ТЕОРІЯ АЛГОРИТМІВ

Теорія алгоритмів – розділ математики, що вивчає загальні властивості алгоритмів.

Алгоритм – скінченна множину точно визначених правил для чисто механічного розв'язку задач певного класу.

Характерні властивості алгоритму:

- *Фінітність*
- *Масовість*
- *Дискретність*
- *Елементарність*
- *Результативність*
- *Детермінованість*

Для опису алгоритму вказуємо:

– множину його *початкових (вхідних)* даних

– множину *вихідних* даних, до яких належать результати роботи алгоритму.

За допомогою алгоритму кожний конкретний результат отримується за скінченну кількість кроків із скінченної множини вхідних даних.

До таких даних алгоритм *застосовний*.

В деяких ситуаціях процес виконання алгоритму для певних вхідних даних продовжується необмежено.

До таких даних алгоритм *незастосовний.*

Кожний алгоритм $\aleph$ із множиною вхідних даних X та вихідних Y визначає часткову функцію $f : X \to Y$.

Якщо $\aleph$ застосовний до d, то значення $f(d)$ рівне $\aleph(d)$.

Якщо $\aleph$ незастосовний до d, то $f(d)$ невизначене.

Такий алгоритм $\aleph$ *обчислює* функцію f.

Функція *алгоритмічно обчислювана* (АОФ), якщо існує алгоритм, який її обчислює.

Множина L *алгоритмічно перелічна*, якщо L є множиною значень деякої АОФ, тобто існує алгоритм, який перелічує елементи множини L і тільки їх.

Множина L *алгоритмічно розв'язна* відносно множини U, якщо існує алгоритм, який дозволяє для кожного $x \in U$ визначати, $x \in L$ чи $x \notin L$.

Відносний алгоритм (*алгоритм з оракулом*): на деяких кроках може звертатися до зовнішнього відносно алгоритму об'єкту – оракула. Видані оракулом відповіді – це дані, вироблені на таких кроках звертання.

Функція *алгоритмічно обчислювана відносно оракула* $\wp$, якщо існує алгоритм з оракулом $\wp$, який її обчислює.

Теорія алгоритмів як окремий розділ математики виникла в 30-х рр. 20 ст.

ТА сформувалась як розділ МЛ, перші її застосування – саме в МЛ.

Засобами ТА доведено *алгоритмічну нерозв'язність* проблем істинності формул логіки 1-го порядку, істинності арифметичних формул.

Поняття алгоритму тісно пов’язане з поняттям *числення*

Поняття алгоритму можна звести до поняття числення в смислі зведення алгоритмічного процесу до процесу *породження*: кожний алгоритм можна трактувати як числення з такими ПВ, що виконання кожного із них відповідає виконанню одного кроку алгоритму.

Поняття числення можна звести до поняття алгоритму в смислі зведення *розгалуженого* процесу породження до *послідовного* процесу переліку так, щоб алгоритм переліку відтворив усі породжені численням об’єкти і тільки їх.

Усвідомлення *неможливості* існування алгоритмів розв'язку низки масових проблем ⇒ необхідність *математичного* уточнення поняття алгоритму. Тому після сформування поняття алгоритму як *нової та окремої сутності* першочерговою стала проблема знаходження адекватних *формальних моделей* алгоритму та дослідження їх властивостей. Було запропоновано:

– моделі для первісного поняття алгоритму (машини Тьюрінга, регістрові машини, нормальні алгоритми Маркова)
– моделі для похідного поняття АОФ (λ-означувані функції, частково рекурсивні функції та ін.).

Доведено: кожна з відомих моделей задає (з точністю до кодування) один і той же клас функцій.

Тому є всі підстави вважати, що кожна з таких моделей дає строге математичне уточнення інтуїтивного поняття АОФ.

Таке твердження стосовно АОФ та строго визначеного класу частково рекурсивних функцій – ***теза Чорча*** (А.Чорч, 1936).

