УДК 681.142.1.01

НЕКОТОРЫЕ АЛГОРИТМЫ ПОСТРОЕНИЯ КРАТЧАЙШЕЙ ЦЕПИ

В.А. Тюренков

1. Алгоритм Форда

В настоящей статье рассматриваются некоторые эффективные алгоритмы построения кратчайшей цепи между заданными вершинами α и β неориентированного связного графа (X, Γ) с ребрами положительной длины [1]. Как правило, такие алгоритмы состоят из трех частей: настройки, основной части и обратного хода. При выполнении настройки и основной части элементы графа (т.е. вершины или ребра) помечаются некоторыми индексами, которые затем используются при обратном ходе для построения собственно цепи. Под п а м я т ь ю $V(A)$ алгоритма A будем понимать количество двоичных разрядов, которые необходимо отводить для запоминания этих индексов в памяти ЭВМ при программировании алгоритма. Все описываемые ниже алгоритмы (кроме алгоритма A_2) работают, когда длины ребер выражаются произвольными (т.е. не обязательно целыми) положительными числами. Для случая, когда длины ребер выражаются целыми числами, даются оценки памяти $V(A)$.

Для построения кратчайшей цепи между вершинами α и β взвешенного графа обычно применяется алгоритм Форда (см., например, [1]). Этот алгоритм заключается в следующем:

Н а с т р о й к а. Помечаем вершину α индексом $\lambda(\alpha) = 0$, остальные вершины x - индексом $\lambda(x) = \infty$.

О с н о в н а я ч а с т ь. Ищем ребро (x, y) такое, что вершина x помечена конечным индексом $\lambda(x)$ и

$$\lambda(y) - \lambda(x) > \lambda(x, y), \qquad (1)$$

где $\lambda(x,y)$ – длина ребра (x,y) ; затем заменяем индекс $\lambda(y)$ индексом

$$\lambda(y)=\lambda(x)+\lambda(x,y).$$

Повторяем процесс уменьшения индексов, пока они не установятся.

О б р а т н ы й х о д. Среди вершин множества $\Gamma(b)$ выбираем одну из вершин x, для которой

$$\lambda(b)-\lambda(x)=\lambda(x,b),$$

и обозначаем выбранную вершину через x_1. Среди вершин множества $\Gamma(x_1)$ выбираем одну из вершин x, для которой

$$\lambda(x_1)-\lambda(x)=\lambda(x,x_1),$$

и обозначаем выбранную вершину через x_2. Продолжаем выбор вершин, пока на каком-то $(k+1)$-м шаге не окажется выбранной вершина a. Вершины $[a,x_k,x_{k-1},\dots,x_1,b]$ и образуют кратчайшую цепь между a и b.

Для алгоритма Форда индексы $\lambda(x)$, выражая длину цепи, могут быть весьма велики. В самом деле, так как алгоритм не предписывает какого-либо правила, по которому выбирается очередное ребро (x,y), удовлетворяющее условию (I), не исключена возможность, что вершины y будут сначала выбираться вдоль некоторой максимальной элементарной цепи, исходящей из a. Поэтому в процессе работы алгоритма максимальное возможное значение индекса $\lambda(x)$ равно $\max\limits_{x\in X} L(a,x)$, где $L(a,x)$ – длина максимальной элементарной цепи между вершинами a и x. Применяя при программировании вместо символа ∞ максимальное возможное значение символа $\lambda(x)$, имеем (для случая, когда длины ребер выражаются целыми числами):

$$V(A_1)=n\cdot(1+[\log_2 \max_{x\in X} L(a,x)]),$$

где через A_1 обозначен алгоритм Форда, n – общее количество вершин графа (X,Γ), $[\dots]$ – целая часть числа.

Ниже предлагаются алгоритмы с меньшей памятью и в меньшей степени связанные с перебором, чем алгоритм Форда.

2. Волновая модификация алгоритма Форда

Рассмотрим сначала случай, когда длины всех ребер равны единице. Тогда для построения кратчайшей цепи между вершинами a и b может быть применен следующий алгоритм A_2:

Н а с т р о й к а совпадает с настройкой алгоритма Форда.

О с н о в н а я ч а с т ь. Если все вершины, помеченные индексом $\lambda(x)=i$, образуют известное множество $A(i)$, то помечаем индексом $i+1$ вершины множества

$$A(i+1)=\{x \mid x\in\Gamma A(i),\ x\notin A(j) \text{ при всех } j\le i\}.$$

Останавливаемся, как только вершина b окажется помеченной.

