



# Агрегація

- SUM, AVG, COUNT, MIN, MAX

## Групування

$\delta_{\substack{\text{shop} \\ \text{sweet}}}, \text{AVG}(\text{Price}) (\text{Sells})$   
 $\text{Max}(\text{Price})$

## Outer join (зовн. з'єдн.)

$R \bowtie_c S - \text{з'єдн. по умові } C$

dangling - висорі рядки

R	S	A B C
A B	C D	1 2 3
1 2	2 3	4 5 Null
4 5	6 7	Null 6 7

Condition  $R.B = S.R$

- Left, Right, Full

Count(\*) - підрахунок всіх

Count(Price) - кількість не Null значень

\* --

Count Distinct Price - ~~уникальні~~  
інші ~~Null~~

Null іноруються в операції, яким Count

• **HAVING** - заснов. наявне групування

• чибо заснов. на Конкретній групі.

• **DEFAULT Values**

• при створенні

< > - не !=

D/B Truncate table..

TRUNCATE - операція видалення усіх рядків таблиці.

чиа різнице big Delete From

Insert into PetBuddies

Select d2.Person

From Frequent d1, Frequent d2.

Where de.person = 'Selly'

AND de.shop = d2.shop

AND d2.shop <> 'Selly'

D/3 uc nabit Yaponia mpeda banan zohero urod elanne  $2^{63}-1$  jopen

EXISTS -

ORDER BY (DESC)

Insert, Delete, Update mungkince y 2 elmanu.

The **PRIMARY KEY** constraint uniquely identifies each record in a table.

Primary keys must contain **UNIQUE** values, and cannot contain **NULL** values.

A table can have only **ONE** primary key; and in the table, this primary key can consist of single or multiple columns (fields).

The **FOREIGN KEY** constraint is used to prevent actions that would destroy links between tables.

A **FOREIGN KEY** is a field (or collection of fields) in one table, that refers to the **PRIMARY KEY** in another table.

The table with the foreign key is called the **child table**, and the table with the primary key is called the **referenced or parent table**.

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

# CHECK

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int CHECK (Age>=18)
);
```

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255),
    CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Sandnes')
);
```

# Transactions, Views, Indexes

max



min



del



ins

new transaction re

ii rischio ne faranno.

## COMMIT

## Isolation levels

- Serializable - non incide sui logici Acid
- Read Committed
- Repeatable-Read
- Read Uncommitted - ha incidenze, non leggono

## Views

- Virtual
- Materialized

## Trigger → posizioni

# Index

An index is a schema object. It is used by the server to speed up the retrieval of rows by using a pointer.

An index helps to speed up select queries and where clauses, but it slows down data input, with the update and the insert statements.

## SQL Constraints

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified
- CREATE INDEX - Used to create and retrieve data from the database very quickly

♡ HONIG + nAsiTibiy ♡

## З моделі БП

КОНЦЕНТУАЛЬНА - діє засвічене

ЛОГІЧНА - діє погранично

ФІЗИЧНА - реагування в БД

Аналіз засвічене - одні і місці засвічені  
зберіг. у 2 місцях.

Тристрічкова машина Тьюрінга - це модифікація класичної машини Тьюрінга, яка має три робочі стрічки замість однієї.

Це дозволяє збільшити потенціал обчислення та вирішувати більш складні завдання шляхом паралельної обробки інформації на кількох стрічках.

$$\begin{array}{r} 10 \\ - \lambda \\ \hline 1 \end{array}$$

$$\begin{array}{r} 100 \\ - 1 \\ \hline 11 \end{array}$$

$Q = \{q_0, q_1, q_2\}$  - множина станів

$S = \{1, 0, \lambda\}$  - здійснений алфавіт

$q_0$  $q_1$  $\times | \times | \times | 0 | \times | 1 | 0 | 1 | 0 | \lambda | \lambda | \lambda | \lambda | \lambda$  $\downarrow q_0$  $\downarrow q_2 \downarrow q_1$  $\times | \times | \times | 0 | 0 | 1 | 1 | \times | \times | \times | \lambda | \lambda | \lambda | \lambda$  $\times | \times | \times | \times | \times | \times | \times | \lambda | \lambda | \lambda | \lambda | \lambda | \lambda$  $\uparrow$  $q_{0,1},$   
 $q_2$ 

$$\begin{array}{r}
 1011010 \\
 - 10011 \\
 \hline
 1000111
 \end{array}$$

 $1000111$  $1 q_0 \rightarrow 1 q_S$  $q_0$  $\langle q_0 1, q_0 1, q_0 \lambda \rangle \Rightarrow \langle q_0 R, q_0 L, q_0 S \rangle$  $\langle q_0 1, 0, \lambda \rangle \Rightarrow \langle q_0 R, q_0 L, q_0 S \rangle$  $\langle q_0 0, 1, \lambda \rangle \Rightarrow \langle q_0 R, q_0 L, q_0 S \rangle$  $\langle q_0 0, 0, \lambda \rangle \Rightarrow \langle q_0 R, q_0 L, q_0 S \rangle$  $\langle q_0 \lambda, \lambda, \lambda \rangle \Rightarrow \langle q_1 L, L, S \rangle$ q1: изначально 1 deg заменен $\langle q_0, 1, \lambda, \lambda \rangle \Rightarrow \langle q_0 R, S, S \rangle$  $\langle q_0, 0, \lambda, \lambda \rangle \Rightarrow \langle q_0 R, S, S \rangle$  $\text{стаци}$  $\text{стаци}$  $\text{стаб.}$  $\text{Без нуты.}$  $\text{No спиральах}$  $\rangle$  ком 2 видо  
изменение

9.

2-dimensional 1 deg freedom

$$\langle q_1^1, \odot, \lambda \rangle \rightarrow \langle q_1 L, \frac{1}{1} L \rangle$$

$$\langle q_1^1, 1, \lambda \rangle \rightarrow \langle q_1 L, \angle, 0L \rangle$$

$$\langle q_1^1, 0, \odot, \lambda \rangle \rightarrow \langle q_1 L, L, 0L \rangle$$

$$\langle q_1^1, \alpha, 1, \lambda \rangle \rightarrow \langle q_2 L, L, 1L \rangle$$

$$\langle q_1^1, 1, \lambda, \lambda \rangle \rightarrow \langle q_1 L, S, 1L \rangle$$

$$\underbrace{\langle q_1^1, 0, \lambda, \lambda \rangle \rightarrow \langle q_1 L, S, 0L \rangle}$$

$$\underbrace{\langle q_1^1, \lambda, \lambda, \lambda \rangle \rightarrow \langle q_F^S, S, S \rangle}$$

ком 2 сюда  
использовать

$q_2$

$\langle q_2 1, 0, \lambda \rangle \rightarrow \langle q_1 L, L, 0L \rangle$

$\langle q_2 0, 0, \lambda \rangle \rightarrow \langle q_2 L, L, 1L \rangle$

$\langle q_2 0, 1, \lambda \rangle \rightarrow \langle q_2 L, L, 0L \rangle$

$\langle q_2 1, 1, \lambda \rangle \rightarrow \langle q_2 L, L, 1L \rangle$

$\langle q_2 1, \lambda, \lambda \rangle \rightarrow \langle q_1 L, S, 0L \rangle$

