Білет №4

1. Різновиди алгоритмів. Композиція алгоритмів.

1) Нехай система П є такою, що система величин σ_{i+1} однозначно визначена системою σ_i. Алгоритм $A=<\varphi, П>$ називається детермінованим, якщо задовольняє цю властивість, і недетермінованим в іншому випадку.

У детермінованому алгор. $A=<\varphi, П>$ алфавітний оператор φ є однозначним, у недетермінованому - багатозначним.

2) Алгоритм $A=<\varphi, П>$ називається самозмінним, якщо в процесі переробки алгоритмом вхідних слів система П змінюється, і несамозмінним в іншому випадку.

3) Нехай система правил алгоритму $A=<\varphi, П>$ закодована певним способом у вхідному алфавіті алгоритму A. Позначимо цей код P^{cod}.

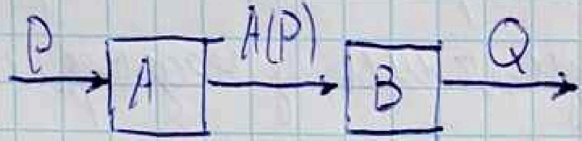
$A=<\varphi, П>$ називається самозастосовним, якщо слово P^{cod} входить в область визначення алгоритму і несамозастосовним в протилежному випадку.

4 Алгоритм називають універсальним, якщо він еквівалентний довільному наперед заданому алгоритму $A = < \varphi, \Pi >$.

Композиція алгоритмів.

Є 4 ~~со~~ способи композиції алгоритмів: суперпозиція, об'єднання, розгалуження та ітерація.

1. Суперпозиція алгоритмів.



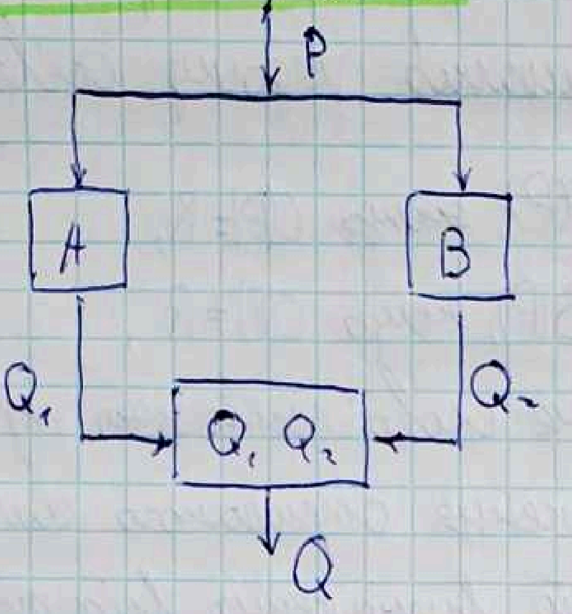
При суперпозиції двох алгоритмів A та B вихідне слово одно з них розглядають як вхідне слово іншого. $C(P) = B(A(P))$, $P \in D(A)$, $A(P) \in D(B)$

Суперпозиція може виконуватись для довільної скінченності к-сті алгоритмів.

Суперпозиц. алгоритмів $A(P) = xP$ та $B(P) = Py$ є алг.

