1973 год ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ Выпуск 54

УДК 681.142.2:519.9

ЯЗЫК ВЫЧИСЛЕНИЙ НА ГРАФАХ

В.О. Васин

Методы и средства теории графов позволяют сравнительно просто формулировать и решать довольно широкий класс задач в экономике, теории оптимального планирования, лингвистике, химии, физике, социологии, теории автоматов и. т.д. Известно также, что в большинстве своем эти задачи носят комбинаторный характер и имеют большие размерности, поэтому для их решения требуется применение современных ЭЦВМ, что ьедет к необходимости программирования алгоритмов их решения.

В настоящее время программирование сводится к записи алгоритма на одном из алгоритмических языков, но распространенные универсальные языки (такие как АЛГОЛ, ФОРТРАН, ЛИСП и др.) не приспособлены для записи алгоритмов указанного типа. Известно несколько специализированных языков. Среди них наиболее удобны для задач теории графов и множеств язык GRASPE [1] и 2-й уровень языка ЛЯПАС [2]. Однако язык GRASPE не оснащен собственной программирующей системой, а в языке ЛЯПАС не проведены накопление и классификация Л-операторов для решения задач данного типа.

Работа по выделению и классификации таких операторов проведена на кафедре кибернетики и вычислительной техники Севастопольского приборостроительного института. Ее результатом была разработка специализированного языка ЯГАЛ (Язык Графов Алгоритмический), предназначенного для записи алгоритмов решения задач теории графов и конечных множеств. В основе своей ЯГАЛ опирается на ЛЯПАС, но обладает собственной семантикой, символикой и синтаксисом. Язык полностью формализован и, подобно АЛГОЛу, описан в языке нормальных форм Бэкуса [3].

Для выбора базовых операций языка проведен анализ различных алгоритмов и программ решения задач теории графов, среди них такие, как программы обработки больших сетевых графиков [4] раскраски графа, определение изоморфизма пары графов, поиска планарной реализации и др. Выделены некоторые наиболее важные структурные единицы программ, представляющие характерные блоки (процедуры, операторы) при решении задач указанного типа.

Символика, семантика и синтаксис языка разрабатывались с таким расчетом, чтобы запись алгоритмов в языке по форме приближалась к общепринятой в дискретной математике [5, 6] символической записи. К общематематическим понятиям добавлены операции управления ходом вычислительного процесса, распределения памяти и общения человека с машиной.

I. Описание языка ЯГАЛ

I. Структура языка. Язык ЯГАЛ представляет собой растущую и развивающуюся открытую систему, как и язык ЛЯПАС, имеет два уровня: первый – предполагает запись алгоритмов в основных символах языка; второй – допускает применение процедур и блоков.

К настоящему времени разработан первый уровень языка и обслуживающая его часть программирующей системы.

Программа на языке ЯГАЛ ("яграмма") строится из последовательности "выражений", представляющих синтаксически правильные цепочки основных символов.

2. Основные символы языка.

<основной символ> ::= <операнд> / <разделитель> .

2.I. <операнд> ::= <буква> / <цифра>.

<разделитель> ::= <знак операции> / <знак оператора> / <указатель>.

Алфавит операндов и смысл каждого символа операнда совпадают с описанием операндов языка ЛЯПАС.

2.2. <буква> ::= <буква прописная> / <буква строчная> / <буква индексная> / <буква специальная> / <буква константная>.

2,3. <цифра> ::= 0/I/2/3/4/5/6/7/8/9.

2.4. <число> ::= <цифра> / <цифра> <цифра> / 0<цифра><цифра> / I <цифра> <цифра>.

2.5. <разделитель> ::= <знак операции> / <знак оператора> / <указатель>.

2.6. <знак операции> ::= <связка> / <знак множественной операции> / <знак арифметической операции> / <знак предикатной операции> / <знак операции управления> / <знак операции отношения>.

<связка> ::= & / ∨ .

& – логическое "И", ∨ – логическое "ИЛИ".

<знак множественной операции> ::= ∪ / ∩ / ∖ / ⊕.

∪ – объединение, ∩ – пересечение, ∖ – вычитание, ⊕ – сравнение множеств.