О б р а т н ы й х о д совпадает с обратным ходом алгоритма Форда.

Этот алгоритм, описанный в книге Бержа [1], впоследствии был назван волновым алгоритмом и затем рассматривался в работах некоторых американских (напр.,[2]) и советских ([3], [4], [5]) авторов. Очевидно,

$$V(A_2)=n\cdot(1+[\log_2 l(a,b)]),$$

где $l(a,b)$ – длина кратчайшей цепи между a и b.

Пусть теперь длины ребер выражаются произвольными положительными числами. Модифицируя алгоритм Форда в соответствии с идеей волнового алгоритма, получаем следующий алгоритм A_3:

Н а с т р о й к а. Помечаем вершину a индексами $\lambda(a)=0$ и $\varphi(a)=0$, а все остальные вершины x – индексами $\lambda(x)=\infty$ и $\varphi(x)=1$.

О с н о в н а я ч а с т ь.

1. Во множестве вершин, помеченных индексом $\varphi(x)=0$, выбираем одну из вершин x с минимальным значением индекса $\lambda(x)$. У этой вершины признак $\varphi(x)=0$ заменяем признаком $\varphi(x)=1$. Если $x=b$, то переходим к обратному ходу.

2. Для выбранной вершины x ищем ребро (x,y) такое, что выполняется неравенство (I); затем заменяем индекс $\lambda(y)$ индексом $\lambda(y)=\lambda(x)+\lambda(x,y)$ и индекс $\varphi(y)$ индексом $\varphi(y)=0$. Повторяем п. 2, пока не установятся индексы на множестве $\Gamma(x)$. Переходим к п. 1.

О б р а т н ы й х о д совпадает с обратным ходом алгоритма Форда.

Теорема 2.1. А л г о р и т м A_3 к о р р е к т е н.

Для доказательства теоремы рассмотрим вспомогательный алгоритм A_3', который отличается от A_3 только тем, что заменяет признак $\varphi(x)=0$ для найденной в п. 1 вершины x признаком $\varphi(x)=-1$. Из описания алгоритма A_3 видно, что он эквивалентен алгоритму A_3'. Пусть c – счетчик, который при настройке алгоритма A_3' устанавливается на нуль и значение которого увеличивается на единицу каждый раз при замене какого-либо индекса $\varphi(x)=0$ индексом $\varphi(x)=-1$ и каждый раз при переходе от

п. 2 к п. I. Пусть t - целое неотрицательное число. Через $\lambda_t(x)$ и $\varphi_t(x)$ будем обозначать, соответственно, значения функций $\lambda(x)$ и $\varphi(x)$ при $\tau = 2t$.[x)]

Лемма 2.I. О с н о в н а я ч а с т ь а л г о р и т м а A_3' р е з у л ь т а т и в н а (т.е. в процессе работы алгоритма A_3' обязательно наступит такой момент, когда в п. I будет выбрана вершина b).

В самом деле, индукцией по $t \geq 1$ легко доказывается неравенство

$$\max_{\varphi_t(x)=-1} \lambda_t(x) \leq \min_{\varphi_t(x)=0} \lambda_t(x). \qquad (2)$$

Из (2) следует, что при $t \geq 0$

$$\varphi_{t+1}(x) \leq \varphi_t(x). \qquad (3)$$

Непосредственно из описания алгоритма A_3' следует, что множества $\{x / \varphi_t(x) = -1\}$ и $\{x / \varphi_t(x) = 1\}$ не смежны.[xx)] Поэтому из связности графа (X, Γ) следует, что множества $\{x / \varphi_t(x) = 0\}$ и $\{x / \varphi_t(x) = 1\}$ смежны. Отсюда и из (3) следует справедливость леммы 2.I.

Лемма 2.2. О б р а т н ы й х о д а л г о р и т м а A_3' р е з у л ь т а т и в е н, п р и ч е м ц е п ь

$$[a, x_k, x_{k-1}, \ldots, x_1, b], \qquad (4)$$

п о с т р о е н н а я в п р о ц е с с е о б р а т н о г о х о д а а л г о р и т м а A_3', я в л я е т с я к р а т ч а й ш е й ц е п ь ю м е ж д у в е р ш и н а м и a и b .