$$C(P) = xPy$$

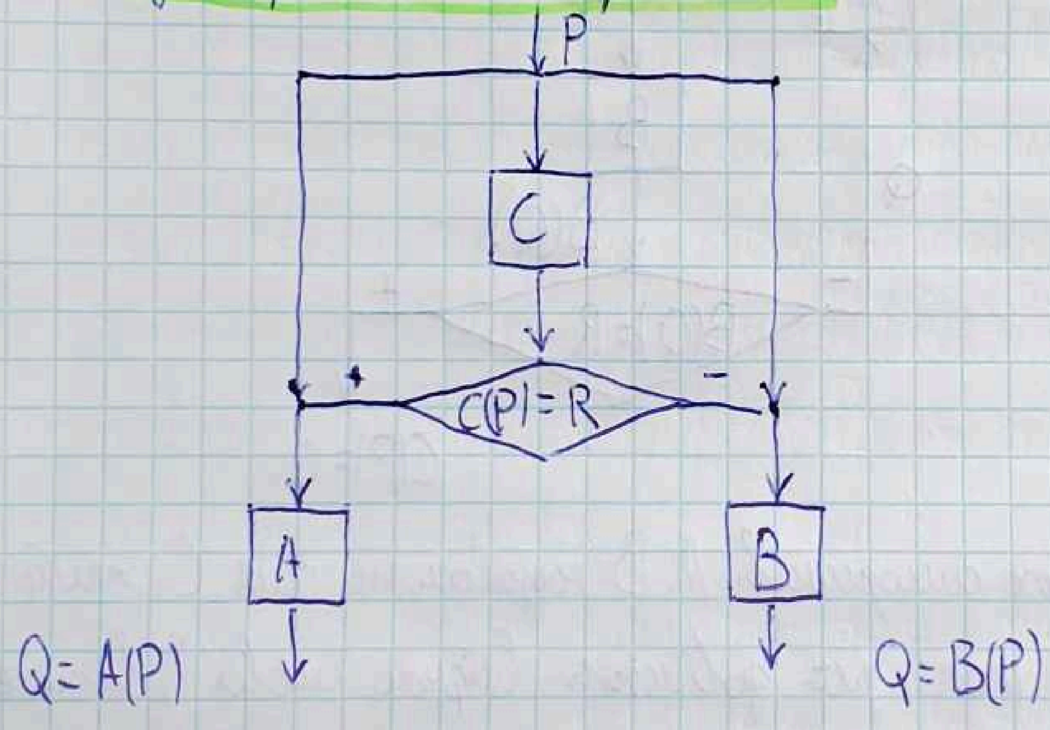
2. Об'єднання алгоритмів



$P \in D(A) \cap D(B)$ $C(P) = A(P)\,B(P)$

На всіх інших словах C вважають невизначеним.

3. Розгалуження алгоритмів



Розгалуженням алгоритмів називають композицію 3-ох алгоритмів задану співвідношенням

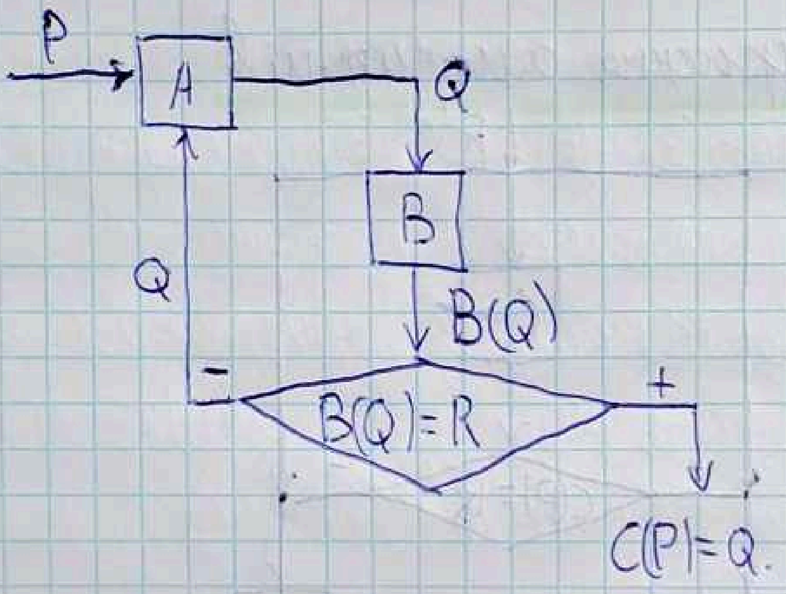
$$F(P)=\begin{cases} A(P), \text{ якщо } C(P)=R; \\ B(P), \text{ якщо } C(P)\neq R, \end{cases}$$

де R – фіксоване слово (переважно порожнє).

Областю визначення отриманого алгоритму буде перетин областей визначення всіх трьох алгоритмів

$$D(F)=D(A)\cap D(B)\cap D(C)$$

4. Ітерація алгоритмів.



Ітерацією двох алгоритмів A і B називають ал. C, який визначають так: для довільного вхідного слова P, вихідне слово $C(P)$ отримують як результат послідовного багаторазового застосування алг. A доти, доки не буде отримано слово, перетворюване алгоритмом B в деяке фіксоване слово R.

