ЛЕКЦІЯ 13
Нормальні алгоритми Маркова

Теорія нормальних алгоритмів
Вступ

Теорія нормальних алгоритмів
(або алгорифмів, як називав їх творець теорії)
була розроблена математиком

Андрієм Андрійовичем Марковим (1903-1979)
(молодшим)
наприкінці 1940-х — початку 1950-х рр. XX в.

Нормальні алгоритми Маркова є **правилами з перетворення слів у заданому алфавіті.**

Тому вхідні дані і результати роботи цих алгоритмів — це слова в даному алфавіті.

Приклад.

Створити алгоритм $f\left(x\right): x \to y$.

1. x – рослина, y – тварина

x = «банан». Букву «н» замінюємо на «р»- y = «баран»

2. x – рослина, y – птах

x = «банан». Букву «н» замінюємо на «кл»- y = «баклан»

3. x – рослина, y – рослина

x = «банан».Букву «н» замінюємо на «клаж»-

y = «баклажан»

Підстановки Маркова

Алфавітом називається будь-яка **непуста множина**.

1. Елементи алфавіту називають *буквами*, а будь-які послідовності букв — *словами* в даному алфавіті.

Для зручності міркувань допускаються порожні слова (вони не мають у своєму складі жодної букви).

2. *Порожнє слово* будемо позначати Λ.
3. Якщо A й B — два алфавіти, причому $A \subseteq B$, то алфавіт B називається *розширенням* алфавіту A.
4. Слова будемо зазвичай позначати латинськими буквами: P, Q, R (або цими ж буквами з індексами).

$$P_1, P_2, ..., P_n \quad Q_1, Q_2, ..., Q_n \quad R_1, R_2, ..., R_n$$

Одне слово може бути складовою частиною іншого слова.

5. Тоді перше називається *підсловом* другого слова або входженням у друге слово.

Приклад.

Нехай A — український алфавіт.

У цьому алфавіті розглянемо такі слова:

P_1 = параграф, P_2 = граф, P_3 = ра

P_1 = параграф, P_2 = граф, P_3 = ра.

- Слово P_2 є підсловом слова P_1.
- Слово P_3 є підсловом P_1 і P_2
- Слово P_3 *входить двічі в* P_1

Особливий інтерес представляє перше входження

АНАГРАМИ

Це слово означає,
Що війни немає.
Як зміниш буквам місце, -
Повстане древнє місто.

Мене розшукують усі,

Коли дефект є в колесі.

Шоферу кожному в біді

Я можу стати у нагоді.

Заміниш місцями склади –

Буду в лісі я рости.

Лежу я на землі,

Прибита до заліза,

Як букви переставиш–

В каструлю я полізу.

Не лама бааб локотуп,
та пилаку ропося.

делоМоць ротип веоць,
а ротип лодмоця і мас цявів.

щераК сицяни в меніж,
жін равежуль у біне.

Визначення 1. *Підстановкою Маркова в слові* R називається операція, яка задається за допомогою впорядкованої пари підслів (P, Q), що полягає в наступному.

У заданому слові R знаходять перше входження підслова P (якщо таке є) і, не змінюючи інших частин слова R, заміняють у ньому це входження підсловом Q.

Приклад.

R_1=«пароплав» (P,Q)=(«плав», «воз») R_2= «паровоз»

R_1=«укол» (P,Q)=(«кол», «хил») R_2= «ухил»

R_1=«порода» (P,Q)=(«род», «год») R_2= «погода»

R_1=«батрак» (P,Q)=(«трак», «тон») R_2= «батон»

R_1=«арарат» (P,Q)=(«ар», «к») R_2= «карат»

Результат підстановки Маркова

Визначення 2. *Результатом застосування підстановки Маркова* $\left(P, Q\right)$ до слова R *називається слово,* отримане після заміни підслова P на підслово Q в слові R.

Якщо ж входження P в слово R *не існує*, то вважається, що підстановка Маркова $\left(P, Q\right)$ незастосовна до слова R.

Окремими випадками підстановок Маркова є підстановки з порожніми словами: $\left(\Lambda, Q\right)$, $\left(P, \Lambda\right)$, $\left(\Lambda, \Lambda\right)$.