Нетрудно видеть, что существуют вершины, обеспечивающие выполнение обратного хода алгоритма A_3'. В самом деле, x_1 - это последняя из вершин, послуживших для уменьшения индекса $\lambda(b)$, x_2 - последняя из вершин, послуживших для уменьшения индекса $\lambda(x_1)$, и т.д.

Дальнейшее доказательство леммы 2.2 опирается на такие два утверждения:

если вершина x принадлежит цепи (4), то в момент окончания работы алгоритма A_3' имеем

$$\varphi(x) = -1 ; \qquad (5)$$

---

x) Заметим, что множество $\{x / \varphi_t(x) = 0\}$ естественно назвать t-м ф р о н т о м в о л н ы.

xx) Множества M_1 и M_2 вершин графа (X, Γ) мы называем с м е ж н ы м и, если существуют смежные вершины $x \in M_1$ и $y \in M_2$.

если $\varphi_t(x) = -1$ и вершины x и y смежны, то

$$-\lambda(x,y) \leq \lambda_t(y) - \lambda_t(x) \leq \lambda(x,y). \qquad (6)$$

Первое из этих утверждений легко доказывается с привлечением неравенства (2), а второе - индукцией по t .

Теперь заметим, что процесс простановки индексов $\lambda(x)$ алгоритмом A_3' можно рассматривать как процесс простановки индексов $\lambda(x)$ алгоритмом Форда, прерванный в момент выбора вершины b . Продолжим прямой ход (т.е. основную часть) алгоритма A_3' до тех пор, пока не будет существовать ни одной вершины, для которой $\varphi(x) = 0$. Тогда все вершины графа (X, Γ) будут помечены индексом $\varphi(x) = -1$, и из неравенства (6) будет следовать, что полученная система индексов $\lambda(x)$ совпадает с окончательной системой индексов $\lambda(x)$ алгоритма Форда. Так как при продолжении прямого хода алгоритма A_3' не изменился индекс $\lambda(x)$ ни у одной из вершин цепи (4) (это следует из (5) и (3)), то цепь (4) могла быть выбрана также и алгоритмом Форда, а потому она является кратчайшей.

Лемма 2.2 доказана полностью.

Из лемм 2.I и 2.2 следуют корректность алгоритма A_3' и справедливость теоремы 2.I.

Если длины ребер выражаются целыми числами, то при программировании алгоритма A_3' для запоминания индексов $\lambda(x)$ необходимо отводить в памяти ЭВМ $n \cdot (1 + [\log_2(l(a,b) + l)])$ двоичных разрядов, где l - длина максимального ребра графа (X, Γ) .Для запоминания индексов $\varphi(x)$ необходимо n разрядов. Следовательно, $V(A_3) = n \cdot (2 + [\log_2(l(a,b) + l)])$.

3. Уменьшение величины индексов $\lambda(x)$

С помощью волновой модификации алгоритма Форда можно построить алгоритм, который вместо индексов $\lambda(x) (0 \leq \lambda(x) \leq l(a,b) + l)$ использует индексы $\lambda'(x) (0 \leq \lambda'(x) \leq 2l)$.

Для этого прежде всего заметим, что в п. I алгоритма A_3' индексы $\lambda(x)$ нужны только для нахождения во множестве $\{x / \varphi_t(x) = -0\}$ вершины с минимальным значением $\lambda_t(x)$, в п. 2 - чтобы для выбранных смежных вершин x и y графа (X, Γ) определить значение предиката

$$P_1(t, x, y) \equiv (\lambda_t(y) - \lambda_t(x) > \lambda(x, y)),$$

и, наконец, при обратном ходе - чтобы для смежных вершин x и y

определить значение предиката

$$P_2(x,y) \equiv (\lambda_\tau(y) - \lambda_\tau(x) = \lambda(x,y)),$$

где $(2\tau+1)$ - значение счетчика τ в момент окончания работы алгоритма A_3'.

Рассмотрим вспомогательный алгоритм A_3'', который отличается от A_3' только наличием дополнительных индексов $\lambda'(x)$: при настройке алгоритма вершина a помечается индексом $\lambda'(a) = 0$, а все остальные вершины x - индексом $\lambda'(x) = \infty$; затем в процессе дальнейшего применения этого алгоритма вершины y графа (X,Γ) каждый раз одновременно с индексами $\lambda(y) = \lambda(x) + \lambda(x,y)$ помечаются еще и индексами:

$$\lambda'(y) = \lambda'(x) + \lambda(x,y) \pmod M,$$

где M - натуральное число, пока неопределенное, и $\lambda'(x)+\lambda(x,y) \pmod M$ -наименьшее неотрицательное число, сравнимое с $\lambda'(x)+\lambda(x,y)$ по модулю M.