<знак арифметической операции> ::= +/-/:/×.

Арифметические операции определяются так же, как в языке ЛЯПАС.

<знак предикатной операции> ::= ∀ / ∃ / { / }.

∀ – квантор общности – "для всех...", ∃ – квантор существования, – "если существует...", {...} фигурные скобки перечисления элементов множества – "сформировать из элементов".

<знак операции управления> ::= §/↣/→/,/↓/⇓.

§ – метка "параграф", ↣ – безусловная передача управления "перейти к...", → – импликация – "если..., то...", , – запятая – "иначе...".

2.7. <знак операции отношения> ::= <знак включения> / <знак сравнения> / <знак вхождения>.

<знак включения> ::= ⊂ / ⊄ / ⊆ / ⊈ .

⊂ – собственно включает (⊄ – не включает), ⊆ – включает (⊈ не включает) подмножеством.

<знак сравнения> :: < / ≥ / =/ ≠.

< – меньше, ≥ – не меньше, = – равно, ≠ – не равно.

<знак вхождения> ::= ∈ / ∉.

∈ – входит (∉ – не входит) элементом множества.

2 3. <знак оператора> ::= <знак оператора памяти> / <знак оператора экстремума> / <знак оператора перевода> / <знак оператора присвоения> / <знак оператора общения> / <знак оператора над графами>.

<знак оператора памяти> ::= Зп / Вс.

Зп – запомнить, Вс – восстановить.

<знак оператора перевода> ::= ② / ⑩.

② – перевести в двоичную систему, ⑩ – перевести в IO-чную систему.

<знак оператора экстремума> :: min / max .

<знак оператора присвоения> ::= Δ / $\circ$ / $\Rightarrow$.

Δ - элементарное приращение (на I), $\circ$ - присвоить значение "ноль", $\Rightarrow$ - присвоить (передать) значения.

<знак оператора общения> ::= / / *.

/ - ввести информацию с ВУ, * - выдать информацию на...

<знак оператора над графами> ::= ∇ / $\bar{\nabla}$ / $\cup$ / / / / Γ / ρ / / .

$\bar{\nabla}$ - исключить ребро, ∇ - добавить ребро, $\cup$ - слить пару вершин / - добавить вершину, / - исключить вершину, Γ - множество соседства, ρ - степень вершины (локальная), / - инверсия порядка следования элементов множества.

2.9. <указатель> ::= (/) / / /. / λ .

( ) - круглые скобки, / - наклонная черта "такие,что...:, . - точка - конец яграммы, λ - пометка сложного множества (типа 2).

Допускается произвольное расширение алфавита символов, но это ведет к перестройке транслятора. В версии ЯГАЛ-71 зафиксированы 54 символа разделителей и 177_{10} символов операндов (см. таблицу). Буквы используются для идентификации величин, а разделители имеют фиксированный смысл, который в большинстве случаев ясен из их названий. В дальнейшем смысловые значения разделителей будут рассмотрены подробно.

3. В е л и ч и н ы . Объекты языка - посылки и результаты вычислений - называются величинами. Величины представляются своими идентификаторами.

3.I. <величина> ::= <идентификатор множества> / <идентификатор ячейки>.

<идентификатор множества> ::= <буква прописная>.

Идентификаторы множества представляют в данной версии следующие величины:

множества типа I, элементы которых расположены по одному в ячейке (для их представления достаточно 32 разрядов);

множества типа 2, элементами которых являются множества типа I;

множества типа 3, элементы которых однозначно сопоставлены элементам множества типа I и представляют "веса" (метки) последних;

множества типа 4, элементы которых "упакованы" по нескольку штук в ячейке машины.

Коды	Символы	
	00	40
0		∪
I	§	∩
2	(	\
3	)	⊕
4		↣
5	↓	
6	.	∀
7		∃
I0	∧	{
II	,	Γ
I2	0	ρ
I3	)	
I4	⇒	→
I5		↑
I6	Зп	•
I7	Вс	∨
20	∈	
2I	∉	/
22	=	&
23	≠	
24	<	⅄
25	≥	:
26	⊂	x
27	⊄	↕
30	⊆	+
3I	⊈	-
32	min	$\bar{\nabla}$
33	max	∇
34		↻
35		/
36	◇10	/
37	◇2	⇓

Коды разделителей языка ЯГАЛ

<идентификатор ячейки> ::= <место> / <константа> / <число>.