У процесі ітерації A формується послідовність слів: $Q = Q_0, Q_1, \ldots, Q_n = C(P)$, $Q_i = A(Q_{i-1})$ $(i = \overline{1, n})$ $B(Q_i) \neq R$

2. Формальне визначення машини Тьюрінга. Теза Тьюрінга

З математичного погляду машина Тьюрінга – це певний алгоритм для переробки вхідних слів. Оск спосіб переробки машинних слів відомий, якщо відома програма машини, то МТ вважають заданою, якщо задані її зовнішній і внутр алфавіти, програма та визначено, які з символів відповідають порожній комірці й заключному стану.

Кожна МТ реалізує ф-ію $\Pi : S \times \{Q / q_F\} \to S \times Q \times Z$ або інакше $\Pi = \{ <s_i, q_i> \to <s_j, q_r, Z> \}, q_i \neq q_F$

Тоді формально МТ називають сімку $(S, Q, q_0, q_F, I, \Lambda, \Pi)$, де $I \subseteq S$ – мн. вхідних символів, Λ – порожн. символ.

Якщо визначити строго формально МТ як матем. об'єкт, то можна на формальному рівні визначити поняття конфігурації, процесу обчислення і всі інші поняття.

Кажуть, що слово P задане в стандартному положенні, коли головка фіксує перший зліва символ вхідного слова. Якщо за початкову конфігурацію K_{t_0} взяти стандартне положення слова P, то процес $MT(K_{t_0})$ є процесом переробки вхідного слова P, і його позначають MT(P).

MT обчислює деяку словникову ф-ію, $\varphi(P)$ яка відображає слова з деякої множини слів у допустимі (не порожні) слова, якщо для довільного P процес MT(P) досягає заключної стандартної конфігурації, асоц. з словом $\varphi(P)$ - результатом цієї ф-ії.

Якщо для часткової словникової ф-ії $\varphi(P)$ існує MT, яка її обчислює, то $\varphi(P)$ наз. обчислюваною за Тьюрінгом.

Теза Тьюрінга.

Для довільного алгоритму $A = <\varphi, \Pi>$ у довільному скінченному алфавіті X існує ф-ія $\varphi(P): \{P\}_x \to \{Q\}_x$, обчислювана за Тьюрінгом.

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

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

Теорема 1.

Всі часткові словникові ф-ї, обчислювані за Тьюрінгом, є частково-рекурсивними.

Теорема 2.

Для кожної частково-рекурсивної словникової ф-ї, визначеної в деякому алфавіті $M=\{a_1, \dots, a_n\}$ існує МТ з відповідним внутрішнім алфавітом, зовнішній алфавіт якої збігається з M $(S=M)$, яка і обчислює задану ф-ію.

3. Довести, що ф-ія $f(x_1, x_2) = x_1 \div x_2$ є примітивно-рекурсивною.

Розглянемо побудову часткової ф-ії f через ф-ї g і h:

$$\begin{cases} f(x_1, x_2, \dots, x_n, 0) = g(x_1, x_2, \dots x_n) \\ f(x_1, x_2, \dots x_n, y+1) = h(x_1, \dots x_n, y, f(x_1, \dots x_n)) \end{cases}$$

Оск ділення на нуль заборонене, розглянемо ф-ію $f(x, 1)$

$$f(x_1, 1) = \frac{x_1}{1} = x_1 = I_1^2(x, 1) = g(x)$$

Тоді для $x_2 \geq 1$, $x_1 \in N$

$$f(x_1, x_2+1) = \frac{x_1}{x_2+1} = S^3(\div, I_1^2, S^1(I_2^2)) = h(x_1, x_2, f(x_1, x_2))$$

$$\Rightarrow g(x) = I_1^2(x_1, 1)$$

$$h(x_1, x_2, f(x_1, x_2)) = S^3(\div, I_1^2, S^1(I_2^2))$$

($h(x_1, x_2, f(x_1, x_2))$ - елементарна ф-ія відносно ділення, оск виражається через оператор суперпозиції та ф-ію вибору аргумента.)

Приклад

$$f(4, 2) = S^3(\div, I_1^2, S^1(I_2^2)) = S^3(\div, 4, S^1(1)) = \frac{I_1^2(4, 1)}{2} = \frac{4}{2} = 2.$$