МАШИНИ З НАТУРАЛЬНОЗНАЧНИМИ РЕГІСТРАМИ

МНР є ідеалізованою моделлю комп'ютера.

МНР складається з регістрів, вмістом яких є натуральні числа.

Регістри позначаємо так: $R_0, R_1,..., R_n,...$

Вміст регістру R_n позначимо $'R_n$

Конфігурація МНР – це послідовність $('R_0, 'R_1,..., 'R_n,...)$ вмістів регістрів МНР.

МНР-програма – скінченна послідовність команд МНР.

Команди програми послідовно нумеруємо, починаючи з 1.

А*дреса* команди – це номер команди в МНР-програмі.

$I_1I_2...I_k$ – МНР-програма з командами $I_1, I_2,..., I_k$

$|P|$ – довжина (кількість команд) МНР-програми P.

Команди МНР

Тип 1. Обнулення n-го регістру $Z(n)$: $'R_n := 0$.

Тип 2. Збільшення вмісту n-го регістру на 1 $S(n)$: $'R_n := 'R_n + 1$.

Тип 3. Копіювання вмісту регістру $T(m, n)$: $'R_n := 'R_m$
($'R_m$ не змінюється).

Команди типів 1–3 – арифметичні.

Наступник арифметичної команди – наступна команда програми.

Тип 4. Умовний перехід $J(m, n, q)$: $'R_n = 'R_m \Rightarrow$ перейти до q-ї команди,
інакше виконувати наступну команда програми

Тут q – адреса переходу

Крок МНР – виконання однієї команди МНР.

Саме *МНР-програми* є формальними моделями алгоритмів.

Поняття МНР використовуємо для опису функціонування МНР-програм

Виконання програми МНР починає з виконання першої команди.

Наступна для виконання команда програми – як описано вище.

Виконання програми завершується, якщо наступник відсутній

(номер наступної команди > номера останньої команди програми).

Фінальна конфігурація МНР – у момент завершення виконання програми.

Вона визначає результат роботи МНР-програми над даною початковою конфігурацією.

$P(a_0, a_1, ...)\uparrow$: МНР-програма P не зупиняється при роботі над початковою конфігурацією $(a_0, a_1,...)$

$P(a_0, a_1, ...)\downarrow$: МНР-програма P зупиниться при роботі над початковою конфігурацією $(a_0, a_1,...)$

$P(a_0, a_1,...)\downarrow(b_0, b_1,...)$: МНР-програма P при роботі над початковою конфігурацією $(a_0, a_1,...)$ зупиняється з фінальною конфігурацією $(b_0, b_1,...)$

Кожна МНР-програма визначає однозначне відображення на множині послідовностей натуральних чисел.

МНР-програми – фінітні об'єкти $\Rightarrow \forall$ МНР-програма в процесі виконання використовує тільки скінченну множину регістрів, усі вони явно вказані у МНР-програмі.

Тому при розгляді відображень, заданих МНР-програмами, доцільно обмежитися скінченними послідовностями натуральних чисел.

Отже, далі розглядаємо тільки ***скінченні*** конфігурації.

Якщо МНР-програма P починає роботу над скінченною початковою конфігурацією, то в процесі виконання P МНР перебуватиме тільки в скінченних конфігураціях.

Конфігурацію $(a_0, a_1, ..., a_n, 0, 0, ...)$, у якій $'R_m = 0 \ \forall \ m > n$,

позначаємо $(a_0, a_1, ..., a_n)$

МНР-програми P та Q *еквівалентні*, якщо вони визначають однакові відображення послідовностей натуральних чисел.

Це означає: при роботі над однаковими початковими конфігураціями вони або обидві зупиняються з однаковими фінальними конфігураціями, або обидві не зупиняються.

МНР-програма P *стандартна*, якщо

$q \le |P| + 1$ $\forall$ команди вигляду $J(m, n, q)$.