$\langle q_2 0, \lambda, \lambda \rangle \rightarrow \langle q_2 L, S, 1L \rangle$

$\langle q_2 1, \lambda, \lambda \rangle \rightarrow \langle q_F S, S \rangle$

- Selection  $n^2$
- Insertion  $n$ ,  $n^2$ ,  $n^2$
- Bubble  $n$ ,  $n^2$ ,  $n^2$

- Kegelroh  $n \log n$
- Umgabe  $n \log n$ ,  $n \log n$ ,  $n^2$
- Zeileme  $n \log n$

- Benutzbar  $n$ ,  $n \cdot n!$ ,  $\infty$

- 2) habt. screenmemory ausgleichen  
durch zugeben bestimmte Argumente
- 3) ead. bsp. abhäng. aufgrund

Виберіть середню оцінку часової складності для кожного з алгоритмів сортування

Виберіть середню оцінку часової складності для кожного з алгоритмів сортування

Випадкове сортування

Випадково

$$\begin{array}{l} n \cdot n! \\ n^2 \\ n \cdot d \\ n \log n \end{array}$$

Сортування бульбашкою

Випадково

Сортування за розрізами

Випадково

Піраміdalne сортування

Випадково

Часова складність алгоритму Швидкого сортування у середньому і в найгіршому випадку

Часова складність алгоритму **Швидкого сортування** у середньому -  $\log n$ , а в найгіршому випадку  $n^k$ .

$$n \log n$$

$$n^k$$

Сортування підрахунком є найбільш ефективним для наборів даних

Сортування підрахунком є найбільш ефективним для наборів даних -

будь-яких  
нільх чисел  
чисел з будь-якого діапазону  
чисел, що рівномірно розподілені по діапазону



Виберіть середню оцінку часової складності для кожного з алгоритмів сортування

Виберіть середню оцінку часової складності для кожного з алгоритмів сортування

Сортування вибором

$O(n^2)$

$$n^2$$

Сортування вставкою

$O(n^2)$

$$n^2$$

Швидке сортування

$O(n \log n)$

$$n \log n$$

Сортування злиттям

$O(n \log n)$

$$n \log n$$

У яких з алгоритмів сортування найгірша оцінка часової складності є  $O(n^2)$ ?

У яких з цих алгоритмів сортування найгірша оцінка часової складності є  $O(n^2)$ ?

Select one or more:

- Сортування злиттям
- Сортування вибором ✓
- Швидке сортування ✓
- Сортування вставкою ✓
- Випадкове сортування

# Оцінка часу виконання

$$O(n) \quad 1 \text{ сек} \quad 1000$$

$$O(n^2) \quad 1 \text{ сек} \quad \begin{array}{r} \sqrt{1000} \\ 1000 \end{array}$$

$$O(n \log n) \quad 1 \text{ сек} \quad \begin{array}{r} 1000 \\ \hline n \log n \end{array}$$

## Приклад задачі

### Приклад 1:

Алгоритм має складність  $O(n^2)$ . Потрібно обчислити час виконання для розміру вхідних даних  $n = 1000$ , якщо відомо, що для  $n = 100$  час виконання складає 0.1 секунди.

### Рішення:

1. Записуємо відому інформацію:

- Час виконання для  $n = 100$ :  $T(100) = 0.1$  секунди
- Складність алгоритму:  $O(n^2)$

2. Порівнюємо час виконання для двох різних

розмірів вхідних даних:

- $T(n) \propto n^2$
- Це означає, що якщо  $n$  збільшується у 10 разів (з 100 до 1000), то час виконання збільшується у  $10^2 = 100$  разів.

3. Обчислюємо новий час виконання:

- $T(1000) = T(100) \times 100 = 0.1 \times 100 = 10$  секунд

Отже, для розміру вхідних даних  $n = 1000$  час виконання буде приблизно 10 секунд.

$$O(n^2)$$

$$n = 100 \quad 0,1 \text{ с}$$

$$n_1 = 1000 \quad ?$$

$$\frac{n_1}{10} = 10$$

$$10^2 = 100$$

$$0,1 \cdot 100 = 10 \text{ с}$$

**Приклад 2:**

Алгоритм має складність  $O(n \log n)$ . Відомо, що для  $n = 500$  час виконання складає 2 секунди. Потрібно знайти час виконання для  $n = 2000$ .

$$\begin{array}{ll} n = 500 & 2 \text{ с} \\ n_1 = 2000 & ? \\ \frac{n_1}{n} = 4 & 4 \log 4 \cdot 2 \end{array}$$

# Алфавіти і властивості алгоритмів

Абстрактним алфавітом називають довільну скінченну сукупність елементів.

Слово у заданому алфавіті – це довільна скінчена впорядкована послідовність букв цього алфавіту. Довжина слова – кількість букв у слові.

На множині слів визначено дві операції: конкатенація (присиднання) та підстановка (заміна).

Конкатенацією двох слів  $A$  та  $B$  називають слово  $AB$ , отримане пристуванням слова  $B$  до слова  $A$ .

Ця операція не комутативна, проте асоціативна.

Якщо в операції заміни  $i=1$ , то підстановку називають стандартизациєю (або канонічною).

Алфавітним оператором, або алфавітним відображенням,  $\varphi$  називають відповідність між словами в одному або різних алфавітах.

Функцію називають словниковою, якщо вона перетворює слово одного алфавіту в слово іншого алфавіту.

## Способи задання алфавітних операторів

### Табличний Скінченною системою правил

виконання операцій під час розв'язування певної задачі. Система  $\mathcal{P}$  повинна мати властивості, притаманні кожному алгоритму: дискретність, ефективність, скінченність, результативність, масовість.

Якщо система  $\mathcal{P}$  з пари  $\langle \varphi, \mathcal{P} \rangle$  задоволяє всі перелічені вище властивості, крім властивості скінченості, то таку пару називають обчислювальним методом.

#### 1.4.2. Різновиди алгоритмів

Наявність чи відсутність у системі правил  $\mathcal{P}$  тієї чи іншої властивості розбиває множину всіх алгоритмів на два класи щодо конкретної властивості.

Розглянемо деякі з цих властивостей – детермінованість, самозмінність, самозастосовність, універсальність.

Алгоритм  $A = \langle \varphi, \mathcal{P} \rangle$  називають самозастосовним, якщо слово  $\mathcal{P}^{\text{cod}}$  входить в область визначення  $A$ , і несамозастосовним в іншому випадку.

4. Алгоритм називають універсальним, якщо він еквівалентний довільному наперед заданому алгоритму  $A = \langle \varphi, \mathcal{P} \rangle$ .

4. Чим відрізняються еквівалентні та рівні алгоритми?

- Еквівалентні алгоритми розв'язують одну і ту ж задачу різними способами, маючи однакові алфавітні оператори, але різні системи правил. Рівні алгоритми мають як однакові оператори, так і однакові системи правил.