Формула підстановки $\left(P, Q\right)$.

Для позначення підстановки Маркова $\left(P, Q\right)$ використовується запис $P \to Q$.

Його називають *формулою підстановки* $\left(P, Q\right)$.

Деякі підстановки $\left(P, Q\right)$ будемо називати заключними (зміст назви пояснимо дещо пізніше).

Для позначення таких підстановок будемо використовувати запис $P \to\!\bullet\, Q$, називаючи його *формулою заключної підстановки*.

Слово P називають *лівою частиною*, а Q — *правою частиною* у формулі підстановки.

Нормальні алгоритми і їх застосування до слів

Упорядкований скінченний список формул підстановок в алфавіті А називається схемою підстановок:

$$\downarrow \begin{cases} 1.\ P_1 \rightarrow Q_1 \\ 2.\ P_2 \rightarrow Q_2 \\ \ldots\ldots\ldots\ldots\ldots \\ r.\ P_r \rightarrow Q_r \end{cases} \downarrow$$

Схема підстановок задає послідовність застосування підстановок до початкового слова

Дана схема задає алгоритм перетворення слів
Цей алгоритм називають нормальний алгоритм Маркова.

Визначення 3. *Нормальним алгоритмом (Маркова) в алфавіті* A називають правило побудови послідовності

$$\{V_i\}_{i=0}^{n}$$

слів з алфавіту A, виходячи з початкового слова V_0 з цього алфавіту.

0. $i = 0$- початкове значення змінної послідовності.

$$\begin{cases} 1 : P_1 \to Q_1 \\ 2 : P_2 \to Q_2 \\ \ldots\ldots\ldots\ldots\ldots \\ r : P_r \to Q_r \end{cases}$$

1. Нехай задане початкове слово V_0

2. Виконаємо першу можливу підстановку зі списку формул підстановок: $P_k \to Q_k$ при $1 \le k \le r$.

3. У результаті підстановки побудовано слово V_1 для деякого $i = 1$.

4. Відбувається аналіз умов завершення алгоритму.

5. Якщо умови завершення алгоритму **не виконані**, то переходимо до п.1 алгоритму із заміною початкового слова на слово $V_{i-1} \to V_i$.
6. Якщо умова завершення виконана, то алгоритм завершує свою роботу.

Умови завершення нормального алгоритму Маркова

У рамках даної лекції розглядаються дві умови завершення алгоритму Маркова:

нетермінальна та термінальна.

1. Нетермінальна умова. Якщо у списку підстановок нормального алгоритму немає формул, ліві частини яких входили б у V_i, то V_{i+1} вважають таким, що дорівнює V_i, і процес побудови послідовності вважають таким, що завершився.

$$V_{i+1} = V_i;\quad V_{i+2} = V_{i+1} = V_i;\ldots\ldots;\ V_{i+n} = V_{i+n-1} = \ldots = V_i$$

2. Термінальна умова. Якщо в результаті чергової підстановки на даному кроці була застосована формула заключної підстановки, подальші кроки нормального алгоритму Маркова припиняються. $P \to .Q$

Застосовність нормального алгоритму Маркова

Якщо процес побудови послідовності підстановок зупиняється, то розглянутий *нормальний алгоритм застосовний до слова* V_0.

(тобто виконується або термінальна або нетермінальна умова зупинки)

Останній член W послідовності називається *результатом застосування нормального алгоритму до слова* V_0.

Говорять, що *нормальний алгоритм перетворює* V_0 *в* W.

Правила для формального запису нормального алгоритму Маркова

Нехай задано алфавіт $A = \{a_1, a_2, \ldots, a_n\}$

Задано початкове слово $V_0 \subseteq A$

Нормальний алгоритм в алфавіті А задають послідовністю.