Если $M \le 2\ell$, то могут существовать две смежные с x вершины y_1 и y_2 такие, что $\lambda(y_1) - \lambda(y_2) = M$. Тогда $\lambda'(y_1) = \lambda'(y_2)$, но $\lambda(y_1) - \lambda(x) \neq \lambda(y_2) - \lambda(x)$. Пользуясь этим, нетрудно показать, что для вычисления предикатов $P_1(t,x,y)$ и $P_2(x,y)$ можно обойтись знанием только чисел $\lambda_t'(x)$,*) $\lambda_t'(y)$ и $\lambda(x,y)$, для чего необходимо выполнение неравенства $M > 2\ell$. Леммы 3.1 и 3.2 утверждают, что достаточно взять $M = 2\ell+1$.

Лемма 3.1. Если $M = 2\ell+1$, вершины x и y смежны и $\lambda'(x) \neq \infty$, то условие (I) эквивалентно выполнению одного из условий:

$$\lambda(x,y) < \lambda'(y) - \lambda'(x) \le \ell, \tag{7}$$

$$\lambda(x,y) - (2\ell+1) < \lambda'(y) - \lambda'(x) \le -(\ell+1), \tag{8}$$

$$\lambda'(y) = \infty. \tag{9}$$

В самом деле, пусть в точке x_0 функция $\lambda_t(x)$ достигает минимального значения во множестве $\{x/\varphi_t(x) = 0\}$ и $x_1 \in \{x/\varphi_{t+1}(x) = 0\}$. Индукцией по t легко доказывается неравенство

$$0 \le \lambda_{t+1}(x_1) - \lambda_t(x_0) \le \ell. \tag{10}$$

Если x и y - произвольные смежные вершины, а $\lambda(x)$ и $\lambda(y)$ конечны, то из (6) и (10) следует:

$$-\ell \le \lambda(y) - \lambda(x) \le \ell. \tag{11}$$

---

*) Здесь и ниже через $\lambda_t'(x)$ обозначается значение функции $\lambda'(x)$ при $\tau = 2t$.

Так как $M = 2\ell+1$, то для любых двух вершин x и y с конечными $\lambda(x)$ и $\lambda(y)$ имеем:

$$\lambda(y) - \lambda(x) = (k_2 - k_1)(2\ell+1) + \lambda'(y) - \lambda'(x), \tag{12}$$

где k_1 и k_2 - целые числа.

Для смежных вершин x и y из (12) и (11), учитывая, что $0 \le \lambda'(y) \le 2\ell$ и $0 \le \lambda'(x) \le 2\ell$, получаем

$$-\left(1 + \frac{\ell}{2\ell+1}\right) < k_2 - k_1 < 1 + \frac{\ell}{2\ell+1},$$

откуда $k_2 - k_1 = -1, 0, 1$. Полагая последовательно $k_2 - k_1 = -1, 0, 1$, из (12) и (11) получаем:

а) выполняется одно из неравенств:

$$\ell+1 \le \lambda'(y) - \lambda'(x), \tag{13}$$

$$-\ell \le \lambda'(y) - \lambda'(x) \le \ell, \tag{14}$$

$$\lambda'(y) - \lambda'(x) \le -(\ell+1); \tag{15}$$

б) если (13), то $\lambda(y) - \lambda(x) = -(2\ell+1) + \lambda'(y) - \lambda'(x)$;

в) если (14), то $\lambda(y) - \lambda(x) = \lambda'(y) - \lambda'(x)$;

г) если (15), то $\lambda(y) - \lambda(x) = (2\ell+1) + \lambda'(y) - \lambda'(x)$.

С учетом того, что $\lambda'(y) - \lambda'(x) < 2\ell+1$, справедливость леммы 3.1 получаем непосредственно из утверждений (а), (б), (в) и (г).

Лемма 3.2. Если $M = 2\ell+1$, вершины x и y смежны и индекс $\lambda'(y) \neq \infty$, то условие

$$\lambda(y) - \lambda(x) = \lambda(x,y)$$

эквивалентно выполнению одного из условий: $\lambda'(y) - \lambda'(x) = \lambda(x,y)$,

$$\lambda'(y) + (2\ell+1) - \lambda'(x) = \lambda(x,y).$$

Справедливость леммы 3.2 получаем непосредственно из утверждений (а), (б), (в) и (г).