## Суперпозиція алгоритмів

A виконується після B

$$C(P) = B(A(P))$$

## Розщеплення алгоритмів

Якщо результат алгоритму C = заданому слову R

То виконуємо алгоритм A

Інакше B

$$\text{Нp: } A: ab \rightarrow bba \\ \qquad \qquad \qquad bab \rightarrow bb$$

$$B: ab \rightarrow baa \\ \qquad \qquad \qquad bab \rightarrow a$$

$$C: ab \rightarrow b; bab \rightarrow \Lambda$$

$$R: \Lambda$$

$$1) ab$$

$$C(ab) = b \neq \Lambda$$

$$\Rightarrow B(ab) = baa$$

$$2) bab$$

$$C(bab) = \Lambda$$

$$\Rightarrow A(bab) = bb$$

# Iтерація

Застосовуємо до слова суперпозиції алгоритмів A (A від A від слова) доки результат алгоритму B та результат цих суперпозицій не стане рівне слову R

$$\text{Np: } A: \varphi_1 \times y^2 \varphi_2 \rightarrow \varphi_1 y^2 \times \varphi_2$$

$$B: \dot{y}^2 y^2 \times x \rightarrow \Lambda$$

$$\varphi_i \rightarrow \varphi_i$$

Виконуємо A поки не можна виконати B

$$R = \Lambda$$

$$\begin{aligned} a) \quad & \underbrace{x \times y^2}_{A} \underbrace{y^2}_{A} \rightarrow \underbrace{x y^2 \times y^2}_{A} \rightarrow \underbrace{\dot{y}^2 \times y^2 x}_{A} \rightarrow \\ & \dot{y}^2 y^2 \times x \xrightarrow{B} \Lambda = R \quad \checkmark \end{aligned}$$

## Об'єднання

$$C(P) = A(P) B(P)$$

$$\text{Np: } A(P) = x P$$

$$B(P) = P y$$

$$C(P) = x P P y$$

## 3. abgemeine re se

1. A:  $ab \rightarrow ba$ ,  $ba \rightarrow ab$

B:  $ab \rightarrow aaab$ ,  $ba \rightarrow ab$

C:  $ab \rightarrow a$ ,  $ba \rightarrow \lambda$

R =  $\lambda$

$C(ab) = a \neq \lambda$

$C(ba) = \lambda$   
 $\Rightarrow A(ba) = ab$

$B(ab) = aaab$

2.

A:  $P_1 ab P_2 \rightarrow P_1 b a P_2$

B:  $bbbaaa \rightarrow \lambda$ ,  $b \rightarrow p$

R =  $\Delta$

$p \neq bbbbba$

$P_1 ab P_2$

$C(babbab) = bbbbba$

$babbab \rightarrow \underline{bbab}bab \rightarrow bbba\underline{ab} \rightarrow bbb\underline{a}ba \rightarrow$   
 $\rightarrow bbbbbaa$

# Завдання з чеснів

Чому дорівнює результат розгалуження  $F(ba)$ ?

Нехай алгоритми  $A$ ,  $B$  і  $C$  задані такими правилами:

$A: ab \rightarrow bb; ba \rightarrow aa,$

$B: ab \rightarrow aba; ba \rightarrow bab,$

$C: ab \rightarrow a; ba \rightarrow b,$

$R = a.$

зуб с  
Іншо R<sub>1</sub> TO A  
Іншо не R<sub>1</sub> TO B

НІ R<sub>1</sub> ТОЧУ В

Чому дорівнюватиме результат розгалуження  $F(ba)$ ?

- bb
- aba
- bab ✓
- aa
- a

$$C(bb) = b \neq a \Rightarrow$$
$$B(ba) = bab$$

Вкажіть результат ітерації алгоритмів на слові babbab

Задано алгоритми  $A$ ,  $B$  і фіксоване слово  $R$  ( $P_1, P_2, P$  - довільні слова з алфавіту  $\{a, b\}$ ):

$A: P_1abP_2 \rightarrow P_2abP_1,$

$B: P_1bbbP_2 \rightarrow P_1b; P \rightarrow P,$

$R = abb.$

$$\underline{bab}bab \rightarrow \underline{bab}ab \underline{bb} \rightarrow$$
$$\rightarrow \underline{abb}ab \underline{b} \rightarrow \underline{bab}bab \rightarrow$$
$$\rightarrow bab ab b$$

Вкажіть результат ітерації алгоритмів на слові babbab.

- abbb
- ababab
- abbbbab

ітерація повторюватиметься нескінченну кількість разів ✓

Чому дорівнює результат розгалуження  $F(ab)$ ?

Нехай алгоритми  $A$ ,  $B$  і  $C$  задані такими правилами:

$A: ab \rightarrow bb; ba \rightarrow aa,$

$B: ab \rightarrow aba; ba \rightarrow bab,$

$C: ab \rightarrow a; ba \rightarrow b,$

$R = a.$

Чому дорівнюватиме результат розгалуження  $F(ab)$ ?

- a
- aba
- aa
- bb ✓
- bab

$$C(ab) = a = R \Rightarrow$$
$$A(ab) = bb$$

A:  $P_1 ab P_2 \rightarrow P_2 ab P_1$

B:  $P_1 bbb P_2 \rightarrow P_1 b$

$P_1 \rightarrow P_1$

R = bab

Iterating

$bba\overbrace{bb} \rightarrow b\underbrace{bab}_{bb} \rightarrow \underline{bab}$

Начиная с этого момента - это упражнение.

вывесение **нагружения**  $P_{r_1}, P_{r_2}, \dots, P_r$

$l \rightarrow w$  - **вывеска навеска**

$w \rightarrow l$  - **закрепление нагрузки**

• **Несимметрическое**

• **Симметрическое**

## Задачи

1) На рисунке даны зоны нагр. не симм.

Сформулируйте зонирование

2) На рисунке зонирование несимметрическое

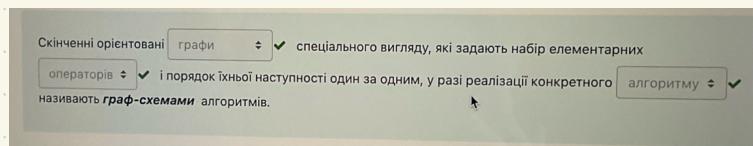
$$P_{r_1} : ab \rightarrow b$$

$$P_{r_2} : ac \rightarrow c$$

$$P_{r_3} : aa \rightarrow a$$

$bacaabaa \rightarrow ba cabaa \rightarrow \underline{ba} cabaa \rightarrow$   
 $\rightarrow bca \underline{baa} \rightarrow \boxed{bca}$

# Поняття Граф-схеми Кануки



Усі алгоритми називаються  
Деяльність якої с. може бути об.  
корисні. алгоритми.

### 2.3.2. Принцип нормалізації

Кожна алгоритмічна система повинна задовільняти дві вимоги: бути математично строгою та універсальною.