Идентификатор ячейки представляет величину, значение которой хранится в одной ячейке машины (32 разряда).

3.2. <место> ::= <идентификатор переменной> / <мощность множества> / <простой элемент множества>.

<идентификатор переменной> ::= <буква строчная> / <буква индексная>.

<мощность множества> ::= /<идентификатор множества> /.

<простой элемент множества> ::= <идентификатор множества> <буква индексная> / / <идентификатор множества> <число>.

Термин "место" означает величину, хранящуюся в одной ячейке, значение которой может меняться в процессе вычислений.

3.3. <константа> ::= <буква константная> <цифра> <цифра> / <буква константая> <буква индексная>.

Не допускается изменение значений величин "константа" и "число" в процессе вычислений.

4. В ы р а ж е н и я . Фундаментальным понятием для построения яграммы является понятие "выражение".

4.I. <выражение> ::= <инородное выражение> / <условное выражение> / <предикатное выражение> / <вычислительное выражение> / <операторное выражение> / <выражение выражение> / <пусто>.

4.2. Понятие "пусто" это либо последовательность нулей, перед которыми нет значащей цифры, либо пробел.

4.3. <инородное выражение> ::= / <идентификатор переменной> / ⇓ <ЯПАС-цепочка> ⇓ .

$\uparrow\alpha$ означает переход к реализации машинной программы, начинающейся с ячейки (α) МОЗУ (аналогично переходу в МП языка ЛЯПАС).

$\Downarrow\alpha\Downarrow$ означает реализацию произвольного выражения α , записанного в языке ЛЯПАС. Недопустимо применение символа . (точка) в выражении α .

4.4. <условное выражение> ::= <отношение> → <выражение> , <выражение>

4.4.а. <отношение> ::= <идентификатор ячейки> <знак сравнения> <идентификатор ячейки> / <идентификатор ячейки> <знак вхождения> <идентификатор множества> / <идентификатор множества> <знак операции отношения> <идентификатор множества> / <отношение> <связка> <отношение> / <арифметическое выражение> <отношение> <арифметическое выражение>.

Отношение означает логическое высказывание, которое может быть истинным или ложным. Например, отношения $a=b$, $c\perp\alpha$, $f\in A$, $K=L$, $A\subset M$, $|A|\geq|B|$, $D\subseteq S$, $k\neq l\ \&\ M\in S$, $b\geq m\vee a\in B$, $a+b=|B|-3\times m$.

Условное выражение $\alpha\rightarrow\gamma,\beta$ означает, что если выполняется отношение α , то выполнять действия, указанные в выражении γ , иначе перейти к выполнению действий, предписанных выражением β . Следует отметить, что выполнение действий производится обычно в порядке их следования, если нет операций перехода, поэтому после выполнения операций выражения γ возможно также выполнение операций выражения β .

Например, участок ярограммы ... $a+b=c\rightarrow k$, $a+b\Rightarrow c$ ... имеет следующий смысл: "если $(a)+(b)=(c)$, то перейти к реализации машинной программы, начинающейся в ячейке (k), после этого c присвоить значение $(a)+(b)$, в противном случае реализация МП не выполняется".

4.5. <предикатное выражение> ::= <существование> / <всеобщность> / <выборка>.

4.4.а. <существование> ::= $\exists$ <вхождение> / <условное выражение>.

<вхождение> ::= <идентификатор переменной> $\in$ <идентификатор множества> / <идентификатор переменной> <идентификатор переменной> $\in$ <идентификатор множества> / <идентификатор множества> $\in$ <идентификатор множества> / (<идентификатор множества> <число>) $\in$ <идентификтор множества>.

Термин "вхождение" означает принадлежность элементов некоторому множеству. Например:

$a\in A$ – элементы множества A идентифицируются знаком a (a принадлежит A).