Для нахождения точки минимума функции $\lambda(x)$ на множестве $\{x/\varphi(x) = 0\}$ также можно обойтись знанием одной лишь функции $\lambda'(x)$, как это утверждает следующая

Лемма 3.3. Если $M = 2\ell+1$, в точке x_0 функция $\lambda_t(x)$ достигает минимального значения на множестве $\{x/\varphi_t(x) = 0\}$, $\lambda_t'(x_0) = i$ и функция $\lambda_{t+1}^i(x)$ определена как:

$$\lambda_{t+1}^i(x) = \begin{cases} \lambda_{t+1}'(x) & \text{при } \lambda_{t+1}'(x) \ge i, \\ \lambda_{t+1}'(x) + (2\ell+1) & \text{при } \lambda_{t+1}'(x) < i, \end{cases}$$

то точки минимума функции $\lambda_{t+1}(x)$

на множестве $\{x/\varphi_{t+1}(x)=0\}$ совпадают с точками минимума функции $\lambda^i_{t+1}(x)$ на этом множестве.

В самом деле, пусть $x_1 \in \{x/\varphi_{t+1}(x)=0\}$. Из (I2) следует, что

$$\lambda_{t+1}(x_1)-\lambda_{t+1}(x_0)=(K_2'-K_1')(2\ell+1)+\lambda'_{t+1}(x_1)-\lambda'_{t+1}(x_0), \quad (I6)$$

где K_1' и K_2' – целые числа. Из (I6) и (I0), учитывая, что $\lambda'_{t+1}(x_0)=\lambda'_t(x_0)$, $0\le\lambda'_{t+1}(x_1)\le 2\ell$ и $0\le\lambda'_{t+1}(x_0)\le 2\ell$, получаем $K_2'-K_1'=0$ или $K_2'-K_1'=1$. Полагая последовательно $K_2'-K_1'=0$ и $K_2'-K_1'=1$, из (I6) и (I0) получаем:

д) выполняется одно из неравенств:

$$0\le\lambda'_{t+1}(x_1)-\lambda'_{t+1}(x_0)\le\ell, \quad (I7)$$

$$\lambda'_{t+1}(x_1)-\lambda'_{t+1}(x_0)\le-(\ell+1); \quad (I8)$$

е) если (I7), то $\lambda_{t+1}(x_1)-\lambda_t(x_0)=\lambda'_{t+1}(x_1)-\lambda'_{t+1}(x_0)$;

ж) если (I8), то $\lambda_{t+1}(x_1)-\lambda_t(x_0)=\lambda'_{t+1}(x_1)+(2\ell+1)-\lambda'_{t+1}(x_0)$.

Так как $\lambda^i_{t+1}(x_0)=\lambda'_t(x_0)=i$, то при $\lambda'_{t+1}(x_1)\ge i$ имеет место (I7), а при $\lambda'_{t+1}(x_1)<i$ имеет место (I8). Поэтому из утверждений (е) и (ж) следует, что

$$\lambda_{t+1}(x_1)-\lambda_{t+1}(x_0)=\lambda^i_{t+1}(x_1)-\lambda'_{t+1}(x_0). \quad (I9)$$

Из этого равенства справедливость леммы 3.3 вытекает непосредственно.

Из лемм 3.I, 3.2 и 3.3 следует корректность такого алгоритма A_4:

Настройка. Помечаем вершину a индексами $\lambda'(x)=0$ и $\varphi(a)=0$, а все остальные вершины x – индексами $\lambda'(x)=\infty$ и $\varphi(x)=1$. Полагаем $i=0$.

Основная часть.

I. Во множестве вершин, помеченных индексом $\varphi(x)=0$, выбираем одну из вершин x с минимальным значением функции:

$$\lambda^i(x)=\begin{cases}\lambda'(x) & \text{при } \lambda'(x)\ge i;\\ \lambda'(x)+(2\ell+1) & \text{при } \lambda'(x)<i.\end{cases}$$

У этой вершины признак $\varphi(x)=0$ заменяем признаком $\varphi(x)=1$. Если $x=b$, то переходим к обратному ходу.