Математичної строгості досягають, використанням певного математичного апарату (у цьому випадку розпізнавання входження і підстановка). Тому побудована А. Марковим теорія нормальних алгоритмів повністю задовільняє першу вимогу.

Універсальність теорії нормальних алгоритмів формулюють у вигляді **принципу нормалізації**, який є однією з головних гіпотез (тез) теорії алгоритмів.

А. Марков сформулював і довів необхідну умову універсальності алгоритмічної системи Маркова.

**Теорема.** Для того, щоб реалізувати в схемах нормальних алгоритмів довільний алгоритм  $A = \langle \varphi, \mathcal{P} \rangle$ , необхідно, щоб у системі нормальних алгоритмів були як звичайні, так і заключні підстановки.

**Принцип нормалізації.** Для будь-якого алгоритму  $A = \langle \varphi, \mathcal{P} \rangle$  в довільному алфавіті  $X$  можна побудувати еквівалентний йому нормальний алгоритм над алфавітом  $X$ .

Поняття нормального алгоритму над алфавітом  $X$  означає таке. Інколи не вдається побудувати нормального алгоритму, сквівалентного заданому в алфавіті  $X$ , використовуючи в підстановках лише букви алфавіту  $X$ . Однак потрібний нормальний алгоритм можна побудувати, розширяючи алфавіт  $X$  додаванням певної кількості нових букв, але залишаючи область визначення алгоритму попередньою.

Перехід від інших способів опису алгоритмів до сквівалентних нормальних алгоритмів називають **зображенням у нормальній формі**, або **нормалізацією**.

Алгоритм  $A = \langle \varphi, \mathcal{P} \rangle$  в алфавіті  $X$  називають **нормалізованим**, якщо можна побудувати еквівалентний йому нормальний алгоритм над алфавітом  $X$ . В іншому випадку алгоритм називають **ненормалізованим**.

На підставі цього означення принцип нормалізації можна сформулювати так: **усі алгоритми є нормалізованими**.

Принцип нормалізації уточнює поняття алгоритму у формі нормальних алгоритмів. Він визначає відповідність між інтуїтивним поняттям алгоритму і точним математичним поняттям нормального алгоритму.

Універсальним нормальним алгоритмом (УНА) називають алгоритм, здатний виконувати роботу довільного нормального алгоритму  $A$ .

В основі універсального нормального алгоритму є такий самий метод, як і в сучасних комп'ютерах. А саме: УНА отримує інформацію про схему конкретного алгоритму  $A$ , а також про вхідне слово  $P$  і виконує над  $P$  підстановки згідно з заданою схемою.

НЕ ПОЖЛІВО  
ДОВРЕСТЬ

Що таке Універсальний нормальний алгоритм?

Що таке Універсальний нормальний алгоритм?

Select one:

- Це алгоритм, здатний виконувати роботу довільного **нормального** алгоритму. ✓
- Це будь-який нормальний алгоритм.
- Це алгоритм, здатний виконувати роботу довільного алгоритму.
- Це нормальний алгоритм, який можна зобразити граф-схемою.

[Clear my choice](#)

Нормальний алгоритм незастосовний до заданого вхідного слова, якщо на даному кроці

Нормальний алгоритм незастосовний до заданого вхідного слова, якщо на даному кроці

Select one:

- Зрозуміло, що процес підстановок не зможе зупинитися ✓
- жодна підстановка не підходить - заснов.
- Застосована остання підстановка зі списку підстановок, що задають даний алгоритм X
- Застосована заключна підстановка - заснов.

Які два види продукції можливі в нормальніх алгоритмах Маркова?

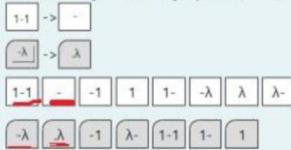
Які два види продукції можливі в нормальніх алгоритмах Маркова?

Select one or more:

- початкова продукція X
- проста продукція ✓
- заключна продукція ✓
- складна продукція X

Побудувати нормальній алгоритм Маркова, який реалізує віднімання А-В, де значення є натуральними числами, поданими як ланцюжок символів '1'

Побудувати нормальній алгоритм Маркова, який реалізує віднімання  $A - B$ , де значення  $A$  і  $B$  є натуральними числами, поданими як ланцюжки символів '1' (у так званій унарній системі).



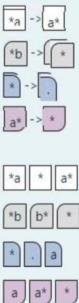
$$\underline{111 - 11} = 1\underline{1} - \underline{1} = \underline{1}$$

1 - 1 → -

-1 → +1

Побудуйте нормальній алгоритм, який видає всі входження, крім першого, символа 'a' з заданого рядка в алфавіті  $M = \{a, b\}$ .

Побудуйте нормальній алгоритм, який видає всі входження, крім першого, символа 'а' з заданого рядка в алфавіті  $M = \{a, b\}$ .



babaaqabca

\* $\alpha$  $\rightarrow$ \*

\*6 → 6\*

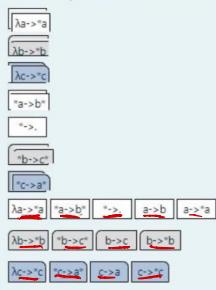
\* ۹.

$$\alpha \rightarrow \alpha^*$$

Скласти нормальній алгоритм, що в рядку символів з алфавіту {a, b, c}

здійснює таку заміну

Склади нормальній алгоритм, що в рядку символів з алфавіту {a,b,c} здійснює таку заміну: всі символи 'a' міняють на символ 'b', всі символи 'b' міняють на символ 'c', а всі символи 'c' міняють на 'a'. Наприклад: abbcacaacbba=>bccbabacbb



\*a → b\*

\*b → c\*

\*C → a\*

x ↗

10 → \*Q

16 → \*6

$$\lambda c \rightarrow *c$$

Побудуйте нормальній алгоритм, який видає всі входження, крім останнього, символа 'a' з заданого рядка в алфавіті  $M=\{a,b,c\}$

Побудуйте нормальній алгоритм, який видає всі входження, крім останнього, символа 'a' з заданого рядка в алфавіті  $M=\{a,b,c\}$ .

$\boxed{ab} \rightarrow b$

$\boxed{ac} \rightarrow c$

$\boxed{aa} \rightarrow a$

'bacabaabaa' => 'bacabaa' => 'bacbaa' => 'bcbaa' => 'bcba'

$a b \rightarrow b$

$a c \rightarrow c$

$a a \rightarrow a$

Побудувати нормальній алгоритм Маркова, який реалізовує додавання  $A+B$

Побудувати нормальній алгоритм Маркова, який реалізує додавання  $A+B$ , де значення  $A$  і  $B$  є натуральними числами, подані ланцюжки символів '1' (у так званій унарній системі).

