Байкенич. І-А. В.
ПМі-23

Білет №2

① Абстрактні алфавіти й алфавітні оператори.
Кодузальні алфавітні оператори.

– Абстрактним алфавітом наз. довільну скінченну сукупність елементів.

Природа елементів, які наз. буквами алфавіту, може бути довільною, проте вони повинні бути попарно різними, а їхня к-ть – скінченною

• Слово у заданому алфавіті – це довільна скінч. впорядкована послідовн. букв цього алфавіту. Довжина слова – к-ть букв у слові.

Алфавіти можна розширювати, додаючи до них нові букви. Слова можна впорядковувати у лексикографічному порядку, що визнач. порядком літер в алфавіті.

Конкатенацією двох слів P_1 та P_2 наз. слово P_1P_2, отримане приписуванням слова P_2 до P_1. Ця операція комутативна, проте, асоціативна

Приклад: Конкатенація:

$$P_1 = abc \;,\; P_2 = xy \Rightarrow P_1P_2 = abcxy$$

• Слово P наз. підсловом слова B, якщо B можна записати

$$B = CPD \quad (1)$$

– де C і D – деякі слова (можливо порожні) – P є вхідженням в B

• Якщо в розкладі (1) підслово C має мінімальну довжину, то таке вхідження слова P в слово B наз. першим вхідженням

Нех. задано слова P, B, F і P є і-тим вхідж. в B: B=CPD
Тоді слово G отрим. операцією заміни (підстановки) і-го вхідження слова P словом F, якщо G = CFD.

якщо в операції замінн $i = 1$, то підстановку наз. сінгулярною (або канонічною).

Алфавітним оператором, або алфавітним відображенням φ наз. відповідність між словами в одному або різних алфавітах

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

• Нех. $\{P\}_X$ - деяка множина вхідних слів в алфавіті X, множ. вихідних слів у алфавіті Y. Тоді алфавітним оператором φ є словникова ф-ція такого вигляду:

$$\varphi: \{P\}_X \to \{Q\}_Y$$

якщо слово P оператор φ не ставить у відповідність жодного вихідного слова, то кажуть, що на слові P оператор φ не визначений

Сукупність усіх слів на яких оператор φ невизнач. наз. областю визначення оператора φ

Сукупність усіх слів, на яких оператор φ невизнач. наз. областю заборони φ.

• Однозначний оператор кожному вхідному слову ставить у відповід. не більше 1го вхідн. слова, а багатозначний оператор - декілька різних слів

$$P: \begin{cases} Q_1 \\ \vdots \\ Q_n \end{cases}$$

Вибір вихідного слова відбув. або випадково, або відповідно до імовірностей, приписаних словам $Q_1, Q_2, \dots, Q_n$ (імовірнісний оператор)

Зазнач., що розподіл імовірностей між $Q_1, Q_2, \dots, Q_n$ зовн. неможливо визначити в процесі опрацювання інформації.

Кодувальні алфавітні оператори

• Нех. L - деякий алфавіт, який наз. стандартним, M - довільний алфавіт. Нех. $\{P\}_M$, $\{Q\}_L$ множ. слів у відповідних алфавітах. Кажуть, що множ. $\{P\}_M$ закодоване в алфавіті L, якщо задано такий однозн. оператор.

$$\varphi: \{P\}_M \to \{Q\}_L$$

• Оператор φ наз. кодувальним, а слова з $\{Q\}_L$ кодами об'єктів з $\{P\}_M$

Теорема. Кодувальний оператор є взаємнооднозначним тоді і тільки тоді, коли:

1) коди різних букв алфавіту M - різні.

2) код довільної букви алфавіту M не може бути початковим відрізком у коді ін. букв цього алфавіту.

Кодування об'єктів алфавіту M словами однакової довжини наз. нормальним кодуванням

Кодування слів у багатобуквеному алфавіті. Нех M - довільний алфавіт з n букв, L - стандартний з m букв ($m > 1$). Для довільної пари (m, n) завжди можна знайти таке число l, що $m^l \geq n$.

Проте відомо, що довжина l дорівн. к-ть різних слів довж. l в m-буквеному алфавіті. Отже, ост. нерівність засвідчує, що всі букви M завжди можна закод. словами довж l в алфавіті L так, щоб коди різних букв були різними. Кожне таке кодування є нормальним і забезпечує взаємну однозначність оператора кодування φ.

