkowita $b \neq 0$,
  dany zbiór $\mathcal{R}$ kolejnych $|b|$ liczb całkowitych,
  wówczas dla każdej liczby całkowitej a istnieje iloraz i reszta i są jednoznaczne:
  $$a = q \cdot b + r, \quad r \in \mathcal{R}$$
- dowód: zaczynamy od $q = 0, r = a$
  jeśli $r \in \mathcal{R}$, to koniec dzielenia
  jeśli r jest na prawo od odcinka $\mathcal{R}$, to odejmujemy $|b|$, a jeśli na lewo, to dodajemy $|b|$
  czyli $r = \mp b, q = q \pm 1$
  r nie może przeskoczyć $\mathcal{R}$, więc algorytm kiedyś skończy działanie

Dziwne układy liczenia

- $b = 10$, cyfry $-1, 0, 1, \ldots, 8$
- zmiana -177 na ten układ oraz na układ dziesiętny (cyfry $0 \ldots 9$)

x	$10 \cdot q$	r
-177	-18	3
-18	-2	2
-2	-1	8
-1	0	-1
0		

x	$10 \cdot q$	r
-177	-18	3
-18	-2	2
-2	-1	8
-1	-1	9
-1	$\ldots$	$\ldots$

- $-177 = (-1)823$ w dziwnym układzie liczenia
  $-177 = ((-1 \cdot 10 + 8) \cdot 10 + 2) \cdot 10 + 3 = -1 \cdot 1000 + 8 \cdot 100 + 2 \cdot 10 + 3$
- wykonanie algorytmu dla układu dziesiętnego nie ma końca, produkowane są ciągle 9, zapis byłby nieskończony $\ldots 999823$

Dziwne układy liczenia, ujemne cyfry

- klasyczny układ liczenia i algorytm zamiany stosuje się tylko dla liczb nieujemnych
- liczby ujemne zapisuje się zwyczajowo jako minus liczba dodatnia
- układ z ujemnymi i dodatnimi cyframi jest dobry na wszystko, na liczby dodatnie też, np. zamiana 177 oraz 189 na ten układ

x	$10 \cdot q$	r
177	17	7
17	1	7
1	0	1
0		

x	$10 \cdot q$	r
189	19	-1
19	2	-1
2	0	2
0		

- $177 = (1 \cdot 10 + 7) \cdot 10 + 7 = 1 \cdot 100 + 7 \cdot 10 + 7$
- $189 = 2(-1)(-1)$ w dziwnym układzie liczenia
  $189 = (2 \cdot 10 - 1) \cdot 10 - 1 = 2 \cdot 100 + (-1) \cdot 10 + (-1)$

Dziwne układy liczenia, ujemna podstawa

- $b = -10$, cyfry $0, 1, \ldots, 9$ – ujemna podstawa liczenia
- zmiana 177 oraz -177 na ten układ

x	$-10 \cdot q$	r
177	-17	7
-17	2	3
2	0	2
0		

x	$-10 \cdot q$	r
-177	18	3
18	-1	8
-1	1	9
1	0	1
0		

- $177 = 237$ w dziwnym układzie liczenia
  $177 = (2 \cdot (-10) + 3) \cdot (-10) + 7 = 2 \cdot (-10)^2 + 3 \cdot (-10) + 7$
- $-77 = 1983$ w tym układzie liczenia
  $-177 = ((1 \cdot (-10) + 9) \cdot (-10) + 8) \cdot (-10) + 3 =$
  $1 \cdot (-10)^3 + 9 \cdot (-10)^2 + 8 \cdot (-10) + 3$

Układy liczenia o niejednorodnej podstawie

- różne pozycje mogą mieć różne znaczenia i różne cyfry
- np. *DdHhMmSs*, gdzie $D \in \mathbb{N}$, $0 \leqslant H < 24$, $0 \leqslant M, S < 60$ oznacza czas liczony w dniach, godzinach, minutach i sekundach
- dopuszczalne cyfry na pozycji mogą zależeć od wartości bardziej znaczących cyfr
- np. *YyMmDd* oznacza rok Y, miesiąc M i dzień D, $Y \in \mathbb{N}$, $1 \leqslant M < 13$, ale legalna wartość D zależy od *YyMm*
- standard czasu przewiduje też sekundę przestępną dodawaną w niektórych latach
- w kalendarzu księżycowym w niektórych latach bywa 13 miesięcy
- zamiana z układu „rok/miesiąc/dzień miesiąca" na układ „rok/tydzień/dzień tygodnia" wymaga trochę zachodu

Długość liczby

- $|w| = n$, $w = \Sigma_{j=0}^{n-1} w_j \cdot b^j$, $0 \leqslant w_0, \ldots, w_{n-1} < b$, $0 < w_{n-1}$
- czyli $b^{|w|-1} \leqslant w < b^{|w|}$
- czyli $|w| - 1 \leqslant \log_b w < |w|$
- logarytmy przy różnych podstawach różnią się tylko współczynnikiem

$$\log_b w = \log_c w \cdot \log_b c$$

- a więc $|w| = O(\log w)$, podstawa liczenia wpływa tylko na stałą, ale nie na rząd wielkości
- $\log_{10} 2 \approx 0,301$, $\log_2 10 \approx 3,322$, długość liczby dziesiętnej wynosi ok. 30% długości w układzie binarnym