$\boxed{+1} \rightarrow \boxed{1+}$

$\boxed{1+} \rightarrow \boxed{1}$

$\boxed{+1} \quad \boxed{+} \quad \boxed{\lambda}$

$\boxed{1+} \quad \boxed{1} \quad \boxed{1+1}$

Please put an answer in each box.

$$\begin{array}{r} +1 \rightarrow 1+ \\ 1+ \rightarrow 1 \end{array}$$

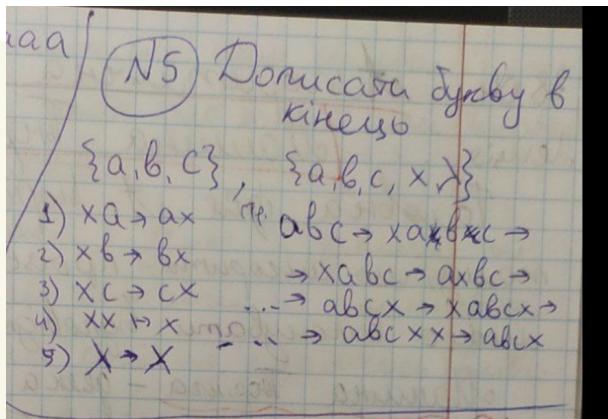
Склади нормальній алгоритм, що виконує збільшення унарного числа на 1

Склади нормальній алгоритм, що виконує збільшення унарного числа на 1.

$\boxed{1} \rightarrow \boxed{.11}$

$\boxed{11} \quad \boxed{111} \quad \boxed{.1}$

$$\begin{array}{l} 111 \rightarrow 111 \\ 11 \rightarrow 11 \\ \text{або} \qquad 1 \rightarrow .11 \end{array}$$



Яким буде результат застосування нормального алгоритму в алфавіті  $A = \{a, b, c\}$

Яким буде результат застосування нормального алгоритму в алфавіті  $A = \{a, b, c\}$ :

$cb \rightarrow abc$

1.  $bca bacab$

$bca \underline{bac} ab \rightarrow bca ac \underline{ab} \rightarrow bca a b$

$bac \rightarrow ac$

2.  $bca a \underline{c} ab$

$cab \rightarrow b$

3.  $bca a \underline{ab}$

до слова  $R = bcabacab$

Answer: **bcaab**



Яким буде результат застосування нормального алгоритму до слова  $R = bbaabab$

Яким буде результат застосування нормального алгоритму

$ab \rightarrow \lambda$

$bba \underline{ab} ab \rightarrow bba \lambda \underline{ab} \rightarrow bba \underline{\lambda} \lambda \rightarrow b \underline{a} \lambda \lambda \rightarrow b \lambda \lambda \rightarrow b$

$ba \rightarrow ab$

до слова  $R = bbaabab$

Відповідь:

**b**



як  
бум, ком  
 $\lambda$  в кінці



Що буде результатом застосування нормального алгоритма до слова  $R = abbc$

Що буде результатом застосування нормального алгоритма

$ab \rightarrow bd$

$db \rightarrow ba$

$bba \rightarrow abb$

$c \rightarrow \lambda$

до слова  $R = abbc$

$abbc \rightarrow bd \underline{bc} \rightarrow b \underline{ba} c \rightarrow abbc \rightarrow \dots$

Виберіть одну відповідь:

Алгоритм не може бути застосовним  
до цього слова

bb

aa

cc



Яким буде результат застосування нормального алгоритму в алфавіті  $A = \{a, b, c\}$

Яким буде результат застосування нормального алгоритму в алфавіті  $A = \{a, b, c\}$ :

$abc \rightarrow c$

1.  $bacaa$



$bacaa \underline{abc} \rightarrow baca \underline{c} \rightarrow cb \underline{cac} \rightarrow$

$ba \rightarrow cb$

2.  $bacaa$



$baca \underline{ab} c \rightarrow cb \underline{ab} c \rightarrow cbc$

$ca \rightarrow ab$

3.  $bacaa$



$baca \underline{ab} c \rightarrow cb \underline{ab} c \rightarrow cbc$

до слова  $R = bacaaabc$

**cbc**

# Найпростіші функції

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

$$O^n(x_1, \dots, x_n) = 0$$

$$I^n_i(x_1, \dots, x_n) = x_i$$

Дп:

$$S^1(5) = 6$$

$$O^4(3, 6, 2, 1) = 0$$

$$I^3_2(4, 3, 8) = 3$$

- пр. безпередовість

- куль пр.

- пр. бедору функції

Які з наступних характеристик притаманні найпростішим рекурсивним функціям?

Select one or more:

- Найпростіші функції визначені на множині цілих чисел
- Найпростіші функції є всюди визначені ✓
- Найпростіші функції можуть бути визнечені через інші найпростіші функції X
- Є три найпростіші функції ✓
- Є дві найпростіші функції X
- Найпростіші функції - це числові функції

$$S^3(I^2_2, I^3_1, I^3_2) = x_2$$

1) Оператори суперпозиції

$$I^2_2(x_1, x_2)$$

$$S^{(n+1)}(f, f_1, \dots, f_n) = f(f_1, \dots, f_n)$$

Дп: 1)  $f(x) = x^2 + 3$        $S^2 - ?$

$$g(x) = 5 - x$$

$$S^2(f, g) = f(g(x)) = (5 - x)^2 + 3$$

$$S^2(g, f) = g(f(x)) = 5 - (x^2 + 3)$$

$$S^2(f, f) = f(f(x)) = (x^2 + 3)^2 + 3$$

$$S^2(g, g) = g(g(x)) = 5 - (5 - x)$$

$$(x_1 + x_2)^2 = S^3(+, I_1^2, I_2^2)$$

$$x_1 \cdot x_2 + x_3 = S^3(+, I_3^3, S^3(\cdot, I_1^3, I_2^3))$$

## 2) Определение производной

$$g^{(n)} : N^{(n)} \rightarrow N$$

$$h^{(n+2)} : N^{(n+2)} \rightarrow N$$

$$f^{(n+1)} : N^{(n+1)} \rightarrow N$$

$$\{ f(x_1 \dots x_n, 0) = g(x_1 \dots x_n)$$

$$\{ f(x_1 \dots x_n, y+1) = h(x_1 \dots x_n, y, f(x_1 \dots x_n, y))$$

$$f = R(g, h)$$

2

Елементарні - deg R

Приємні - нес

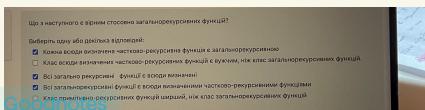
Задовільно - нес

$$\{ S^1, 0^n, I_n \} \xrightarrow{S^{n+1}, R} f$$

$$\{ S^1, 0^n, I_n, \tilde{O}_j \} \xrightarrow{S^{n+1}, P, M} f$$

$$\xrightarrow{S^{n+1}, R, M}$$

задовільно - нес



$$1) f(x, y) = x + y$$

$$f(x, 0) = x = I_1^1(x) \quad g(x) = I_1^1(x)$$

$$f(x, y+1) = x + y + 1 = f(x, y) + 1 = S^1(f(x, y))$$

$$h(x, y, z) = S^1(z) = S^1(I_3^3(x, y, z))$$

$$f(x, y) = R(I_1^1(x), S^1(I_3^3(x, y, z)))$$

$$2) f(x, y) = x \cdot y$$

$$f(x, 0) = 0 = O^1 \quad g(x) = O^1(x)$$

$$f(x, y+1) = x(y+1) = xy + x = f(x, y) + x =$$

$$= S^3(+, f(x, y), x)$$

$$h(x, y, z) = S^3(+, f(x, y), x) = S^3(R(I_1^1(x), S^1(I_3^3(x, y, z))), z, x) = S^3(R(I_1^1(x), S^1(I_3^3(x, y, z))), I_3^3(x, y, z), I_3^1(x, y, z))$$

$$f(x) = R(O^1(x), S^3(R(I_1^1(x), S^1(I_3^3(x, y, z))), I_3^3(x, y, z), I_3^1(x, y, z)))$$