Приклад: якщо $n = 16$, $m = 2$ (н-д: $\{0, 1\}$), то $l = 4$, $2^4 \geq 16$ і кожну з 16 букв можна закодувати словами довж. 4:

0000, 0001, ..., 1111.

Вказані коди використ. для кодув. цифр 16-кової сист. в комп'ютері

(2) Важкорозв'язні задачі

Алгоритм можна розглядати як "чорну скриньку", яка за вхідною послідовністю будує вихідну.

Вхідна послідовність → Алгоритм → Вихідна послідовність

Зокрема, можна вважати, що вхідна і вихідна послідовності склад. з 0 і 1, які кодують вхід і вихід алгоритму. Тоді алгоритм розглядають як послідовність елементарних двійкових операцій, таких, як або, і, не, друге, і т.д. що працюють з пам'яттю двійкових символів, яка може бути досить великою.

Звичайно, можливі різні розрах. дані, однак на практиці природні способи розрах є еквівалентними, оскільки вони можуть бути поліноміально перетвор. один в інший. У цьому разі різницю між алгоритмами з поліноміальним і експоненц. часом роботи незалежно. оск. всі наші типи даних містять скін. к-ть двійк. символів, то перевед. алгоритмів з мови високого рівня на рівень двійкових символів збільш к-ть операцій лише поліноміально.

Якщо поліноміальний алгоритм потребує не більше $T(n)$ операцій високого рівня на вході розм. n, то він потребує не більше $P(T(n))$ операцій нар двійковими зображ. такої ж вхідної послідовності. Тут P - поліноміальна ф-ція, яка визнач. розм. к-ті операцій у разі переходу від операцій високого рівня до двійкових операцій. Отже, поліноміальність чи експоненціальність часу роботи алгоритму інваріантне, оск. $P(T(n))$ обмеж. зверху деяким поліномом відносно n.

Якщо вхідні та вихідні послідовності кодовані "розумним" способом (код не більший $O(\log_2(n))$, то поліноміальність чи експоненціальність довж. вхідної послідовності також інваріантне.

Нагадаємо, що алгоритм з поліноміальним часом - це алгоритм час роботи якого обмежений зверху деякою поліномом $P_k(n)$ (k - степінь полінома). Якщо степінь полінома точної оцінки, то вона залежить від реалізації.

• Якщо оцінка алгоритму зростає швидше ніж поліном, то такі алгоритми наз. експоненціальними.

Більшість задач, цікавих з практичного погляду мають поліноміальні алгоритми розв'язування. Такі задачі наз. нерозв'язними. Задачу наз. нерозв'язною, якщо не існує поліноміальних алгоритмів для її розв'язання. Для деяких експонент. алгоритмів важливо знати коли "погано" вони працюють

Поняття поліноміально розв'язної задачі грубою вважати уточненим ідеї "практично розв'язної" задачі. Це пов'язано з:

-1) використ. на практиці поліноміальні алгоритми зазвичай справді працюють досить швидко.

-2) обсяг вхідних даних практично не залежить від вибору конкретної моделі обчислень.

-3) клас поліноміально розв'язних задач має природні властивості замкненості

Наприклад: композиція двох поліноміальних алгоритмів теж працює за поліноміальний час.

(3) На машині Тюринга скласти програму для інверсії двійкового числа (замінити всі нулі на 1 і всі 1 на 0)

... ЗЗ 00101010 ЗЗ ... ⇒ ... ЗЗ 11010101 ЗЗ ...

	З	0	1
q_0	З q_0 R	1 q_1 R	0 q_1 R
q_1	З q_F H	1 q_1 R	0 q_1 R

Приклад:

ЗЗ 00101 ЗЗ (q_0) ⇒ ЗЗ 00101 З З (q_0) ⇒

⇒ ЗЗ 0 0101 З (q_0) ⇒ ЗЗ 1 0 101 З (q_1) ⇒ ЗЗ 11 0 101 З (q_1)

⇒ ЗЗ 110 1 01 З (q_1) ⇒ ЗЗ 1101 0 1 З (q_1) ⇒

⇒ ЗЗ 11010 З (q_1) ⇒ ЗЗ 11010 З (q_F).