Послідовність V_i будемо записувати в такий спосіб:

Упорядкована схема підстановок

$$\begin{cases} P_1 \to Q_1 \\ P_2 \to Q_2 \\ \ldots\ldots\ldots\ldots\ldots \\ P_r \to Q_r \end{cases}$$

$$V_0 \Rightarrow V_1$$
$$V_1 \Rightarrow V_2$$
$$\ldots$$
$$V_{m-1} \Rightarrow V_m$$

де $V_0 = V$ і $V_m = W$

Якщо ж алгоритм заданий у деякому розширенні $B = A \cup E$ алфавіту A , то говорять, що він *є нормальним алгоритмом над* A.

Розглянемо приклади нормальних алгоритмів.

Упорядкована схема підстановок

$$\begin{cases} P_1 \to Q_1 & \text{пріоритет } 0 \\ P_2 \to Q_2 & \text{пріоритет } 1 \\ \cdots\cdots\cdots\cdots\cdots & \\ P_r \to Q_r & \text{пріоритет } r \end{cases}$$

Послідовність V_i

$$V_0 \Rightarrow V_1$$
$$V_1 \Rightarrow V_2$$
$$V_2 \Rightarrow V_3$$
$$\cdots$$
$$V_{m-1} \Rightarrow V_m$$

$$V_0 = \left(P_1\, P_2\, P_1\right)$$
$$V_1 = \left(Q_1\, P_2\, P_1\right)$$
$$V_2 = \left(Q_1\, P_2\, Q_1\right)$$
$$V_3 = \left(Q_1\, Q_2\, Q_1\right)$$

Приклад запису алгоритму

$$\left(P_1\, P_2\, P_1\right) \Rightarrow \left(Q_1\, P_2\, P_1\right)$$
$$\left(Q_1\, P_2\, P_1\right) \Rightarrow \left(Q_1\, P_2\, Q_1\right)$$
$$\left(Q_1\, P_2\, Q_1\right) \Rightarrow \left(Q_1\, Q_2\, Q_1\right)$$

Приклад 1. Нехай $A = \{a, b\}$ — алфавіт. Розглянемо наступну схему підстановок нормального алгоритму в A:

$$\begin{cases} a \to \Lambda, & k = 1 \\ bb \to \Lambda, & k = 2 \end{cases}$$

Заданий цією схемою нормальний алгоритм працює так:

1. Виберемо початкове слово $aababb$ і підстановку ($k = 1$).
2. Проглядаємо слово зліва направо.
3. Шукаємо можливість підстановки з номером k.
4. Переходимо до п.3, якщо одержали нове слово.
5. Змінюємо $k := k + 1$. Переходимо до п.3.

Пункти 3-5 будемо виконувати поки не буде досягнута нетермінальна умова завершення

$$aababb \Rightarrow ababb$$
$$ababb \Rightarrow babb$$
$$babb \Rightarrow bbb$$
$$bbb \Rightarrow b$$

Приклад 2. Нормальний алгоритм в алфавіті $A = \{a, b, 1\}$ задамо схемою:

$$\begin{cases} a \to 1 \\ b \to 1 \end{cases}$$

Застосуємо його до слова $abaabbb$.

$abaabbb \Rightarrow 1baabbb$
$1baabbb \Rightarrow 1b1abbb$
$1b1abbb \Rightarrow 1b11bbb$
$1b11bbb \Rightarrow 1111bbb$
$1111bbb \Rightarrow 11111bb$
$11111bb \Rightarrow 111111b$
$111111b \Rightarrow 1111111$

До отриманого слова 1111111 жодна з підстановок даної схеми вже не застосовна. Отже, робота алгоритму завершена. (Нетермінальна зупинка)

Приклад 3. Нормальний алгоритм в алфавіті $A = \{a, b\}$ задано схемою:

$$\begin{cases} ab \to a, \\ b \to \Lambda, \\ a \to b \end{cases}$$

Проаналізуємо роботу алгоритму на слові $abbbaaab$.

$ab\,bbaaab \Rightarrow abbaaab$
$ab\,baaab \Rightarrow abaaab$
$ab\,aaab \Rightarrow aaaab$
$aaaab \Rightarrow aaaa$
$a\,aaa \Rightarrow baaa$
$b\,aaa \Rightarrow aaa$
$a\,aa \Rightarrow baa$
$b\,aa \Rightarrow aa$
$a\,a \Rightarrow ba$
$b\,a \Rightarrow a$
$a \Rightarrow b$
$b \Rightarrow \Lambda.$

Якщо брати як V будь-які слова, складені з символів алфавіту A, легко дійти висновку, що даний алгоритм довільне слово перетворює в порожній рядок.