$ab\in B$ – пары элементов множества B обозначены индексами a и b .

$E\in S$ – элемент сложного множества S идентифицируется знаком E .

$(\angle 3)\in S$ – элементы множества S ,"упакованные" по 3 элемента в ячейке,идентифицируются тремя элементами множества $\angle$. Так, если в некоторой ячейке расположены элементы множества S – S_i, S_{i+1}, S_{i+2} то $\angle_0$ соответствует S_i, $\angle_1 - S_{i+1}$ и $\angle_2 - S_{i+2}$.

Понятие "существование" имеет смысл квантора существования. Например, выражение $\exists a\in B\ /\alpha\rightarrow\beta,\gamma$ означает следующее: "если существует элемент a множества B такой, что выполняется отношение α , то реализуются операции выражения β ,иначе – выражения γ ". Аналогично $\exists ab\in C\ /\alpha\rightarrow\beta,\gamma$ – "если существует пара a и b элементов множества C таких, что...".

$\exists\angle\in D\ /\alpha\rightarrow\beta,\gamma$ –"если существует элемент $\angle$ сложного множества D такой, что...".

$\exists\ (\angle\ 12)\in S/\alpha\rightarrow\beta,\gamma$ – "если существует среди упакованных в множестве S по 12 штук в ячейке и представленных множеством $\angle$ элементов S_i найдется такой, что α истинно, то выполнять β , иначе γ ".

4.5,б. <всеобщность> ::= $\forall$ <вхождение> (<выражение>).

Понятие "всеобщность" имеет смысл квантора общности. Например, выражение $\forall a\in B(\alpha)\beta$. означает "для всех элементов a множества B в порядке их перечисления выполнить действия,указанные выражением α , затем выполнять β". Аналогичные значения имеет понятие "всеобщность" с другими видами "вхождения" . Номер текущего элемента при переборе по $\forall$ и $\exists$ фиксируется специальным индексом λ.

4.5.с. <выборка> ::= {<вхождение> / <отношение>} $\Rightarrow$ <идентификатор множества>.

Смысл понятия "выборка" в том, что множество, указанное справа, комплектуется из элементов, указанных во "вхождении" и удовлетворяющих заданному отношению. Например: $\{b\in E/b<c\ \&\ b-1\geq\alpha\vee b=k\}\Rightarrow B$ означает "множество B сформулиро –

вать из элементов b множества E таких, что (b) меньше (c) и (b) – I не меньше (a) или (b) равно (k)."

4.6.<Вычислительное выражение> ::= <арифметическое выражение> / <множественное выражение>.

4.6.а. <арифметическое выражение> ::= <ячейка> /<ячейка> <знак арифметической операции>.

Понятие "арифметическое выражение" имеет обычный смысл и в пояснениях не нуждается.

4.6.b. <множественное выражение> ::= <идентификатор множества><знак множественной операции><идентификатор множества>⇒<идентификатор множества> / <множественное выражение>∧/ /<идентификатор множества> <знак множественной операции> <идентификатор переменной>⇒<идентификатор множества>.

Понятие "множестве..ное выражение" имеет обычный смысл,например, $A \cup B \Rightarrow C$ означает, что результат объединения множеств A и B присваивается множеству C , или $C \oplus D \Rightarrow C\wedge$ означает, что результат сравнения множеств C и D , т.е. множество элементов из $(C \cup D) \setminus (C \cap D)$, присваивается множеству C . Указатель $\wedge$ означает, что в операции участвуют множества типа 2 (сложные).Допускается совпадение левого и последнего идентификаторов множеств.

4.7. <операторное выражение> ::=<оператор> / <оператор> <выражение> / <выражение> <оператор>.

4.7.а. <оператор> ::= <оператор памяти> /<оператор экстремума> / <оператор общения> / <оператор перевода> / <оператор управления> / <оператор присвоения >/ <оператор над графами>.

Понятие "оператор" означает элементарный блок программы, имеющий самостоятельное значение.

4.8. <оператор памяти> ::= Вс <величина> / Зп <величина>.

Этот оператор предназначен для запоминания на ВЗУ значения величины и восстановления его в ОЗУ.

