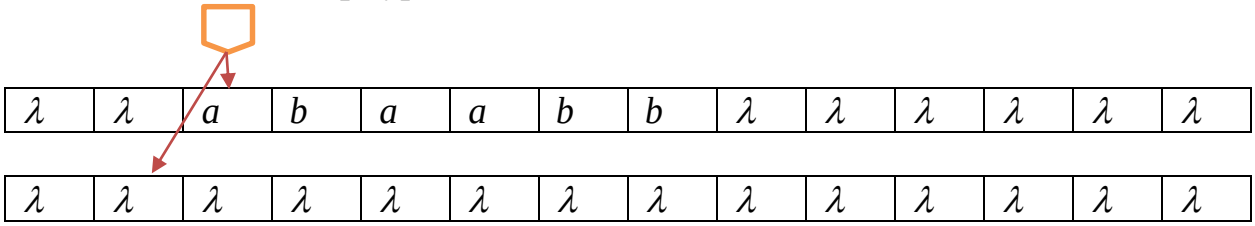
МЛТА. Лекція 12.04.2021

Приклад 1. Нехай МТ є двохстрічковою. Задано слово в алфавіті $A=\{a,b\}$, λ – порожній символ. Скопіювати це слово на другу стрічку в зворотному порядку.

$q_1\{a,\lambda\}q_1\{a,\lambda\}\{R,S\}$ $q_1\{b,\lambda\}q_1\{b,\lambda\}\{R,S\}$	Проходимо на першій стрічці до кінця заданого слова
$q_1\{\lambda,\lambda\}q_2\{\lambda,\lambda\}\{L,S\}$	Побачивши порожній символ (а значить, ми пройшли задане слово) починаємо на першій стрічці рух вліво
$q_2\{a,\lambda\}q_2\{a,a\}\{L,R\}$ $q_2\{b,\lambda\}q_2\{b,b\}\{L,R\}$	Копіюємо на другу стрічку задане слово в зворотному порядку
$q_2\{\lambda,\lambda\}q_0\{\lambda,\lambda\}\{R,S\}$	Зупиняємося на перших символах слів на кожній стрічці

Початкова конфігурація



Приклад 2. Нехай МТ має три стрічки. Алфавіт $A=\{1\}$, λ – порожній символ. Задано два числа в унарному коді. Вони записані на першій і другій стрічках відповідно. Визначити добуток цих чисел і результат записати на третю стрічку.

Зауваження: при записі числа в унарному коді число n кодується $n+1$ одиницею.

$q_1\{1,1,\lambda\}q_2\{1,1,1\}\{R,R,R\}$	На першій і другій стрічках проходимо перші одиниці, які означають ознаку числа (число кодується кількістю одиниць на одну більше величини числа). Ставимо ознаку числа на третю стрічку
$q_2\{1,1,\lambda\}q_3\{1,2,1\}\{S,R,R\}$ $q_3\{1,1,\lambda\}q_3\{1,2,1\}\{S,R,R\}$	На другий стрічці замінюємо одиницю на двійку і записуємо одиницю на третю стрічку
$q_3\{1,\lambda,\lambda\}q_4\{1,\lambda,\lambda\}\{S,L,L\}$	

$q_4\{1,2,1\}q_4\{1,1,1\}\{S,L,S\}$	Якщо на другий стрічці одиниці (крім першої) замінені на двійки, рухаємося вліво і відновлюємо одиниці
$q_4\{1,1,1\}q_2\{1,1,1\}\{R,R,R\}$ $q_2\{\lambda,1,\lambda\}q_0\{\lambda,1,\lambda\}\{L,L,L\}$	Закінчуємо роботу так: на першій і третій стрічках пристрій читання / запису на останньому символі, на другий стрічці - на першому.

Порівняння часу роботи комп'ютерів і машини Тьюрінга

1. *Імітація машини Тьюрінга на комп'ютері і комп'ютера на машині Тьюрінга.*

До основних компонентів обчислювальної машини відносяться оперативна пам'ять і процесор. Програми та дані, представлені в двійковому алфавіті, поміщаються в пам'ять. При виконанні програми окремі її команди і потрібні дані витягуються з пам'яті в процесор і навпаки – значення, отримані при виконанні команд, записуються в комірки пам'яті.