2. Для выбранной вершины x ищем ребро (x,y) такое, что выполняется одно из условий (7), (8) или (9); затем заменяем индекс $\lambda(y)$ индексом

$$\lambda'(y)=\lambda'(x)+\lambda(x,y)(\mathrm{mod}\,(2\ell+1))$$

и индекс $\varphi(y)$ индексом $\varphi(y)=0$. Повторяем п.2, пока не установятся индексы на множестве $\Gamma(x)$. Полагаем $i=\lambda'(x)$ и переходим к п.I.

Обратный ход. Среди вершин множества $\Gamma(b)$ выбираем одну из вершин x, для которой $\lambda'(b)-\lambda'(x)=\lambda(x,b)$ или $\lambda'(b)-\lambda'(x)=\lambda(x,b)-(2\ell+1)$, и обозначаем выбранную вершину через x_1. Среди вершин множества $\Gamma(x_1)$ выбираем одну из вершин x, для которой $\lambda'(x_1)-\lambda'(x)=\lambda(x,x_1)$ или $\lambda'(x_1)-\lambda'(x)=\lambda(x,x_1)-(2\ell+1)$, и обозначаем выбранную вершину через x_2. Продолжаем выбор вершин, пока на каком-то $(k+1)$-м шаге не окажется выбранной вершина a. Вершины $[a,x_k,x_{k-1},\dots,x_1,b]$ и образуют кратчайшую цепь между a и b.

Применяя при программировании вместо символа ∞ число $2\ell+1$, имеем (для случая, когда длины ребер выражаются целыми числами):

$$V(A_4)=n\cdot(2+[\log_2(2\ell+1)])$$

Замечание. Единственное назначение индекса $\varphi(x)$ в алгоритмах A_3 и A_4 состоит в том, чтобы выделить вершины, принадлежащие фронту волны и тем самым избежать при работе алгоритма A_3 проверки условия (I) и при работе алгоритма A_4 проверки условий (7), (8) или (9) для всех вершин графа (X,Γ). Если не очень заботиться о сокращении времени работы алгоритма, то можно обойтись вовсе без индексов $\varphi(x)$. Покажем, например, как это сделать для алгоритма A_4. Пусть M_t – множество всех вершин x таких, что $\lambda_t(x)\ne\infty$ и для каждой из этих вершин существует смежная вершина y, удовлетворяющая одному из условий (7), (8) или (9). Тогда $\varphi_t(x)=0$ (при (7) или (8) это следует из (6) и леммы 3.I, при (9) – из того, что множества $\{x/\varphi_t(x)=-1\}$ и $\{x/\varphi_t(x)=1\}$ не смежны). Поэтому в алгоритме A_4 можно заменить поиск во множестве $\{x/\varphi(x)=0\}$ вершины с минимальным значением $\lambda^i(x)$ на этом множестве поиском во множестве M_t вершины с минимальным значением $\lambda^i(x)$ [x] на M_t. Обозначив полученный в результате этой замены алгоритм через A_5, в случае целочисленных ребер будем иметь:

$$V(A_5)=n\cdot(1+[\log_2(2\ell+1)]).$$

Последний результат является обобщением результата [4], где рассматривался случай $\ell=1$.

---

x) Используя равенство (I9), легко показать правомерность употребления в последнем случае функции $\lambda^i(x)$.

4. Уменьшение количества запоминаемых индексов $\lambda(x)$

Во многих случаях количество вершин, входящих во фронт волны, значительно меньше общего количества вершин графа (X,Γ). Тогда нецелесообразно при выполнении п. I алгоритма A_3' перебирать все вершины графа (X,Γ), чтобы найти вершины множества $\{x/\varphi_t(x)=0\}$ и среди этих вершин – вершину с минимальным значением $\lambda(x)$. Гораздо удобнее заранее иметь в памяти ЭВМ список всех вершин множества $\{x/\varphi_t(x)=0\}$ с указанием индексов $\lambda(x)$ для этих вершин. Если, кроме того, в процессе работы алгоритма некоторым ребрам графа (X,Γ) придавать ориентацию, которая обеспечит выполнение обратного хода, то все остальные индексы $\lambda(x)$ оказываются ненужными. В результате таких преобразований из алгоритма A_3' получим следующий алгоритм A_6:

Н а с т р о й к а. Помечаем вершину α индексом $\varphi(\alpha)=0$, остальные вершины x – индексом $\varphi(x)=1$. В список "фронт волны" (ФВ) включаем единственную строчку, содержащую координаты вершины α в графе (X,Γ) и индекс $\lambda(\alpha)=0$.

О с н о в н а я ч а с т ь.