4.9. <оператор экстремума> ::= max<ячейка> <ячейка>⇒<место> / min <ячейка> <ячейка>⇒<место>.

Этот оператор величине, указанной справа от знака ⇒, присваивает наименьшее (или наибольшее) из двух значений, указанных слева от знака ⇒ . Например: $min\ a\,b \Rightarrow c$ после выполнения такого оператора значение c будет равно $min\{(a),(b)\}$.

4.10. <оператор общения> ::= ↑ / <величина> * <цифра> / /<величина> * I <цифра>.

Оператор общения осуществляет обмен информацией между ОЗУ и внешними устройствами машины. Знак ↑ – ввод очередной порции исходных данных на вводном устройстве.

Знак * – выход к внешним накопителям и печати. Число после * означает вид обращения:

00 – печать 8-я со сдвигом
01 – перфорация 8-я со сдвигом
02 – печать 8-я без сдвига
03 – перфорация 8-я без сдвига
04 – чтение с МБ
05 – запись на МБ
06 – чтение с МЛ
07 – запись на МЛ
10 – печать 10-я
11 – печать АЦПУ
12 – 17 – печать форматная

При обращении к ВЗУ начальный адрес (номер МЛ и зоны или номер начальной ячейки МБ) задается значением выделенной переменной $\underline{z}$, поэтому использовать $\underline{z}$ в качестве обычной переменной нежелательно.

4.11. <оператор перевода> ::= ② <ячейка> / ⑩ <ячейка>.

Оператор ② переводит значение величины, указанной справа, из двоично-десятичной записи в двоичную.

Оператор ⑩ переводит значение величины,указанной справа, из двоичной записи в двоично-десятичную.

4.12. <оператор управления> ::=⟶ <число> / § <число>

⟶ ε – означает безусловную передачу управления на метку § ε .

4.13. <оператор присвоения> ::= Δ <место> /○ <место>/ / ○ <идентификатор множества> / ○ <идентификатор множества> (<идентификатор множества>) / <ячейка>⇒<место> / <ячейка> ⇒(<идентификатор переменной> <идентификатор переменной>)/ (<идентификатор переменной> <идентификатор переменной>)⇒<идентификатор переменной> /⇒<идентификатор множества>⇒<идентификатор множества>.

Операторы присвоения меняют значение величины, указанной справа от знака операции присвоения. Например: $\Delta\, a$ означает, что значение индекса a получит приращение на I. Такая запись

эквивалентна записи $\alpha + 1 \Rightarrow \alpha$. $O\alpha$ означает, что новое значение величины α будет "0". OD означает, что множество D считается пустым, его мощность равна нулю. $OB(A)$ означает, что все элементы множества меток B (тип 3) принимают значение "нуль"; i -й элемент множества B является "весом" i -го элемента множества A , т.е. мощности множества A и B совпадают. $\alpha \Rightarrow b$ -величина α передает свое значение b . $A \Rightarrow B$ -новое значение i-го элемента множества B равно (A_i) , новое значение мощности $/B/$ равно $/A/$. $\alpha \Rightarrow (bc)$ - оператор введен для "распаковки" значения α . Новые значения b составляют левые 20_8 разрядов α , сдвинутые вправо на 20, c - правые 20_8 разрядов α . $(bc) \Rightarrow \alpha$ - оператор "упаковки" переменных b и c в одну ячейку α . При этом b сдвигается на 20_8 разрядов влево, объединяется с 20_8 правыми разрядами c и заносится в α .

4.I4. <оператор над графами> ::= <идентификатор множества> ∇ <идентификатор переменной> <идентификатор переменной> / /<идентификатор множества> $\bar{\nabla}$ <идентификатор переменной> <идентификатор переменной> / $\updownarrow$ <идентификатор множества> / <идентификатор множества> <идентификатор множества> / <идентификатор переменной> <идентификатор множества> / <идентификатор множества> <идентификатор множества> / <идентификатор переменной> / /<идентификатор множества> $\cup$ <идентификатор переменной> <идентификатор переменной> / Γ <идентификатор множества> (<идентификатор переменной>) $\Rightarrow$ <идентификатор множества> / ρ <идентификатор множества> (<идентификатор переменной>) $\Rightarrow$ <место>.