Конкатенація стандартних МНР-програм $P = I_1 I_2 ... I_k$ та $Q = I_1 I_2 ... I_m$ –

це стандартна МНР-програма $I_1 ... I_k I_{k+1} ... I_{k+m}$,

де $I_{k+1}, ..., I_{k+m}$ – команди програми Q, у яких

$\forall$ команда $J(m, n, q)$ замінена командою $J(m, n, q + k)$.

МНР-програма P *обчислює* часткову n-арну функцію $f : N^n \to N$:

$f(a_1, a_2, ..., a_n) = b \Leftrightarrow P(a_1, a_2, ..., a_n)\downarrow b.$

Пишемо $P(a_1, a_2, ...)\downarrow b$ замість $P(a_1, a_2, ...)\downarrow(b, ...)$

– значення аргументів посл-но розміщуються в регістрах, поч-чи із R_0

– значення функції знімається з R_0.

Еквівалентне визначення обчислюваності $f : N^n \to N$ МНР-програмою P:

– за умови $(a_1, a_2, ..., a_n) \in D_f$ та $f(a_1, a_2, ..., a_n) = b$ $P(a_1, a_2, ..., a_n)\downarrow b$;

– за умови $(a_1, a_2, ..., a_n) \notin D_f$ маємо $P(a_1, a_2, ..., a_n)\uparrow$.

$f : N^n \to N$ *МНР-обчислювана*, якщо існує МНР-програма, яка обчислює f.

$\forall$ МНР-програма обчислює безліч функцій нат. аргументів і значень.

Фіксуємо наперед *арність* функцій (к-ть компонент початкових конфігурацій) $\Rightarrow$ $\forall$ МНР-програма обчислює *єдину* функцію заданої арності.

Приклад 1. МНР-програма для функції $x + y$:

1) $J(1,2,5)$

2) $S(0)$

3) $S(2)$

4) $J(0,0,1)$

Приклад 2. МНР-програма для функції $x - y$:

1) $J(0,1,5)$

2) $S(1)$

3) $S(2)$

4) $J(0,0,1)$

5) $T(2,0)$

Приклад 3. МНР-програма для всюди невизначеної функції:

1) $J(0,0,1)$

Приклад 4. МНР-програма для функції $[x/2]$:

1) $J(0,2,7)$
2) $S(2)$
3) $J(0,2,7)$
4) $S(2)$
5) $S(1)$
6) $J(0,0,1)$
7) $T(1,0)$

Приклад 5. МНР-програма для функції $f(x,y) = x \dot{-} y$:

1) $J(0,1,7)$

2) $J(0,2,6)$

3) $S(1)$

4) $S(2)$

5) $J(0,0,1)$

6) $Z(2)$

7) $T(2,0)$

Приклад 6. МНР-програма для функції $f(x, y) = x \cdot y$:

1) $J(3,1,9)$

2) $J(0,2,6)$

3) $S(2)$

4) $S(4)$

5) $J(0,0,2)$

6) $Z(2)$

7) $S(3)$

8) $J(0,0,1)$

9) $T(4,0)$

Нехай P – стандартна МНР-програма для n-арної функції f.

Найбільший номер регістру, задіяного при обч-ні f, позначимо $\rho(P)$.

Для n-арної функції вважатимемо $\rho(P) \geq n - 1$.

$P[k_1, k_2,..., k_n \rightarrow R]$ позначимо МНР-програму, яка обчислює ту саму f, що й МНР-програма P, але за умови:

– початкові значення аргументів занесені в регістри $k_1,..., k_n$,

– значення функції знімається з регістру R.

Для обчисл-я f задіяно найбільше $\rho(P) + 1$ регістрів – з 0-го по $\rho(P)$-й.

МНР-програма $P[k_1, k_2,..., k_n \rightarrow R]$ для вип. $k_1 < k_2 <...< k_n$ має вигляд

$T(k_i, i - 1), 1 < i \leq n$

$Z(k), \ n \leq k \leq \rho(P)$

P'

$T(0, R)$

P' відрізняється від P тільки зміщенням адрес на $\rho(P)$.