3.) Доведіть членів змінної:

$$M_2(y+z=x) = x - y$$

$M^l f = \begin{cases} Mf, & \text{якщо } Mf \text{ всюди визначена;} \\ \text{не визначена, якщо } Mf \text{ визначена не всюди.} \end{cases}$

Методи членів змінної

Доведувати

Творінг, Knini, Гедель, Черч.

Теза Черча

- **не може бути  
доведена**

Що собою представляє наступне твердження?

"Клас алгоритмічно (або машинно) обчислюваних часткових числових функцій збігається з класом усіх частково-рекурсивних функцій"

Виберіть одну або декілька відповідей:

Теза Творінга

Що собою представляє наступне твердження?

"Клас функцій, алгоритмічно обчислюваних відносно деякого класу функцій  $\alpha$ , збігається з класом частково-рекурсивних функцій відносно  $\alpha$ ".

Виберіть одну або декілька відповідей:

- Це твердження доведено
- Це теза Творінга
- Це теза Черча
- Це твердження не може бути доведене

**не може бути  
доведена**

# Навін. оп. питання

Запишіть результат виконання найпростіших функцій

Запишіть результати виконання найпростіших функцій:

$$s^1(3) = \boxed{3+1=4} \quad 3+1=4$$



$$\sigma^5(1,2,3,4,5) = \boxed{0} \quad 0$$

$$I_2^3(1,2,3) = \boxed{2} \quad 2$$

Запишіть результати виконання операторів над такими найпростішими функціями:

$$I_2^2(x,y,z), I_1^3(x,y,z), I_2^3(x,y,z), \sigma^1(x)$$

$$S^3(I_2^2, I_1^3, I_2^3) = \boxed{y}$$

$$S^2(\sigma^1, S^3(I_2^2, I_1^3, I_2^3)) = \boxed{0}$$

Запишіть результати виконання операторів над такими функціями

Запишіть результати виконання операторів над такими функціями:

$$h(x)=0, f(x)=x+2.$$

$$S^2(f,h) = \boxed{2} \quad \circ f(h(x)) = 2$$



$$S^2(f, S^2(f,h)) = \boxed{4} \quad \circ f(f(h(x))) = 2+2=4$$

$$S^2(f, S^2(h,h)) = \boxed{2} \quad \circ f(h(h(x))) = 0+2=2$$

Функції, які отримують з функцій системи  $\sigma$  і найпростіших функцій із застосуванням скінченної кількості операторів суперпозиції, примітивної рекурсії та слабкої мінімізації, називають \_\_\_\_\_ відносно системи  $\sigma$ .

Часткові функції, які отримують з функцій системи  $\sigma$  і найпростіших функцій із застосуванням скінченної кількості операторів суперпозиції, примітивної рекурсії та слабкої мінімізації, називають \_\_\_\_\_ відносно системи  $\sigma$ .

Select one:

- елементарними
- рекурсивними
- загальнорекурсивними
- частково рекурсивними



З допомогою якого оператора утворена наступна часткова функція? ( $\mu$ )

З допомогою якого оператора утворена наступна часткова функція?

$$\mu_y(f(x_1, \dots, x_{n-1}, y) = x_n)$$



Select one:

- Оператора слабкої мінімізації
- Оператора мінімізації
- Оператора суперпозиції
- Оператора примітивної рекурсії

З допомогою якого оператора утворена наступна часткова функція? (всюди визначена)

З допомогою якого оператора утворена наступна часткова функція?

$$M^l f = \begin{cases} Mf, & \text{якщо } Mf \text{ всюди визначена;} \\ \text{не визначена,} & \text{якщо } Mf \text{ визначена не всюди.} \end{cases}$$

Select one:

- Оператора суперпозиції
- Оператора примітивної рекурсії
- Оператора слабкої мінімізації
- Оператора мінімізації



Які з наступних характеристик притаманні найпростішим рекурсивним функціям?

Які з наступних характеристик притаманні найпростішим рекурсивним функціям?

Select one or more:

- Найпростіші функції визначені на множині цілих чисел
- Найпростіші функції є всюди визначені ✓
- Найпростіші функції можуть бути визначені через інші найпростіші функції ✗
- Є три найпростіші функції ✓
- Є дві найпростіші функції ✗
- Найпростіші функції - це числові функції



# Теза Тьюрінга

**Теза Тьюрінга.** Для довільного алгоритму  $A = \langle \varphi, \mathcal{D} \rangle$  у довільному скінченному алфавіті  $X$  існує функція  $\varphi(P): \{P\}_x \rightarrow \{Q\}_x$ , обчислювана за Тьюрінгом.

Теза Тьюрінга визначає відповідність між інтуїтивним поняттям алгоритму і точним математичним поняттям функції, обчислюваною на МТ: будь-який алгоритм, заданий у довільній формі, можна замінити еквівалентною йому МТ.

Згідно з цією тезою, питання про можливість алгоритмізації того чи іншого процесу рівносильне питанню про можливість реалізації цього процесу на МТ.

Як і принцип нормалізації, теза Тьюрінга стверджує неможливість побудови в майбутньому алгоритму, який не можна було б реалізувати на МТ.

Тезу Тьюрінга, як і тези Маркова і Черча, довести неможливо, оскільки в її формулюванні використане інтуїтивне поняття алгоритму.

## Різновиди

- Багатострічкова
- Багатоголовкова
- Машина з багатоповерховою стрічкою
- Одноголовкова з однією двовимірною стрічкою
- Одностороння
- З декількома функціональними механізмами

Множина  $S$  - **програма**

МТ

Тоді **формально машиною Тьюрінга** називають сімку

$$(S, Q, q_0, q_F, I, \Lambda, \delta), \quad (2.12)$$

де  $I \subseteq S$  - множина вхідних символів;  $\Lambda$  - порожній символ.

• Головка

• Тип. стрічка

• Пристрій керування

Застосування, яке  
звершуємо зав.  
кодування F.