Операторы над графами преобразуют множества вершин и ребер графа. Например:

Bvc - множество ребер B дополнить ребром (cc).

$B \bar{\nu} \alpha\beta$ из множества ребер B исключить ребро $(\alpha\beta)$.

$\updownarrow A$ - изменить порядок перечисления элементов множества A на обратный.

AB/cD в множество вершин A и ребер B графа добавить вершину c , соединенную с вершинами, перечисленными в множестве D .

AB/c из множества вершин A и ребер B графа исключить вершину c и все инцидентные ей ребра.

$AB \cup c\alpha$ - в графе, представленном множеством вершин A и множеством ребер B, провести объединение (склеивание) вершин c и α . Полученную при этом вершину назвать c .

$\Gamma B(a) \Rightarrow D$ - множество вершин, смежных a в графе, представленном множеством ребер B , представить множеством D .

$\rho B(a) \Rightarrow k$ - значение локальной степени вершины a в графе, представленном множеством ребер B присвоить индексу k .

При обработке направленных графов понятие "ребро" заменяется понятием "дуга", понятие "множество соседства" (смежных вершин) - понятием "множество образов" (достижимых за один шаг вершин) и понятие "локальная степень" - понятием "полустепень исхода".

5.I. <яграмма> ::= <выражение>./<выражение> <яграмма>.

Таким образом, программа в языке ЯГАЛ определяется как произвольное выражение, оканчивающиеся знаком . (точка).

П. Программирующая система ПС-ЯГАЛ

ПС-ЯГАЛ производит перевод записи алгоритма в языке ЯГАЛ на язык конкретной машины. Перевод осуществляется в два этапа:

I. Блоки ПС-ЯГАЛ преобразуют запись яграммы на язык ЛЯПАС П уровня с осуществлением синтаксической проверки и коррекции яграммы.

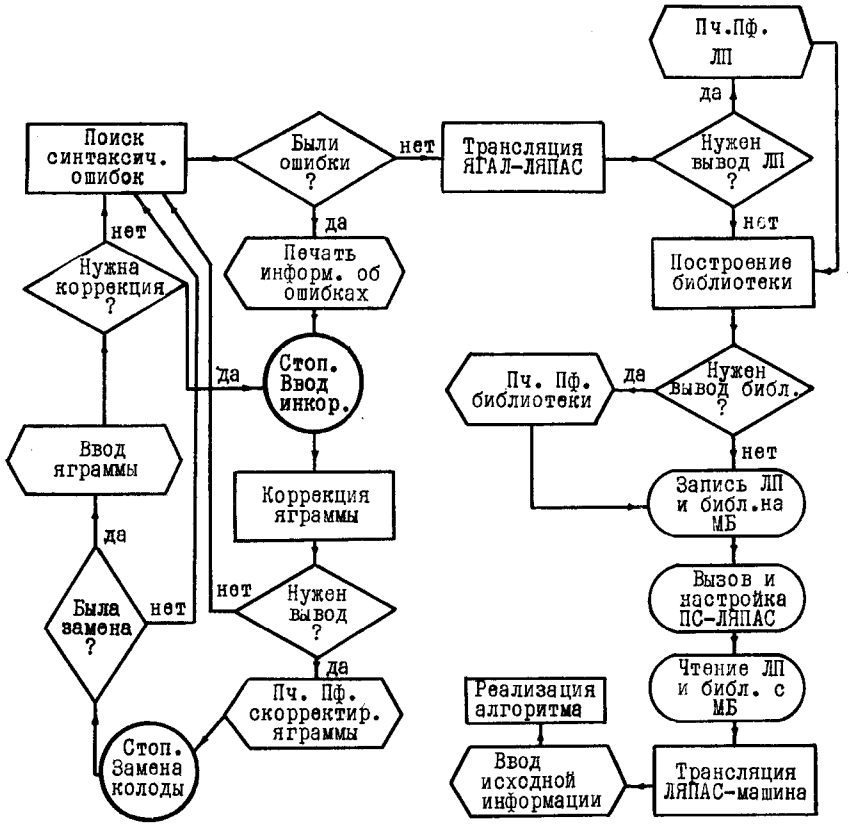
2. Вызывается ПС-ЛЯПАС, осуществляющая перевод программы на язык конкретной машины.