МАШИНИ ТЬЮРІНГА

А.М.Тюрінг, Е.Пост 1936 р.

Варіанти МТ: багатострічкові, машини з еластичною стрічкою і т. п.

МТ – це (Q, T, δ, q_0, F), де:

– Q – скінченна множина внутрішніх станів;

– T – ск. алфавіт символів стрічки, символ порожньої клітини $\lambda \in T$

– $\delta : Q \times T \to Q \times T \times \{R, L, \varepsilon\}$ – функція переходів;

– $q_0 \in Q$ – початковий стан;

– $F \subseteq Q$ – множина фінальних станів.

Функцію переходів задають ск. множиною команд одного з типів:

$qa \to pbR$

$qa \to pbL$

$qa \to pb$

де $p, q \in Q$, $a, b \in T$, $\to \notin Q \cup T$.

Вважаємо δ тотальною $\Rightarrow$ $\forall$ пар $(q, a) \notin D_\delta$ неявно, не додаючи відп. команди $qa \to qa$, вводимо довизначення $\delta(q, a) = (q, a, \varepsilon)$

Неформально МТ:

– скінченна пам’ять

– розділена на клітини необмежена з обох боків стрічка

– голівка читання-запису.

У кожній клітині – єдиний символ $\in T$,

в $\forall$ момент стрічка містить скінченну кількість символів $\neq \lambda$.

Голівка читання-запису в $\forall$ момент оглядає єдину клітину стрічки.

Нехай МТ знаходиться в стані q та голівка читає символ a

– при виконанні команди $qa \rightarrow pbR$ МТ переходить у стан p, замість a записує на стрічці b та зміщує голівку на 1 клітину направо

– при виконанні команди $qa \rightarrow pbL$ – “ – “ – та на 1 клітину наліво

– при виконанні команди $qa \rightarrow pb$ – “ – “ – та залишає голівку на місці.

Конфігурація МТ – це слово xqy, де $x, y \in T^*, q \in Q$.

Неформально:

– на стрічці записано xy (зліва і справа від xy можуть стояти тільки λ),

– МТ знаходиться в стані q,

– голівка читає 1-й символ підслова y.

Конфігурація вигляду q_0x – *початкова*

Конфігурація вигляду xqy, де $q \in F$ – *фінальна*

Після переходу фінальної конфігурації, МТ зупиняється.

Нехай МТ знаходиться в конфіг. $xcqay$, де $x, y \in T^*, a, c \in T, q \in Q$.

Після виконання команди $qa \to pbR$ МТ перейде до конфіг. $xcbpy$.

Після виконання команди $qa \to pbL$ МТ перейде до конфіг. $xpcby$.

Після виконання команди $qa \to pb$ МТ перейде до конфіг. $xcpby$.

Кожна МТ задає деяке вербальне відображення $T^* \to T^*$.

МТ M переводить $u \in T^*$ у $v \in T^*$, якщо вона з початкової конфігурації $q_0 u$ переходить до фінальної xqy,

де $q \in F$, $xy = \alpha v \beta$, $\alpha, \beta \in \{\lambda\}^*$.

При цьому 1-й та останній символи слова v відмінні від λ, або $v = \varepsilon$.

МТ M переводить слово u в слово v: $v = M(u)$.

МТ M *зациклюється* при роботі над словом u: починаючи роботу з початкової конфігурації $q_0 u$, M ніколи не зупиниться.

Тоді $M(u)$ невизначене.

МТ M_1 та M_2 *еквівалентні*, якщо задають одне й те ж відображення.

МТ *детермінована*, якщо функція δ однозначна

інакше МТ *недетермінована*

Можна вважати, що F складається з єдиного фінального стану q^*:

Нехай $M=(Q, T, \delta, q_0, F)$. Візьмемо $q^* \notin Q$.

Тоді МТ $M'=(Q \cup \{q^*\}, T, \delta', q_0, \{q^*\})$, де

$\delta'=\delta \cup \{qa \to q^*a \mid q \in F, a \in T\}$,

еквівалентна початковій машині M.