Пам'ять складається з деякого числа запам'ятовуючих комірок (регістрів), призначених для проміжного зберігання значень операндів і для зберігання іншої інформації, яка є необхідна для виконання команд, регістрів для керування запам'ятовуючими комірками, адрес комірок і полів самих комірок.

Процесор складається з пристрою керування (ПК) і арифметичного пристрою (АП). Пристрій керування містить лічильник тактів, команд тощо, виробляє керуючі сигнали для виконання команд, передачі даних тощо. Процесор містить регістри операндів, лінії зв'язку і лінії затримки для безпосередньої реалізації процесів обчислень.

Окрім процесора і пам'яті комп'ютеру необхідні ще пристрої введення/виведення.

Імітація машини Тьюрінга на комп'ютері. Нехай T – машина Тьюрінга, однією зі складових якої є її скінченне керування (керуючий пристрій). Оскільки T має скінченне число станів і скінченне число правил переходу, програма комп'ютера може закодувати стани у вигляді ланцюжків символів, як і символи її зовнішнього алфавіту, і використовувати таблицю переходів машини T для перетворення ланцюжків. Нескінченну стрічку машини Тьюрінга можна імітувати змінними дисками, розміщеними в двох магазинах, відповідно для даних, розташованих на стрічці зліва і праворуч від голівки, що зчитує. Чим далі в магазині розташовані дані, тим далі вони від головки на стрічці.

Для імітації комп'ютера на машині Тьюрінга є істотними два факти:

– чи існують інструкції, що виконуються комп'ютером, і недоступні для машини Тьюрінга;

– чи працює комп'ютер швидше за машину Тьюрінга.

Неформальна модель реального комп'ютера:

– пам'ять, що складається з послідовності слів і їх адрес. За адресу вибираються натуральні числа 0,1, …;
– програма комп'ютера, що записана в слова пам'яті, кожне з яких представляє просту інструкцію. Допускається "непряма адресація" за вказівниками;
– кожна інструкція використовує скінченне число слів і змінює значення не більше одного слова;
– є слова пам'яті з швидким доступом (регістри), але швидкість доступу до різних слів впливає лише на константний співмножник, що не змінює поліноміальних залежностей.

Можлива конструкція машини Тьюрінга для імітації комп'ютера представлена на рис.

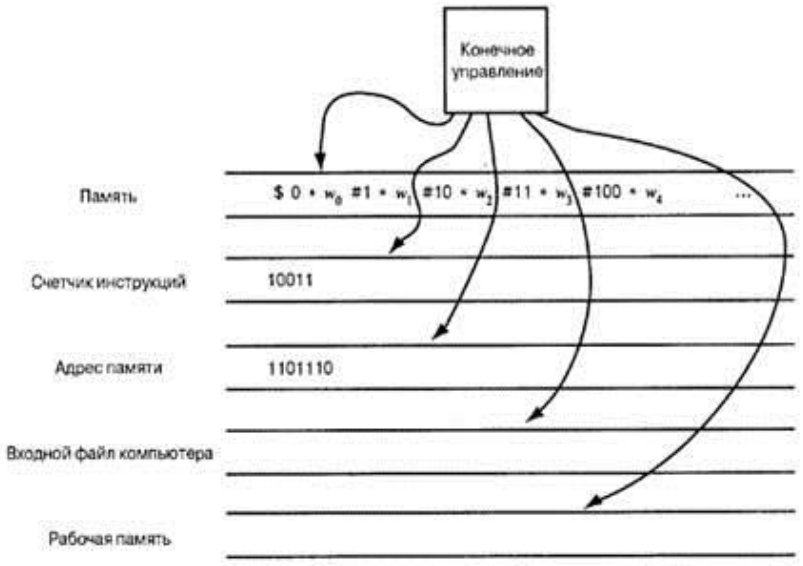


Рис.

Машина Тьюрінга має кілька стрічок. Перша стрічка представляє всю пам'ять комп'ютера – адреси і значення (в двійковій системі). Адреси закінчуються маркером *, значення – маркером #. Початок і кінець записів 1-й стрічки позначаються маркером $.