1. Функція наступності  $s^1$  реалізується машиною Тьюрінга із зовнішнім алфавітом  $M = \{\Lambda, 1\}$  та внутрішніми станами  $\{q_0, q_1\}$  і програмою

	$\Lambda$	1
$q_0$	R	$1 q_1 R$
$q_1$	$1 q_F$	R

2. Нуль-функція  $o^1$  реалізується машиною Тьюрінга із зовнішнім алфавітом  $M = \{\Lambda, 1\}$  та внутрішніми станами  $\{q_0, q_1\}$  і програмою

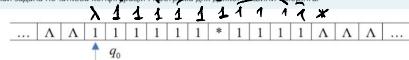
	$\Lambda$	1
$q_0$	R	$q_1 R$
$q_1$	$q_F$	$\Lambda R$

3. Функція вибору аргументу. Побудуємо програму для машини Тьюрінга із зовнішнім алфавітом  $M = \{\Lambda, 1\}$  та внутрішніми станами  $\{q_0, q_1\}$  і програмою

	$\Lambda$	1
$q_0$	R	$q_1 R$
$q_1$	$\Lambda q_F H$	$1R$

Нехай задано початкова конфігурація і програма для лякої машини Тьюрінга.  
Яким результатом буде застосування цієї машини до зображеного на стрічці слова?

Нехай задана початкова конфігурація та програма для лякої машини Тьюрінга:



Яким результатом буде застосування цієї машини до зображеного на стрічці слова?

Select one:

- 1111111111\*
- 1111111111
- 1111111111\*



Чи можуть дві різні команди машини Тьюрінга мати однакові ліві частини?

Чи можуть дві різні команди машини Тьюрінга мати однакові ліві частини?

Виберіть одну відповідь:

- так, але тільки для однострічкових машин
- так, ще можливо
- ні, це неможливо



Будь-який алгоритм, заданий у довільній формі, можна замінити еквівалентною йому МТ

Виберіть правильні твердження стосовно наступного формулювання:

Будь-який алгоритм, заданий у довільній формі, можна замінити еквівалентною йому Машиною Тьюрінга.

Select one or more:

- це твердження неможливо довести
- це твердження встановлене відродженні між інтуїтивним поняттям алгоритму і точним математичним поняттям функції, обчислюваної на МТ
- це теорему довів Тьюрінг
- це теза Тьюрінга
- це теза Черка



Внутрішнім алфавітом машини Тьюрінга називають

Внутрішнім алфавітом машини Тьюрінга називають

Select one:

- символи, які можуть бути записані на стрічці
- множина станів машини
- множина команд машини
- символи, які допускає задача

[Clear my choice](#)



## Що є різновидом машини Тьюрінга?

Що з наступного є різновидом машини Тьюрінга?

Select one or more:

- багатостанова машина Тьюрінга
- багатоголовкова машина Тьюрінга
- багатострічкова машина Тьюрінга
- Машина Тьюрінга з багатофункціональним механізмом
- машина Тьюрінга з багатоповерховою стрічкою



Чи може заключний стан зустрічатися в правій частині команди машини Тьюрінга?

Чи може заключний стан зустрічатися в правій частині команди машини Тьюрінга?

Select one:

- так, це можливо
- ні, це неможливо
- так, але тільки для однострічкових машин



Що реалізує машина Тьюрінга?

Що реалізує така машина Тьюрінга?

	A	I
$q_0$	R	$I q_1 R$
$q_1$	$I q_F$	R



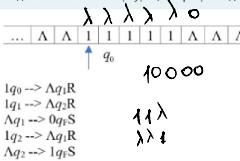
$\lambda \ 1 \ 1 \ 1 \ \lambda$   
 $\lambda \ 1 \cdot \cdot \ - \ 1$

Select one:

- функцію додавання двох чисел в унарній системі числення
- нуль-функцію
- функцію вибору аргументу
- функцію наступності

Задана початкова конфігурація і програма для деякої машини Тьюрінга.  
Вказати вірні твердження стосовно даної машини.

Нехай задана початкова конфігурація і програма для деякої машини Тьюрінга:



Вкажіть які з наступних тверджень є вірні стосовно даної машини.

Select one or more:

- дана програма замінює символи 1 на 0 і символи 0 на 1

- дана програма залишить певну послідовність символів 0 або 1 після її застосування до будь-якого слова в алфавіті {0,1}
- дана програма залишить один символ 0 або 1 після її застосування до будь-якого слова в алфавіті {0,1}
- дана програма визначає парність числа, записаного в унарній системі числення
- дана програма залишиться

My Career | SoftSe... Report Manager MCSO

Що реалізує така машина Тьюрінга?

	a	b	$\Lambda$
$q_0$	R	R	$q_1 \text{ a } R$
$q_1$	$q_0$	$q_0$	$q_2 \text{ b } R$
$q_2$			$q_3 \text{ a } R$
$q_3$			$q_F$

$\lambda a b b \lambda$   
 $\uparrow \uparrow \uparrow \uparrow \uparrow \uparrow$   
 $a b a$

?

Pitannya 10 Версія 7 (остання)

Питання 1 Відповіді ще не було  
Макс. оцінка до 1,00

Що реалізує така машина Тьюрінга?

	1	*	$\Lambda$
$q_0$	$1 q_1 L$	-	$\Lambda q_1 R$
$q_1$	$\Lambda q_1 R$	$\Lambda q_2 R$	-
$q_2$	$1 q_2 R$	$\Lambda q_3 R$	-
$q_3$	$\Lambda q_3 R$	$\Lambda q_4 R$	-
$q_4$	$\Lambda q_4 R$	$\Lambda q_5 R$	-
$q_5$	$\Lambda q_5 R$	$\uparrow$	$q_F$

$h(x_1 \cdot x_2 \cdot x_3 \cdot x_4) = x_4$

$$h(x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5) = x_2$$

Як визначається ємнісна складність машини Тьюрінга?

Виберіть одну відповідь:

- дорівнює найбільшій кількості кроків, зроблених МТ під час опрацювання входу довжиною  $n$
- дорівнює середній відстані, яку потрібно пройти головці під час опрацювання входу довжини  $n$ .
- дорівнює відстані, яку потрібно пройти головці під час опрацювання входу середньої довжини.
- дорівнює максимальній відстані, яку потрібно пройти головці під час опрацювання входу довжини  $n$ .

Очистити мій вибір

Як визначається часова складність машини Тьюрінга?

Виберіть одну відповідь:

- дорівнює кількості станів, використаних у програмі МТ **×**
- дорівнює найбільшій кількості кроків, зроблених МТ під час опрацювання входу довжиною  $n$
- дорівнює середній кількості кроків, зроблених МТ під час опрацювання входу довжиною  $n$
- дорівнює кількості кроків, зроблених МТ під час опрацювання входу середньої довжини

Ваша відповідь правильна

Правильна відповідь: дорівнює найбільшій кількості кроків, зроблених МТ під час опрацювання входу довжиною  $n$

## Версія 1 (остання)

Машина Тьюрінга – це математична модель пристрою, який породжує обчислювальні процеси. Її використовують для теоретичного уточнення поняття алгоритму та його дослідження.