Działania w układzie liczenia

- dodawanie jedynki w układzie dwójkowym

$$\begin{array}{r} 10101001001010101000011111111111 \\ 1 \\ \hline 10101001001010101001000000000000 \end{array}$$

$$\begin{array}{r} 11111111111 \\ 1 \\ \hline 100000000000 \end{array}$$

- złożoność dodawania jedynki – $O(|w|)$
- porównywanie liczb – porządek kanoniczny, liczby, oprócz 0, przedstawiamy bez wiodących zer, złożoność $O(|w|)$

Dodawanie liczb

- tabliczka dodawania, dodanie dwu cyfr może skutkować liczbą dwucyfrową (przeniesienie)
- dane dwie liczby x i y, krótszą uzupełniamy wiodącymi zerami

```
c=0;
for(j=0;j<|x|;j++){
      z[j]=(x[j]+y[j]+c)%b;
      c=(x[j]+y[j]+c)/b; //dzielenie bez reszty
}
if (c>0) z[j]=c;
```

  przeniesienie może wynieść co najwyżej 1
- złożoność $O(|x|)$

Mnożenie i dzielenie liczb

podstawa 5, cyfry $0, 1, 2, 3, 4$

- szkolny algorytm mnożenia
- każda cyfra jednej liczby jest przemnożona przez każdą cyfrę drugiej liczby, używamy tabliczki mnożenia

	1	2	3	4
1	1	2	3	4
2	2	4	11	13
3	3	11	14	22
4	4	13	22	31

- złożoność kwadratowa – $O(|x| \cdot |y|)$
- istnieją algorytmy zmniejszające tę złożoność, niedawno znaleziono algorytm o złożoności $O(n \log n)$, $n = \max(|x|, |y|)$
- algorytm dzielenia
  - na pozór złożony
  - każda cyfra dzielnika przyrównana do początku dzielnej
  - tyle razy ile cyfr będzie w ilorazie
- złożoność kwadratowa – $O((|x| - |y|) \cdot |y|)$

Duże liczby

- w kryptografii zastanawiamy się nad możliwością przeprowadzenia 2^n obliczeń dla różnych n
- wiek układu słonecznego w latach – 10^{10}
  liczba sekund w roku – $3 \cdot 10^7$
  liczba cykli procesora w sekundzie – $2 \cdot 10^9$
  liczba procesorów na świecie – 10^9 (wzięte z powietrza)
  iloczyn – $6 \cdot 10^{35}$
- liczba ciągów 128-bitowych – $2^{128} \approx 3 \cdot 10^{38}$, czyli 500 razy więcej
- liczbę 160-bitową w chcemy rozłożyć na czynniki dzieląc kolejno przez $\sqrt{w}$ liczb
  czyli 10^{24} dzieleń, każde np. $5 \cdot 80$ cykli procesora
- razem $4 \cdot 10^{26}$ cykli, czyli siedem lat działania wszystkich komputerów na świecie
- tak naprawdę istnieje nieco łatwiejszy rozkład na czynniki

Ułamki

- schemat zamiany układu liczenia działa też na ułamkach
- część ułamkową zapisujemy po kropce (czyli przecinku)
  każda pozycja w prawo jest b razy mniej znacząca
  zapis liczby analizujemy tym razem od lewego końca
- niech $w \in \{0, \ldots, b-1\}^*$ – słowo złożone z cyfr
  $[[0.w]]$ – znaczenie ułamka $0.w$ jest zdefiniowane

$$[[w]] = \begin{cases} ([[v]] + c)/b & w = cv \\ 0 & w = \varepsilon \end{cases}$$

np. $[[0.4290]] = ((((0 + 0)/10 + 9)/10 + 2)/10 + 4)/10$
uwaga: tutaj dzielenie jest dzieleniem liczb rzeczywistych, a nie całkowite z resztą

Przykład zamiany

- $(0.6875)_{10} = (0.1011)_2$
- algorytm zamiany mnoży w pętli przez podstawę układu liczenia i odejmuje części całkowite
-

x	$2 \cdot x$	$\lfloor 2 \cdot x \rfloor$
0.6875	1.375	1
0.375	0.75	0
0.75	1.5	1
0.5	1	1
0		

x	$2 \cdot x$	$\lfloor 2 \cdot x \rfloor$
0.2	0.4	0
0.4	0.8	0
0.8	1.6	1
0.6	1.2	1
0.2	od nowa to samo	

- wyżej są cyfry bardziej znaczące (czyli bliżej przecinka)
- $(0.2)_{10}$ jest ułamkiem okresowym w układzie binarnym
  tylko potęgi 2 mają skończone ułamki binarne
  $(0.2)_{10} = (0.001100110011\ldots)_2$