Надалі – тільки детерміновані МТ з єдиним фінальним станом

позначаємо їх (Q, T, δ, q_0, q^*).

Конкретні МТ задаємо, указуючи множину команд

початковий стан позначаємо q_0, фінальний – q^*.

МТ M *обчислює* часткову функцію $f: N^n \to N$, якщо вона

– $\forall$ слово вигляду $|^{x_1} \#|^{x_2} \#...\#|^{x_k}$ переводить в $|^{f(x_1,...,x_k)}$ у випадку $(x_1,...,x_k) \in D_f$

– $M(|^{x_1} \#|^{x_2} \#...\#|^{x_k})$ невизначене у випадку $(x_1,..., x_k) \notin D_f$.

Функція *обчислювана за Тьюрінгом*, або *МТ-обчислювана*:

існує МТ, яка її обчислює.

Кожна МТ обчислює безліч функцій натуральних аргументів і значень

Зафіксуємо наперед арність функцій $\Rightarrow$ $\forall$ МТ обчислює єдину функцію заданої арності.

Приклад 1. МТ, яка обчислює функцію $x + y$:

$q_0|\rightarrow q_1\lambda R$

$q_1|\rightarrow q_1|R$

$q_1\#\rightarrow q^*|$

$q_0\#\rightarrow q^*\lambda$

Приклад 2. МТ, яка обчислює функцію $f(x) = sg(x)$:

$q_0\lambda\rightarrow q^*\lambda$

$q_0|\rightarrow q_1|R$

$q_1|\rightarrow q_1\lambda R$

$q_1\lambda\rightarrow q^*\lambda$

Приклад 3. МТ, яка обчислює предикат "x парне":

$q_0|\rightarrow q_1\lambda R$

$q_1|\rightarrow q_0\lambda R$

$q_0\lambda\rightarrow q^*|$

$q_1\lambda\rightarrow q^*\lambda$

Приклад 4. МТ, яка обчислює функцію $x - y$:

$q_0|\rightarrow q_1\lambda R$

$q_1|\rightarrow q_1|R$

$q_1\#\rightarrow q_1\#R$

$q_1\lambda\rightarrow q_2\lambda L$

$q_2|\rightarrow q_3\lambda L$

$q_3|\rightarrow q_3|L$

$q_3\#\rightarrow q_3\#L$

$q_3\lambda\rightarrow q_0\lambda R$

$q_2\#\rightarrow q^*|$

$q_0\#\rightarrow q_4\lambda R$

$q_4\lambda\rightarrow q^*\lambda$

Приклад 5. МТ, яка кожне слово $x \in T^*$ переводить у слово $x\#x$, $\# \notin T$

$q_0 t \to q_0 tR$ для всіх $t \in T$

$q_0 \lambda \to q_1 \# L$

$q_1 t \to q_1 tL$ для всіх $t \in T$

$q_1 \lambda \to q_2 \lambda R$

$q_2 t \to q_t \lambda R$ для всіх $t \in T$

$q_t p \to q_t pR$ для всіх $t \in T, p \in T \cup \{\#\}$

$q_t \lambda \to q'_t tL$ для всіх $t \in T$

$q'_t p \to q'_t pL$ для всіх $t \in T, p \in T \cup \{\#\}$

$q'_t \lambda \to q_2 tR$ для всіх $t \in T$

$q_2 \# \to q^* \#$

НОРМАЛЬНІ АЛГОРИТМИ МАРКОВА

НА *в алфавіті* T – упорядкована послідовність продукцій (правил) вигляду

$\alpha \to \beta$ або

$\alpha \to \cdot \beta$,

де $\alpha, \beta \in T^*$ та $\cdot \notin T$, $\to \notin T$.

Продукції вигляду $\alpha \to \cdot \beta$ – *фінальні*.