I. В списке ФВ выбираем одну из вершин x с минимальным значением индекса $\lambda(x)$. В графе (X,Γ) индекс $\varphi(x)=0$ заменяем индексом $\varphi(x)=-1$. Если $x=\beta$, то переходим к обратному ходу.

2. В графе (X,Γ) для выбранной вершины x ищем ребро (x,y) такое, что выполняется одно из условий:

$$\varphi(y)=1, \qquad (20)$$

$$\varphi(y)=0 \quad \text{и} \quad \lambda(y)-\lambda(x)>\lambda(x,y). \qquad (21)$$

В случае (20) заменяем в графе (X,Γ) индекс $\varphi(y)=1$ индексом $\varphi(y)=0$ и пополняем список ФВ строчкой, содержащей координаты вершины y в графе (X,Γ) и индекс $\lambda(y)=\lambda(x)+\lambda(x,y)$; в случае (21) снимаем в графе (X,Γ) ориентацию с ребра, направленного из вершины y, и заменяем в списке ФВ индекс $\lambda(y)$ для вершины y индексом $\lambda(y)=\lambda(x)+\lambda(x,y)$. В обоих случаях придаем ориентацию от y к x ребру (x,y). Повторяем п. 2, пока во множестве $\Gamma(x)$ существуют вершины y, удовлетворяющие условиям (20) или (21). Затем вычеркиваем из списка ФВ строчку, соответствующую вершине x, и переходим к п. I.

О б р а т н ы й х о д заключается в движении из вершины β по направлению ориентированных ребер до тех пор, пока такое движение будет возможным. Путь, противоположный найденному, совпадает с кратчайшей цепью между α и β.

Корректность алгоритма A_6 очевидным образом следует из корректности алгоритма A_3'. В случае целочисленных ребер имеем:

$$V(A_6)=(n_1+1)(K+1+[\log_2(\ell(\alpha,\beta)+\ell)])+2n+2m,$$

где n_1 – количество вершин в максимальном фронте волны; K – максимальное количество двоичных разрядов, необходимое для записи координат в графе (X,Γ) вершины этого графа, m – количество ребер графа (X,Γ).

Нетрудно видеть, что в каждый момент работы алгоритма A_6 нужны не сами индексы $\lambda(x)$ для вершин фронта волны, а их разности. Поэтому можно пополнить п. I алгоритма A_6 таким преобразованием: для всех вершин z, принадлежащих фронту волны, индекс $\lambda(z)$ заменяем индексом $\lambda(z)=\lambda(z)-\lambda(x)$, где $\lambda(x)$ – минимальный из индексов $\lambda(z)$. Из (I0) следует, что тогда все индексы $\lambda(x)$ будут расположены в числовом сегменте $[0,\ell]$. Обозначив полученный алгоритм через A_7, для целочисленного случая имеем:

$$V(A_7)=(n_1+1)(K+1+[\log_2\ell])+2n+2m.$$

З а м е ч а н и е. Можно еще больше сократить память алгоритма поиска кратчайшей цепи, если не делать различия между направлением ребер (x,y) от x к y и направлением от y к x, а помечать используемые в процессе работы алгоритма ребра какой-либо единой меткой. Однако такое сокращение памяти приведет к некоторому усложнению алгоритма и увеличению времени его работы по сравнению со временем работы алгоритма A_7. Изменения, которые надо произвести в алгоритме A_7, заключаются в следующем:

Во-первых, ориентировку и снятие ориентировки с ребер графа (X,Γ) надо заменить, соответственно, простановкой на ребрах некоторой метки μ и снятием этой метки с ребер.

Во-вторых, в основной части алгоритма следует добавить 3-й пункт, который гласит: если при выполнении п. 2 хотя бы одно ребро было помечено меткой μ, то переходим к п. I; в противном случае после выполнения 2-го пункта двигаемся из вершины x по ребрам, помеченным меткой μ, до тех пор, пока это движение однозначно определено или пока не дойдем до вершины α, и одновременно стираем метку μ с пройденных ребер; затем переходим к п. I.

В-третьих, при осуществлении обратного хода следует выполнить из каждой вершины, отличной от β и принадлежащей фронту волны, движение по однозначно определенной цепи, состоящей из помеченных ребер и не проходящей ч е р е з вершину α, и одновременно стирать метки μ с пройденных ребер; после этого

кратчайшая цепь между вершинами α и β будет состоять из всех тех и только тех ребер, на которых сохранилась метка μ.