Нормально обчислювані функції і принцип нормалізації Маркова

Як і машини Тьюринга, нормальні алгоритми не здійснюють обчислень: вони лише перетворюють слова, заміняючи в них одні букви іншими за запропонованими їм правилами.

У свою чергу, ми пропонуємо їм такі правила, результати застосування яких ми можемо інтерпретувати як обчислення.

Розглянемо приклади, де числа представлені в кодуванні чисел для машини Тьюринга відповідною кількістю одиниць.

Приклад 4. В алфавіті $A = \{1\}$ задамо схему $1 \to .11$

Крапка перед одиницею означає, що дана заміна виконується заключно.

Проілюструємо роботу алгоритму.

1) $1 \to .11$ 0+1=1

2) $11 \to .111$ 1+1=2

3) $111 \to .1111$ 2+1=3

……..

Алгоритм приписує 1. Отже, алгоритм реалізує (обчислює) функцію слідування

$$S(x) = x + 1.$$

Отже, алгоритм Маркова здатний реалізувати функцію слідування в стилі машини Тьюринга.

Приклад 5. В алфавіті $A = \{1\}$ задамо схему: $11 \to 1$

Розглянемо роботу алгоритму з початковим словом 11111.

$$\begin{cases} 11111 \to 1111 & 4 \to 3 \\ 1111 \to 111 & 3 \to 2 \\ 111 \to 11 & 2 \to 1 \\ 11 \to 1 & 1 \to 0 \end{cases}$$

У результаті роботи алгоритму слово, що кодує число 4, перетворено в слово, що кодує число 0.

Алгоритм зупинився через відсутність формули підстановки з підходящою лівою частиною.

Отже, даний нормальний алгоритм Маркова реалізує нуль-функцію в стилі машини Тьюринга: $0(x) = 0$.

Приклад 6. В алфавіті $A = \{\Lambda, 1\}$ задамо схему:

$\Lambda 1 \to \Lambda$.

Нехай задане початкове слово $1111\Lambda 111$.

Розглянемо роботу нормального алгоритму Маркова.

$$\begin{cases} 1111\Lambda 111 \to 1111\Lambda 11 \\ 1111\Lambda 11 \to 1111\Lambda 1 \\ 1111\Lambda 1 \to 1111 \end{cases}$$

У результаті роботи алгоритму слово, що кодує вхідні значення аргументів функції проектування $I_1^2(3,2)$ перетворено в слово 3, що є результатом роботи функції проектування.

Приклад 7. Дана функція

$$\varphi\left(111...1\right)=\begin{cases}1,\ 1^{n+1}\ \bmod 2=\Lambda,\\ \Lambda,\ 1^{n+1}\ \bmod 2\neq 1^{1}\end{cases}$$

де n – число в десятковій системі числення.

Розглянемо нормальний алгоритм в алфавіті $A=\left\{1\right\}$ з наступною схемою:

$$11\rightarrow\Lambda$$

Цей алгоритм працює за таким принципом: алгоритм послідовно стирає по дві одиниці.

1. Якщо число одиниць парне, тобто у числовому кодування задане непарне число, то результатом алгоритму буде Λ.
2. Якщо число одиниць непарне, тобто у числовому кодування задане парне число, то результатом алгоритму буде 0, що предсталене числовим кодом, як $1^{1}=1$.

Наприклад

$11111111 \Rightarrow 11111$ 6 mod 2 =0

$11111 \Rightarrow 111$

$111 \Rightarrow 1$

Вхідне число 6 є парним числом. Отже, результат роботи алгоритму 1.

Наступний приклад виконує перевірку 9.

$111111111 \Rightarrow 11111111$ 9 mod 2 =1

$11111111 \Rightarrow 111111$

$111111 \Rightarrow 1111$

$1111 \Rightarrow 11$

$11 \Rightarrow \Lambda$

Число 9 ділиться не на 2 без остачі. Отже слово, що є результатом —Λ .