# Машинна Поста

Для машини Поста існує шість типів команд:

**V<sub>j</sub>** – поставити мітку та перейти до  $j$ -го рядка програми;

**X<sub>j</sub>** – стерти мітку та перейти до  $j$ -го рядка програми;

**→<sub>j</sub>** – переміститися вправо та перейти до  $j$ -го рядка програми;

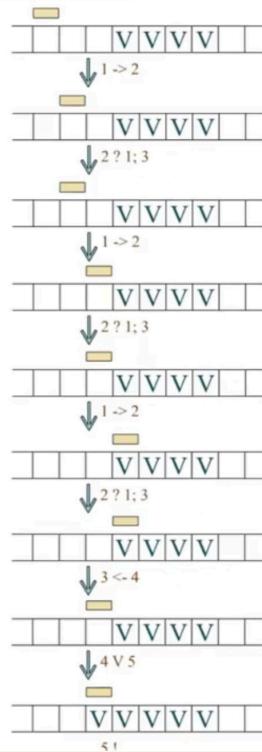
**←<sub>j</sub>** – переміститися вліво та перейти до  $j$ -го рядка програми;

**?<sub>j<sub>1</sub>; j<sub>2</sub></sub>** – якщо в комірці немає мітки, то перейти до  $j_1$ -го рядка програми, інакше – перейти до  $j_2$ -го рядка програми

**!** – завершення програми (стоп)

## Приклад

1. → 2
2. ? 1; 3
3. ← 4
4. V 5
5. !



Що виконує наступна програма для машини Поста, якщо на стрічці задана послідовність відміченіх комірок і початково каретка знаходиться над однією з відмічених комірок?

- 1. ? 2, 3
- 2. V 4
- 3. → 1
- 4. → 5
- 5. V 6
- 6. !



Select one:

- програма виконає недопустиму команду і зупиниться
- подвоєє кількість відмічених комірок
- збільшує кількість відмічених комірок на 2
- програма зациклиться
- стирає дві відмічені комірки зправа

### Що означає кожна з наведених команд машини Поста?

Що означає кожна з наведених команд машини Поста?

- i змістити активну комірку на одну позицію вправо і перейти до виконання i-ї команди
- ← i змістити активну комірку на одну позицію вліво і перейти до виконання i-ї команди
- ! зупинка, закінчення роботи алгоритму

вправо

вліво

### Що виконає програма для машини Поста?

Що виконує наступна програма для машини Поста, якщо на стрічці задана послідовність відміченіх комірок і початково каретка знаходиться над однією з відмічених комірок?

- 1. ? 2, 3
- 2. ε 4
- 3. → 1
- 4. !

11!  
↑↑↑

1111  
↑↑↑↑↑↑



Select one:

- програма виконає недопустиму команду і зупиниться
- подвоєє кількість відмічених комірок
- збільшує кількість відмічених комірок на 1
- програма зациклиться
- стирає крайню праву відмічену комірку

[Clear my choice](#)

## Система Поста еквівалентна

Система Поста еквівалентна ...

Select one or more:

- системі Тюрінга
- системі Гаяса
- системі Черча-Кліні
- системі Маркова
- жодній системі

1. ? 2,3

2.  $\rightarrow$  3

3.  $\rightarrow$  1

4.  $\rightarrow$  5

5.  $\vee$  6

6. !

1 1 1 1 1

↑

1 1  
↑ ↑ ↑ ↑ ↑ ↑ !

Задача №1

У якому з наступних випадків відбудеться зупинка машини Поста?

Виберіть одну або декілька відповідей:

- Коли машина зациклиться
- У ході виконання програми машина пройде всі відмічені комірки ✗
- У ході виконання програми машина дійде до виконання невиконуваної команди
- У ході виконання програми машина дійде до виконання команди ! ✓

Ваша відповідь правильна

Правильні відповіді: У ході виконання програми машина дійде до виконання невиконуваної команди, У ході виконання програми машина дійде до виконання команди !

Виберіть одну або декілька відповідей:

- На стрічку машини Поста записують інформацію в двійковому алфавіті
- Кожному кроku відповідає на інформаційній стрічці активна комірка. ✓
- Машина Поста має скінченну інформаційну стрічку
- На стрічку машини Поста записують інформацію в довільному алфавіті
- Машина Поста має нескінченну інформаційну стрічку ✓
- Кожному кроku машини відповідає один з визначених станів.

Ваша відповідь частково правильна.

У вас правильних відповідей: 2.

Правильні відповіді: Машина Поста має нескінченну інформаційну стрічку, На стрічку машини Поста записують інформацію в двійковому алфавіті, Кожному кроku відповідає на інформаційній стрічці активна комірка.

Теза Поста: Для кожного алгоритму  $A = \langle \phi, P \rangle$  існує алгоритм Поста  $\phi$ , що реалізує словникову функцію  $\phi$ .



Конкурс

G університет...

G Nazarii14 (N...

BogoSort or...

G спортзал на...

x Практическ...

## Скачать материал

\* Важно! Нужно проверить, что массив состоит не менее чем из трех меток, иначе придется правее трех и снова решать ту же задачу. Если правее очередных трех меток окажется пробел, то за ним поставить еще одну метку.

- |                    |                    |
|--------------------|--------------------|
| 1. $\rightarrow$ 2 | 6. $\rightarrow$ 7 |
| 2. ? 3; 4          | 7. ? 8; 1          |
| 3. !               | 8. $\rightarrow$ 9 |
| 4. $\rightarrow$ 5 | 9. V 3             |
| 5. ? 3; 6          |                    |

### 3. Ориентация на ленте

12 На ленте имеется некоторое множество меток (общее количество меток не менее 1). Между метками множества могут быть пропуски, длина которых составляет одну ячейку. Заполнить все пропуски метками.

Решение.

- |                    |                   |
|--------------------|-------------------|
| 1. $\rightarrow$ 2 | 5. !              |
| 2. ? 3; 1          | 6. $\leftarrow$ 7 |
| 3. $\rightarrow$ 4 | 7. V 1            |
| 4. ? 5; 6          |                   |

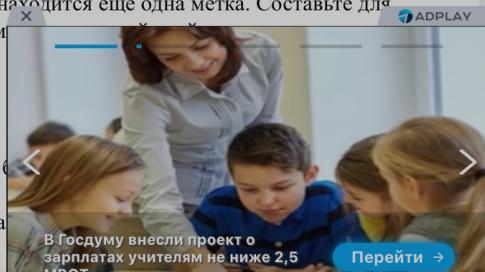
XX 1 X 111 X  
| - |

13. На ленте имеется массив из  $n$  отмеченных ячеек. Каретка обозревает крайнюю левую метку.

Справа от данного массива на расстоянии в  $m$  ячеек находится еще одна метка. Составьте для машины Поста программу, придвигающую данную машину.

Решение.

1. X 2 (удаляем левую метку массива)
2.  $\rightarrow$  3
3. ? 4; 2 (передвигаем каретку к концу массива)
4. V 5 (ставим справа от массива метку, ранее нами)
5.  $\rightarrow$  6
6. ? 7; 10 (проверяем, передвинули ли мы уже наш массив)
7.  $\leftarrow$  8
8. ? 9; 7 (заново проверяем)



Мой доход



Фильтр



Поиск курсов



Войти



Готово

Qv пропуск

1 із 2