Друга стрічка – "лічильник інструкцій", містить одне двійкове ціле, що представляє одну з позицій голівки, що зчитує, на першій стрічці, адресу інструкції, яка повинна бути виконана наступною.

Третя стрічка містить адресу і значення за нею після того, як ця адреса встановлюється на першій стрічці. Для виконання інструкції машина Тьюрінга повинна знайти значення за одним або декількома адресами пам'яті, де зберігаються дані, які беруть участь в обчисленні. Потрібна адреса копіюється на стрічку 3 і порівнюється із адресами на стрічці 1 до співпадання. Значення за цією адресою копіюється на третю стрічку і переміщується на потрібне місце, як правило, за однією з початкових адрес, які представляють регістри комп'ютера.

Четверта стрічка імітує вхідний файл.

П'ята стрічка – робоча пам'ять, яка служить для виконання обчислень.

Функціонування такої машини, що імітує комп'ютер:

1. Знайшовши на 1-й стрічці адресу, що співпадає з номером інструкції на 2-й стрічці, досліджуємо значення за цією адресою і копіюємо на 3-ю стрічку. Перші біти інструкції задають дію (копіювати, вставити, розгалузитися тощо), інші біти – адреса або адреси, які використовуються в цій дії.

2. Якщо в інструкції міститься значення за деякою адресою, то ця адреса копіюється на 3-ю стрічку, а позиція інструкції на 2-ю доріжку 1-й стрічки.

3. Далі інструкція виконується. З новим значенням можна зробити наступне:

a) скопіювати за іншою адресою;

Іншу адресу вилучають із інструкції, записують на 3-ю стрічку, знаходять на 1-й стрічці і значення по ньому копіюють в зарезервований для нього простір. Якщо для нового значення треба більше (менше) пам'яті, ніж для старого, простір змінюється шляхом зсуву, а саме,

(1) на робочу стрічку копіюється частина стрічки праворуч від того місця, куди потрібно ввести нове значення;

(2) нове значення записується на 1-ю стрічку;

(3) робоча частина копіюється назад на 1-ю стрічку праворуч від нового значення.

b) додати знайдене значення за іншою адресою;

Шукаємо другий адресу на першій стрічці, виконуємо додавання значення за цією адресою і значення, записаного на 3-й стрічці.

c) перейти до виконання інструкції за адресою, записаною на 3-й стрічці, для чого стрічка 3 копіюється на стрічку 2, і цикл інструкцій починається знову.

4. Виконавши інструкцію (яка не є переходом), додаємо 1 до лічильника на стрічці 2 і знову починаємо цикл інструкції.

Тепер потрібно переконатися, що якщо проблему можна розв'язати за поліноміальний час на комп'ютері, то її можна розв'язати за поліноміальний час на машині Тьюрінга і навпаки.

Час роботи машини Тьюрінга, що імітує комп'ютер

Введемо такі обмеження на модель комп'ютера:

– Жодна комп'ютерна інструкція не повинна породжувати слово, яке довше, ніж на 1 біт, від своїх операндів.

– Інструкція, що застосовується до слів довжини m повинна виконуватися не більше, ніж за $O\left(m^2\right)$ кроків на багатострічковій машині Тьюрінга (відмінність у часі роботи однострічковій і багатострічковій машин Тьюрінга є поліноміальна).

Назвемо такі операції допустимими.

Цим умовам задовольняють додавання, зсув на 1 біт, порівняння значень, які виконуються на багатострічковій машині Тьюринга за $O\left(m\right)$ кроків. А

також множення m-бітових цілих, якщо його імітувати за допомогою m послідовних складань зі зрушеннями на 1 біт вліво. Час виконання операції множення буде пропорційним до квадрату довжини співмножників.

Теорема. Для комп'ютера, що має вказані властивості, описана вище модель машини Тьюрінга може імітувати m кроків комп'ютера не більше, ніж за $O\left(m^3\right)$ кроків.

Доведення. Спочатку перша стрічка містить тільки програму комп'ютера, довжина якої не залежить від n (числа кроків виконання інструкцій). Найбільше з комп'ютерних слів або адрес, що зустрічаються в програмі, позначимо через c, а через d – число слів програми.