Обчислювана функція за Марковим

Фіксований алфавіт

Визначення 4. Функція f, задана на деякій множині слів алфавіту A, називається нормально обчислюваною у вхідному алфавіті, якщо кожне слово V в алфавіті A з області визначення функції f нормальний алгоритм Маркова перетворює в слово $f\left(V\right)$.

Розширений алфавіт

Визначення 5. Функція f, задана на деякій множині слів алфавіту A, називається нормально обчислюваною, якщо існує таке розширення B даного алфавіту $\left(A \subseteq B\right)$ й такий нормальний алгоритм в B, що кожне слово V (в алфавіті A) з області визначення функції f цей алгоритм перетворює в слово $f\left(V\right)$.

ПІДСУМОК

Таким чином, нормальні алгоритми прикладів 5-7 показують, що функції

$S\left(x\right) = x + 1,$

$0\left(x\right) = 0,$

$I_m^n\left(x_1,...,x_n\right) = x_m,$

$\varphi\left(x\right)$

нормально обчислювані на вхідному алфавіті.

Відповідні нормальні алгоритми вдалося побудувати в тому ж самому алфавіті A, на словах якого були задані розглянуті функції.

Приклад алгоритму обчислюваної функції на розширеному алфавіті

Приклад 8. Побудуємо нормальний алгоритм для обчислення функції $S\left(x\right) = x + 1$ у десятковій системі числення.

Представимо алфавіт цифрами $A = \left\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\right\}$.

Нормальний алгоритм Маркова будемо будувати з використанням розширення

$$B = A \cup \left\{a, b\right\}.$$

Для реалізації алгоритму виберемо список формул підстановок.

Продовження прикладу 8.

Схема даного алгоритму має вигляд (читати по стовпцях):

1	$0b \to .1$	12	$a0 \to 0a$	23	$1a \to 1b$
2	$1b \to .2$	13	$a1 \to 1a$	24	$2a \to 2b$
3	$2b \to .3$	14	$a2 \to 2a$	25	$3a \to 3b$
4	$3b \to .4$	15	$a3 \to 3a$	26	$4a \to 4b$
5	$4b \to .5$	16	$a4 \to 4a$	27	$5a \to 5b$
6	$5b \to .6$	17	$a5 \to 5a$	28	$6a \to 6b$
7	$6b \to .7$	18	$a6 \to 6a$	29	$7a \to 7b$
8	$7b \to .8$	19	$a7 \to 7a$	30	$8a \to 8b$
9	$8b \to .9$	20	$a8 \to 8a$	31	$9a \to 9b$
10	$9b \to b0$	21	$a9 \to 9a$	32	$\Lambda \to a$
11	$b \to .1$	22	$0a \to 0b$		

Продовження прикладу 8.

Спробуємо застосувати алгоритм до порожнього слова Λ.

Зрозуміло, що на кожному кроці повинна застосовуватися **«остання»** формула даної схеми: $\Lambda \to a$ Отримуємо нескінченний процес:

$$\Lambda \Rightarrow a$$

$$a\Lambda \Rightarrow aa$$

$$aa\Lambda \Rightarrow aaa$$

.....

Це означає, що до порожнього слова даний алгоритм не застосовний.

Якщо застосувати тепер алгоритм до слова Λ499, одержимо наступну послідовність слів:

Продовження прикладу 8.

$$\Lambda 499 \Rightarrow a499 \qquad \Lambda \to a$$

$$a499 \Rightarrow 4a99 \qquad a4 \to 4a$$

$$4a99 \Rightarrow 49a9 \qquad a9 \to 9a$$

$$49a9 \Rightarrow 499a \qquad a9 \to 9a$$

$$499a \Rightarrow 499b \qquad 9a \to 9b$$

$$499b \Rightarrow 49b0 \qquad 9b \to b0$$

$$49b0 \Rightarrow 4b00 \qquad 9b \to b0$$

$$4b00 \Rightarrow .500 \qquad 4b \to .5$$

У розглянутому прикладі нормальний алгоритм побудований в алфавіті B, що є істотним розширенням алфавіту A (тобто $A \subseteq B$ й $A \neq B$)

Даний алгоритм слова в алфавіті A перетворює знову в слова в алфавіті A. У такому випадку говорять, що алгоритм заданий *над* алфавітом *A*.