Блоки синтаксического контроля, коррекции, трансляции и составляющие библиотеку Л-операторов включаются по мере необходимости программой-диспетчер На рисунке приведена блок-схема ПС-ЯГАЛ-дирижер.

Кодирование исходной информации для ПС-ЯГАЛ осуществляется так же, как в языке ЛЯПАС (см.табл.). Допустима "длина" яграммы около 3600 кодов языка. Время перевода яграммы средней длины до уровня языка ЛЯПАС 2-3 мин. Качество оттранслированных программ не намного хуже качества программ "ручного" программирования.

Ш. Практическое применение языка

Наглядность языка ЯГАЛ дает возможность использовать его в качестве языка публикаций. Рассмотрим для примера записи некоторых алгоритмов.



I. З а д а ч а о б о п т и м а л ь н о й с е т и к о м м у н и к а ц и й . Пусть необходимо построить сеть коммуникаций в некотором районе с n пунктами.

Задан полный граф $G(X,U)$. Каждому ребру $u_s=(i,j)\in U$ сопоставляется вес (стоимость) c_{ij}. Найти суграф $T(X,V)(V\subseteq U)$, сохраняющий связность графа и такой, что $\sum_{ij\cdot T} c_{ij}=min$. Очевидно, что T -дерево.

Алгоритм решения этой задачи дан О.Борувкой (1926) и улучшен Дж.Б. Краскалом (1956) [15]. Заключается он в следующем: выбирается ребро $u_{ij}\in U$ для покрывающего дерева T с минимальным весом c_{ij} и удаляется из U. Далее на каждом шаге выбирается (из оставшихся в U) ребро с минимальным весом и такое, что его добавление в T не приводит к образованию циклов. Алгоритм заканчивается, когда к T невозможно добавить ни одного ребра или количество ребер в T равно числу вершин графа G без единицы. Минимальность $\sum_{ij} c_{ij}$ весов ребер дерева и его связность гарантируется.

Запишем программу алгоритма на языке ЯГАЛ.

Пусть A - множество вершин заданного графа G.

B - множество ребер графа G.

C - множество весов (меток) ребер из B с весом α.

T - множество ребер формируемого дерева, тогда:

$OT\ O\alpha$

§I $f_{10}\Rightarrow b\ \forall a\in D(a<b-a\Rightarrow b\ \dot{\Lambda}\Rightarrow c,)\ B_c\Rightarrow a\Rightarrow(k\ell)$

$B\setminus a\Rightarrow B\ D\setminus b\Rightarrow D\ O\ C\ldots\Rightarrow C$

$\forall c\in C(\Gamma T(c)\Rightarrow E\ C\cup E\Rightarrow C\ \ell\in C-\rightarrowtail 1,)\ \alpha+b\Rightarrow\alpha$

$T\cup\alpha\Rightarrow T/T/<|A|-1-\rightarrowtail 1,T*O\ \alpha*O.$

Запись можно прочесть так:

считать множество T пустым, α приравнять нулю.

§ I. f_{10} заслать в b (b должно иметь большое значение, принято значение f_{10} - константы с I во всех разрядах ячейки).

Для каждого элемента a из множества D выполнить: если a меньше b, то b принимает значение a и адрес веса α (он же адрес ребра из B) запоминается в c, иначе-перейти к анализу очередного элемента $a\in D$. По окончании перебора (когда определится минимальный по величине вес ребра) ребро, представленное значением B_c, запомнить в a, его начало засылать в k, конец - в ℓ.

Исключить ребро α из B. Исключить вес b из D.