Після виконання n кроків комп'ютер не може породити слово, довше за $c+n$, і не може створити або використовувати адресу, що займає більше $c+n$ бітів. Кожна інструкція породжує не більше однієї нової адреси, що одержує значення, тому після виконання n інструкцій маємо $d+n$ адрес. Кожна адреса-значення займає не більше $2(c+n)+2$ розрядів, а після виконання n інструкцій не більше за $2(d+n)(c+n+1)$, або $O\left(n^2\right)$.

Для перегляду адрес однієї інструкції комп'ютера потрібно часу $O\left(n^2\right)$, слова мають довжину $O\left(n\right)$, а інструкції виконуються машиною Тьюрінга за час $O\left(n^2\right)$, зсув для створення простору для нового слова включає копіювання даних об'ємом $O\left(n^2\right)$ з стрічки 1 на робочу стрічку і назад. Отже, машина Тьюрінга імітує один крок комп'ютера за $O\left(n^2\right)$ своїх кроків, а n кроків можна проімітувати за $O\left(n^3\right)$ кроків машини Тьюрінга.

Теорема. Виконання n кроків роботи комп'ютера можна проімітувати на однострічковій машині Тьюрінга не більше ніж за $O\left(n^6\right)$ кроків.

Отже, машина Тьюрінга може імітувати пам'ять і управління реального комп'ютера, використовуючи тільки одну стрічку для запису всіх елементів пам'яті і їх вмісту – регістрів, основної пам'яті, дисків та інших пристроїв, що запам'ятовують. Звідси можна бути впевненим, що все, що не здійсненне машиною Тьюрінга, не може бути обчислено і комп'ютером.

Нормальні алгоритми Маркова

Нормальний алгоритм в алфавіті T (скорочено НА в алфавіті T, Markov algorithm) – це упорядкована послідовність підстановок (продукцій, правил) вигляду

$$\alpha \to \beta$$

або

$$\alpha \to \cdot \beta,$$

де $\alpha, \beta \in T^*$ та $\{\to, \cdot\} \notin T$, T^* – множина слів скінченої довжини в алфавіті T.

Підстановки вигляду $\alpha \to \cdot \beta$ називають ***заключними*** **(*фінальними*)**, а $\alpha \to \beta$ – ***простими***.

Довільну скінченну послідовність підстановок називають ***схемою*** нормального алгоритму.

Кожен НА в алфавіті T задає певне словарне відображення $T^* \to T^*$. Слово, яке є результатом обробки слова x нормальним алгоритмом $\mathcal{A}$, позначимо $\mathcal{A}(x)$.

Цікавою особливістю нормальних алгоритмів Маркова є те, що в них використовується тільки одна елементарна дія – підстановка.

Слово γ входить в слово ξ, якщо існують слова δ_1 і δ_2 (можливо порожні) такі, що

$$\xi = \delta_1 \gamma \delta_2.$$

Марківські підстановки виконуються так. У заданому слові P знаходять ***перше*** входження слова α (якщо воно є) і, не змінюючи інших частин слова P, замінюють в ньому це входження словом β. Отримане слово називається результатом застосування марківської підстановки $\alpha \to \beta$ до слова P. Якщо ж немає першого входження слова α до слова P (і, отже, взагалі немає жодного входження α до P), то вважається, що марковська підстановка $\alpha \to \beta$ не може бути застосована до слова P.

Обробку слова x нормальним алгоритмом $\mathcal{A}$ проводять поетапно.

Нульовий етап. Покладемо $x_0 = x$ і скажемо, що x_0 отримане із слова x після 0 етапів.

$(n+1)$*-й етап*. Нехай слово x_n отримане зі слова x після n етапів. Шукаємо ***першу за порядком*** продукцію $\alpha \to (\cdot) \beta$ таку, що α – підслово x_n. Запис точки в дужках означає, що точка може стояти на цьому місці, а може бути відсутньою. Застосуємо цю продукцію до x_n, тобто замінимо в x_n найлівіше (перше) входження слова α на слово β. Отримане слово позначимо x_{n+1}.

Якщо застосована на $(n+1)$-му етапі продукція не фінальна, тобто $\alpha \to \beta$, то переходимо до $(n+2)$-го етапу.