Приклад 7. В алфавіті $B = A \cup \{a, b\}$, що є розширенням алфавіту A, розглянемо нормальний алгоритм, що задається схемою (читати по стовпцях):

1	$0b \to b9$	11	$a0 \to 0a$	21	$0a \to 0b$
2	$1b \to .0$	12	$a1 \to 1a$	22	$1a \to 1b$
3	$2b \to .1$	13	$a2 \to 2a$	23	$2a \to 2b$
4	$3b \to .2$	14	$a3 \to 3a$	24	$3a \to 3b$
5	$4b \to .3$	15	$a4 \to 4a$	25	$4a \to 4b$
6	$5b \to .4$	16	$a5 \to 5a$	26	$5a \to 5b$
7	$6b \to .5$	17	$a6 \to 6a$	27	$6a \to 6b$
8	$7b \to .6$	18	$a7 \to 7a$	28	$7a \to 7b$
9	$8b \to .7$	19	$a8 \to 8a$	29	$8a \to 8b$
10	$9b \to .8$	20	$a9 \to 9a$	30	$9a \to 9b$
				31	$\Lambda \to a.$

Дана схема дозволяє реалізувати нормальний алгоритм обчислення функції $f\left(x\right) = x - 1$.

Розв'язок

Перевіримо роботу алгоритму, застосувавши його до слова $\Lambda 3000$.

$$\Lambda 3000 \Rightarrow a3000 \qquad \Lambda \to a$$
$$a3000 \Rightarrow 3a000 \qquad a3 \to 3a$$
$$3a000 \Rightarrow 30a00 \qquad a0 \to 0a$$
$$30a00 \Rightarrow 300a0 \qquad a0 \to 0a$$
$$300a0 \Rightarrow 3000a \qquad a0 \to 0a$$
$$3000a \Rightarrow 3000b \qquad 0a \to 0b$$
$$3000b \Rightarrow 300b9 \qquad 0b \to b9$$
$$300b9 \Rightarrow 30b99 \qquad 0b \to b9$$
$$30b99 \Rightarrow 3b999 \qquad 0b \to b9.$$
$$3b999 \Rightarrow .2999 \qquad 3b \to .2$$

Принцип нормалізації Маркова

Творець теорії нормальних алгоритмів Андрій Андрійович Марков висунув гіпотезу, що одержала назву **«Принцип нормалізації Маркова».**

Згідно із принципом нормалізації

Для знаходження значень функції, заданої в деякому алфавіті, тоді й тільки тоді існує який-небудь алгоритм, коли функція нормально обчислювана.

Збіг класу всіх нормально обчислюваних функцій із класом усіх функцій, обчислюваних за Тьюрингом.

Поняття нормально обчислюваної функції рівносильне поняттю функції, обчислюваної **за Тьюрингом**, а разом з ним і поняттю частково рекурсивної функції.

Теорема 1. *Будь-яка функція, обчислювана за Тьюрингом, буде також і нормально обчислюваною.*

Доведена також зворотна теорема.

Теорема 2. *Будь-яка нормально обчислювана функція обчислювана за Тьюрингом*.

Еквівалентність різних універсальних алгоритмічних моделей

Нами *розглянуто* три алгоритмічні моделі, кожна з яких уточнює поняття алгоритму, і *з'ясовано*, що всі ці моделі рівносильні між собою. Отже, вони описують *один і той же* клас функцій, тобто справедливою є наступна теорема.

Теорема 3. *Наступні класи функцій (що задані на натуральних числах та набувають натуральних значень) збігаються:*

а)клас усіх функцій, обчислюваних за Тьюрингом;

б)клас усіх частково рекурсивних функцій;

в)клас усіх нормально обчислюваних функцій.

Можна відзначити, що існують ще й інші, менш відомі алгоритмічні моделі, і для них також доведена рівносильність із розглянутими теоріями.