Кожен НА в алфавіті T задає деяке вербальне відображення $T^* \to T^*$

$\aleph(x)$ – слово, яке є результатом обробки слова x НА $\aleph$

Обробка слова x нормальним алгоритмом $\aleph$ – поетапно.

Покладемо $x_0 = x$ і скажемо: x_0 отримано з x після 0 етапів.

Нехай x_n отримано з x після n етапів. Тоді (n+1)-й етап вик-ся так.

Шукаємо першу за порядком $\alpha \to \beta$ або $\alpha \to \cdot \beta$ таку, що α – підслово x_n.

Застосуємо цю продукцію до x_n – замінимо в x_n найлівіше входження α на β.

Отримане слово позначимо x_{n+1}.

– застосована на (n+1)-етапі продукція не фінальна $\Rightarrow$ перехід до (n+2)-етапу

– ця продукція фінальна $\Rightarrow$

після її застосування $\aleph$ зупиняється, $\aleph(x) = x_{n+1}$.

– на (n+1)-етапі жодна продукція $\aleph$ не застосовна до x_{n+1}, тобто в $\aleph$ немає продукції, ліва частина якої – підслово слова x_{n+1} $\Rightarrow$

$\aleph$ зупиняється, $\aleph(x) = x_n$.

Якщо в процесі обробки x НА $\aleph$ не зупиняється на жодному етапі, то $\aleph(x)$ невизначене.

НА називають *нормальним алгоритмом над алфавітом* T, якщо він є НА у деякому розширенні $T_1 \supseteq T$.

НА над T задає $T^* \to T^*$, використовуючи допоміжні символи $\notin T$.

Зупинка НА $\aleph$ над T при роботі над $x \in T^*$ *результативна,* якщо вона відбулась на слові $y \in T^*$, інакше результат роботи $\aleph$ над x невизначений.

НА $\aleph$ і $\Re$ *еквівалентні відносно алфавіту* T, якщо $\forall\ x \in T^*$

– $\aleph(x)\uparrow$ та $\Re(x)\uparrow$ або

– $\aleph(x)\downarrow$ та $\Re(x)\downarrow$ та $\aleph(x) = \Re(x)$

$\forall$ НА над T існує еквівалентний йому відносно T НА в $T \cup \{s\}$ із єдиним допоміжним $s \notin T$.

Відомо, що вербальне відображення $x \Rightarrow xx$, $x \in T^*$, не може бути заданим жодним НА в алфавіті T.

НА $\aleph$ *обчислює* часткову функцію $f: N^m \to N$, якщо він

– $\forall$ слово вигляду $|^{x_1} \#|^{x_2} \#...\#|^{x_k}$ переводить в $|^{f(x_1,...,x_k)}$ у випадку $(x_1,...,x_k) \in D_f$

– $\aleph(|^{x_1} \#|^{x_2} \#...\#|^{x_k})$ невизначене при $(x_1,..., x_k) \notin D_f$

Функція *обчислювана за Марковим*, або *НА-обчислювана*:

існує НА, який її обчислює.

Кожний НА обчислює безліч функцій натур. аргументів і значень

Зафіксуємо наперед арність функцій $\Rightarrow$ $\forall$ НА обчислює єдину функцію заданої арності.

Приклад 1. НА для функції $f(x, y) = x + y$:

$\#\rightarrow\varepsilon$

Приклад 2. НА для функції $x - y$:

$|\#|\rightarrow\#$

$\#|\rightarrow\#|$

$\#\rightarrow\varepsilon$

Приклад 3. НА для функції $x/2$:

$\#||\rightarrow|\#$

$\#|\rightarrow\#|$

$\#\rightarrow\cdot\varepsilon$

$\varepsilon\rightarrow\#$

Приклад 4. НА, який задає $a^x b^y \Rightarrow |^{x \cdot y}$:

$ab \rightarrow ba|$

$|b \rightarrow b|$

$a \rightarrow \varepsilon$

$b \rightarrow \varepsilon$

Приклад 5. НА для функції $x \cdot y$:

$\# \rightarrow **$

$|* \rightarrow *a$

$*| \rightarrow b*$

$* \rightarrow \varepsilon$

$ab \rightarrow ba|$

$|b \rightarrow b|$

$a \rightarrow \varepsilon$

$b \rightarrow \varepsilon$

Приклад 6. НА для функції 2^x в розширеному кодуванні:

$|a \to a||$

$a| \to |$

$| \to \cdot |$

$\varepsilon \to |$

Приклад 7. НА для функції 2^x:

$*|\to|**$

$|*\to*$

$\#*\to|\#$

$\#\to \cdot \varepsilon$

$*\to\#*$

$\varepsilon\to*$

Приклад 8. НА, який $\forall\ x \in T^*\ \ x \Rightarrow x^R$ (тут $\# \notin T$):

$\#ab \to b\#a \quad \forall\ a, b \in T$

$\#\#a\# \to a\#\# \quad \forall\ a \in T$

$\#\# \to \cdot\, \varepsilon$

$\varepsilon \to \#$

Приклад 9. НА, який $\forall\ x \in T^*\ \ x \Rightarrow xx$ (тут $\# \notin T$):

$\#\#a \to a\#a\# \quad \forall\ a \in T$

$\#ab \to b\#a \quad \forall\ a, b \in T$

$\#a \to a \quad \forall\ a \in T$

$\#\# \to \cdot\, \varepsilon$

$\varepsilon \to \#\#$

Основна література

1. Катленд Н. **Вычислимость. Введение в теорию рекурсивных функций.** – М., 1983.
2. Клини С. **Математическая логика**. – М., 1973.
3. Лавров И.А., Максимова Л.Л. **Задачи по теории множеств, математической логике и теории алгоритмов.** – М., 1975.
4. Нікітченко М.С., Шкільняк О.С. Шкільняк С.С. **Теорія алгоритмів.** – К., 2015.
5. Мальцев А.И. **Алгоритмы и рекурсивные функции.** – М., 1965.
6. Нікітченко М.С., Шкільняк С.С. **Математична логіка та теорія алгоритмів.** – К., 2008.
7. Роджерс Х. **Теория рекурсивных функций и эффективная вычислимость.** – М., 1972
8. Шкільняк С.С. **Теорія алгоритмів. Приклади й задачі.** – К., 2012.

Додаткова література

1. Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов. – М., 1979
2. Гильберт Д., Бернайс П. Основания математики. Т.1, Т.2. – М., 1982.
3. Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л. Алгебра, языки, программирование.– К., 1974.
4. Ю.В.Капітонова, С.Л.Кривий, О.А.Летичевський, Г.М.Луцький, М.К.Печурін. Основи дискретної математики. – К., 2002.
5. Лисовик Л.П., Редько В.Н. Алгоритмы и формальные системы. – К., 1981.
6. Лісовик Л.П., Шкільняк С.С. Теорія алгоритмів. – К., 2003.
7. Манин Ю.И. Доказуемое и недоказуемое. – М., 1979.
8. Манин Ю.И. Вычислимое и невычислимое. – М., 1980.
9. Мендельсон Э. Введение в математическую логику. – М., 1976.
10. Непейвода Н.Н.. Прикладная логика. – Новосибирск: НГУ, 2000.
11. Нікітченко М.С. Теорія програмування. Частина 1. – Ніжин, 2010.
12. Нікітченко М.С., Панченко Т.В., Поляков С.А. Теорія програмування в прикладах і задачах. – К., 2015.
13. Нікітченко М.С., Шкільняк С.С. Основи математичної логіки. – К., 2006.
14. Справочная книга по математической логике (под ред. Дж.Барвайса). Ч.1– 4. – М., 1982–1983.
15. Успенский В.А., Семенов А.Л. Теория алгоритмов: основные открытия и приложения. – М., 1987.
16. Шенфилд Дж. Математическая логика. – М.: Наука, 1975.
17. Шкільняк С.С. Математична логіка: приклади і задачі. – К., 2007.