Якщо ця продукція фінальна, тобто $\alpha \to \cdot\beta$, то після її застосування нормальний алгоритм $\mathcal{A}$ зупиняється і $\mathcal{A}(x) = x_{n+1}$.

Якщо ж на $(n+1)$-му етапі жодна продукція нормального алгоритму $\mathcal{A}$ незастосовна до x_n, тобто в $\mathcal{A}$ немає продукцій, ліва частина яких − підслово слова x_n, то $\mathcal{A}$ зупиняється і $\mathcal{A}(x) = x_n$.

Якщо в процесі обробки слова x НА $\mathcal{A}$ не зупиняється ні на якому етапі, то вважаємо, що результат обробки слова x нормальним алгоритмом $\mathcal{A}$ невизначений.

Нормальний алгоритм називають ***нормальним алгоритмом над алфавітом*** T (НА над T), якщо він – нормальний алгоритм у деякому розширенні $T' \supseteq T$.

НА над T задає словарне відображення $T^* \to T^*$, використовуючи в процесі обробки слів допоміжні символи поза алфавітом T. Зупинка НА над T при роботі над словом $x \in T^*$ є ***результативна,*** коли алгоритм зупинився з результатом $y \in T^*$, інакше результат роботи нормального алгоритму над словом x є невизначений.

Зауважимо, що порожній символ λ у необмеженій кількості зустрічається ***перед/між/після*** будь-яких символів скінченного алфавіту T.

Нормальні алгоритми $\mathcal{A}$ і $\mathcal{B}$ ***еквівалентні відносно алфавіту*** T, якщо для всіх $x \in T^*$ значення $\mathcal{A}(x)$ і $\mathcal{B}(x)$ одночасно визначені або невизначені, та у випадку визначеності $\mathcal{A}(x) = \mathcal{B}(x)$.

Відомо, що для кожного НА над алфавітом T існує еквівалентний йому відносно T НА в алфавіті $T \cup \{s\}$ з єдиним допоміжним символом $s \notin T$. Словарне відображення, яке кожне слово $x \in T^*$ переводить у слово xx, не може бути заданим жодним НА в алфавіті T, але це можна зробити в алфавіті T'.

Нормальний алгоритм $\mathcal{A}$ ***обчислює часткову функцію*** $f : N_0{}^k \to N_0$, якщо він кожне слово вигляду $x_1, x_2, \ldots, x_k$ переводить в слово $f(x_1, x_2, \ldots, x_k)$ у випадку $(x_1, x_2, \ldots, x_k) \in D_f$ и $\mathcal{A}(x_1, x_2, \ldots, x_k)$ – невизначене за $(x_1, x_2, \ldots, x_k) \notin D_f$.

Функцію називають ***обчислюваною за Марковим*** або ***НА-обчислюваною***, якщо існує нормальний алгоритм Маркова, який її обчислює.

Зауважимо, що кожний НА обчислює множину функцій натуральних аргументів та значень, але, зафіксовуючи наперед арність функцій, дістаємо, що кожний НА обчислює єдину функцію заданої арності.

Приклад 1. Нехай заданий алфавіт $T = \{a, b, c, d\}$ і λ – порожній символ. У слові P потрібно замінити перше входження підслова *bb* на *ddd* і видалити всі входження символу c.

Наприклад, $P = abbcabbca$. Слово $P' = adddabba$.

1) $c \to \lambda$ 2) $bb \to \bullet ddd$	1) $bb \to ddd$ 2) $c \to \lambda$
$P = abbcabbca$ (знаходимо перше входження символу c в слово P і замінюємо його на порожній символ) $abbabbca$ (знаходимо перше входження символу c в слово P і замінюємо його на порожній символ) $abbabba$ (починаємо з першої підстановки, слово не містить символ c, переходимо до другої підстановки, знаходимо перше входження підслова bb і замінюємо його на ddd. Підстановка є заключною, тому алгоритм закінчує роботу) $P' = adddabba$ (результат)	$P = abbcabbca$ $adddcabbca$ $adddcadddca$ $adddadddca$ $adddaddda$ Отримали слово, в якому ***усі*** входження підслова bb замінені на ddd.