Сформировать вспомогательное множество C с единственным элементом k. Для каждого элемента c множества C определить множество соседних в T вершин и запомнить в E. Добавить в C элементы из E, и если ℓ является элементом C, то (вершины k и ι достижимы в T и добавление ребра $(k\ell)$ приведет к появлению цикла) перейти к метке § I, в противном случае повторить вычисления с очередным элементом из C. После проверки всех

элементов из C (можно утверждать, что k и ℓ принадлежат различным компонентам связности формируемого дерева и ребро $(k\ell)$ можно включить в T) добавить к весу дерева α вес b нового ребра. Добавить ребро $(k\ell)$, представленное a, в перечень ребер T, и если мощность T меньше количества вершин графа $/A/$, уменьшенного на I, то повторить процесс с метки § I, в противном случае (найдено дерево T и его α) отпечатать в восьмеричной системе перечень ребер дерева T и их суммарный вес α. Конец программы.

Такая запись алгоритма и ее расшифровка не претендуют на оптимальность и служат скорее целям иллюстративного характера.

2. Разметка потенциалов сети. В целом ряде задач экономики (задача о кратчайшем пути, задача об оптимальном потоке, расчет сетевого графика и др.) используется алгоритм Форда разметки потенциалов сети. Представим его записью на языке ЯГАЛ.

Пусть B -перечень ребер сети.

C - метки (веса) ребер из B .

D - потенциалы (метки) вершин сети.

a - номер начальной вершины (источника).

$B*0\ C*0\ \forall\alpha\in D\,(\tilde{C}_f \Rightarrow D_j)\circ D\alpha$

§I $\circ\underline{b}\ \forall c\in B\,(c\Rightarrow(\alpha e)D\alpha+C_j \Rightarrow \underline{a}<D_e \rightarrow \underline{a}\Rightarrow D_e\,\Delta\,\underline{b}$,

$D_e+C_j \Rightarrow \underline{a}<D\alpha \rightarrow \underline{a}\Rightarrow D\alpha\,\Delta\,\underline{b},)\,\underline{b}\neq 0 \rightarrow \rightarrow 1,\ D*0.$

Расшифруем запись.

Отпечатать в восьмеричной системе исходные данные B и C. Каждому элементу α из D присвоить значение $\tilde{C}_f$ (для формирования исходных потенциалов предполагается, что веса ребер много меньше числа, представленного константой $(\tilde{C}_f) = 2^{31}$). Начальной вершине a присвоить потенциал-метку "нуль".

§ I. $\underline{b}$ придать значение "нуль". Для всех элементов c из множества ребер B выполнить: начало ребра заслать в α, конец - в e. Если сумма a потенциала вершины α $D\alpha$ и цены C_j рассматриваемого ребра c меньше потенциала вершины D_e, то значение суммы $\underline{a}$ считать новым потенциалом D_e, дать элементарное приращение $\underline{b}$, иначе, если сумма $\underline{a}$ потенциала D_e и веса ребра C_j меньше потенциала D_α, то считать новым значением потенциала D_α значение суммы $\underline{a}$, дать элементарное приращение b, в противном случае вернуться к анализу очередного ребра. После окончания перебора, если $b\neq 0$ (т.е. процесс уменьшения потенциалов не установился), повторить вычисления с метки §I, иначе считать процесс закончившимся и отпечатать в восьмеричной системе значения потенциалов. i-й элемент множества D представляет потенциал вершины номер i.

Этот алгоритм (несколько модифицированный) включен в программу обработки больших (до I0000 событий) сетевых графиков и успешно эксплуатируется в вычислительных центрах ВМФ и в ВЦ СКБ "Машпроект".

Литература

I. PRATT T.W., FRIEDMAN D.P. A Language Extension for Graph Processing and its Formal Semantics. -"Communications of the ACM", 1971, vol.14, July, N 7, p.460-467.

2. Логический язык для представления алгоритмов синтеза. М., "Наука", I966.

3. Алгоритмический язык АЛГОЛ-60.Пересмотренное сообщение. М., "Мир", I965.

4. ЗЫКОВ А.А. Теория конечных графов. Новосибирск,"Наука" Сиб.отд., I969.

5. ШИХАНОВИЧ Ю.А. Введение в современную математику. М., "Наука", Физматгиз, I965.

6. ОРЕ О. Теория графов.М.,"Наука",Физматгиз,I968,стр.84.

Поступила в ред.-изд.отд.
29 июля I972 г.