Обозначив построенный алгоритм через A_8, при целочисленных ребрах имеем:

$$V(A_8)=(n_1+1)\cdot(K+1+[\log_2 l])+2n+m .$$

5. Рекомендации по применению

В заключение дадим некоторые рекомендации по применению описанных в настоящей статье методов экономии памяти ЭВМ при построении кратчайшей цепи между заданными вершинами α и β.

В разделе 3 (алгоритмы A_4 и A_5) экономия памяти достигается за счет индексации по модулю $2l+1$; в разделе 4 (алгоритмы A_6, A_7 и A_8) – за счет ориентировки некоторых ребер и использования списка ФВ. Нетрудно видеть: если отношение $\frac{n}{n_1}$ невелико, то целесообразно воспользоваться одним из алгоритмов A_3, A_4 или A_5, если $\frac{n}{n_1}$ велико – то одним из алгоритмов A_6, A_7 или A_8.

Если в нашем распоряжении вполне достаточно памяти ЭВМ, чтобы отвести для каждого индекса $\lambda(x)$ $1+[\log_2(l(\alpha,\beta)+l)]$ разрядов, то следует применить алгоритм A_3 или A_6. Если памяти мало, то лучше применить A_4 или A_7. При весьма жестких ограничениях памяти приходится применять алгоритм A_5 или A_8. Но в последнем случае некоторая дополнительная экономия памяти достигается за счет значительного увеличения времени работы программы, и поэтому по возможности следует избегать применения алгоритмов A_5 и A_8. Эти алгоритмы представляют интерес скорее лишь потому, что они показывают, какой предельной экономии памяти можно добиться при каждом из двух рассматриваемых в настоящей статье подходов.

Если отношение $\frac{n}{n_1}$ невелико, то логически возможен также третий подход – применить ориентировку ребер без использования списка ФВ. Применение ориентировки ребер позволит уменьшить величину максимального индекса $\lambda(x)$ до l, а также совместить в памяти ЭВМ место записи индексов $\varphi(x)$ с местом записи индексов $\lambda(x)$: например, для пометки индексом $\varphi(x)=1$ можно использовать число $L+1$, где L – максимально возможное значение индекса $\lambda(x)$, для пометки индексом $\varphi(x)=-1$ – число $L+2$ и для пометки индексом $\varphi(x)=$ – числа $\lambda(x)$. Благодаря этому, алгоритмы A_6, A_7 и A_8 были бы очевидным образом преобразованы в некоторые алгоритмы A_6', A_7' и A_8', для которых в целочисленном случае справедливы следующие равенства:

$$V(A_6')=n\cdot(1+[\log_2(l(\alpha,\beta)+l+2)])+2m,$$
$$V(A_7')=n\cdot(1+[\log_2(l+2)])+2m,$$
$$V(A_8')=n\cdot(1+[\log_2(l+2)])+m .$$

Нетрудно видеть: если граф (X,Γ) не является деревом, то

$$V(A_3)<V(A_6'),V(A_4)\leq V(A_7'),V(A_5)\leq V(A_8') \qquad (22)$$

Докажем, например, последнее из этих неравенств:

$$V(A_8')-V(A_5)=n\cdot([\log_2(l+2)]-[\log_2(2l+1)])+m=$$
$$=n([\log_2(l+2)]-[\log_2(l+1/2)])+(m-n)\geq 0 ,$$

т.е. $V(A_5)\leq V(A_8')$, что и требовалось доказать. Из (22) следует, что с точки зрения экономии памяти применение ориентировки ребер представляется нецелесообразным, если её не сочетать с использованием списка ФВ.

Литература

1. К. БЕРЖ. Теория графов и ее применения. ИЛ., 1962.
2. C.Y.LEE. An algorithm for path connections and its applications. IRE Trans., 1961, EC-10, N 3, p.346-366.
3. Ю.Л. ЗИМАН, Г.Г. РЯБОВ. Волновой алгоритм и электрические соединения. М., ИТМ и ВТ АН СССР, 1965.
4. В.А. ТЮРЕНКОВ. Алгоритм нахождения кратчайшего пути. – Вычислительные системы, Новосибирск, 1963, вып. 6, стр. 41-44.
5. А.В. БУТРИМЕНКО. О поиске кратчайших путей по графу при его изменениях. Изв. АН СССР, сер. технич. киберн., 1964, № 6, 55-58.

Поступила в редакцию
21.I.1968 г.