УДК 004.421.2 : 519.178

Винничук С.Д.

АНАЛІЗ ЧАСОВОЇ СКЛАДНОСТІ КРОКІВ АЛГОРИТМУ КРУСКАЛА ПРИ ВІДСОРТОВАНИХ ВАГАХ РЕБЕР З ЧАСОМ РОБОТИ O(*E*+*V*log*V*) ДЛЯ ВАРІАНТУ ЙОГО *C*-РЕАЛІЗАЦІЇ

Анотація:

Запропоновано С-реалізацію алгоритму Крускала з часом роботи O(E+VlogV) для випадку відсортованих ваг ребер з детальним аналізом часової складності його кроків.

Аннотация:

Предложена С-реализация алгоритма Крускала со временем работы O(E+VlogV) для случая отсортированных весов ребер с детальным анализом временной сложности его шагов.

Abstract:

Proposed C-implementation of Kruskal's algorithm with running time O (E + VlogV) in the case of sorted weights of the edges with a detailed analysis of the time complexity of its steps.

Постановка задачі

У задачі пошуку мінімального остовного дерева відомими вважаються множини вершин і ребер неорієнтованого графа $G(V,E)$ (V – множина вершин, E – множина ребер), а також вага ребер, де значення ваги є дійсним числом. Існує ряд алгоритмів пошуку мінімального остовного дерева (Крускала, Пріма, Борувки), що використовують ідею пошуку безпечних ребер (див. [1], разд.23). У загальному випадку зв'язного графа обчислювальна складність жадібного алгоритму Крускала при відсортованих вагах ребер, згідно з [1], визначається способом реалізації структури даних для множин, що не перетинаються, і, як відмічається в роботі [1], для асимптотично найшвидшої відомої реалізації алгоритму є величиною порядка $O(E\log V)$.

При програмній реалізації алгоритму Крускала важливо не тільки використати один з ефективних алгоритмів реалізації задачі, але й вміти обґрунтовувати його ефективність. І в першу чергу це важливо при вивченні чисельних методів та методів алгоритмізації. Для таких цілей в роботі для задачі побудови остовного дерева мінімальної ваги за методом Крускала при відсортованих вагах ребер проводиться детальний аналіз часової складності всіх кроків алгоритму та дається загальна оцінка його складності.

1. Алгоритм Крускала та загальна оцінка його складності

Нехай однозв'язний зважений граф задається списком інцидентності і для кожного ребра задано його довільна невід'ємна вага.

Робота алгоритму Крускала (для випадку відсортованих у порядку зростання ваг ребер) може бути описана наступною послідовністю кроків:

A1. Формируется множина V дерев (ліс), кожне з яких містить всього одну вершину. Число ребер, включених в остовне дерево, що будується, k=0;

А2. В циклі по числу ребер вибираються ребра з вагою, що збільшується, та проводиться аналіз:

А2.1. Якщо вершини (*u,v*), що описують ребро, належать одному дереву, то ребро виключається з розгляду. Перехід до п. А.2.

А2.2. Якщо вершини (*u,v*), що описують ребро, належать двом різним деревам, то ребро додається в остовне дерево. $k = k + 1$. Перехід до п.А3.

А3. Два дерева, яким належать вершини *u* та *v*, об'єднуються в одне.

А4. Якщо $k = V$-1, то побудова остовного дерева завершена, а інакше перехід до п.А2.

На кроці *A2* організується цикл по числу ребер, в ході реалізації якого формується остовне дерево. При цьому для кожного з ребер можуть виконуватися або операції блоків *A2.2*, *A3* та *A4* (*V*-1 раз), або операції блоку *A2.1*, число яких не перевищує *E-V+1*. Тому для сумарного числа елементарних операцій *N*, що виконуються на кроках *A1*, *A2*, *A2.1*, *A2.2*, *A3* та *A4*, тобто величин N_{A1}, N_{A2}, $N_{A2.1}$, $N_{A2.2}$, N_{A4} можно записати наступну верхню оцінку

$$N \le N_{A1} + N_{A2} + (V-1)\cdot(N_{A2.2} + N_{A3} + N_{A4}) + (E-V+1)\cdot N_{A2.1}, \qquad (1)$$

де N_{A1}, N_{A2}, $N_{A2.1}$, $N_{A2.2}$, N_{A3}, N_{A4} - число елементарних операцій для відповідного кроку алгоритму.

Відповідно для N_{A1}, N_{A2}, $N_{A2.1}$, $N_{A2.2}$, N_{A4} справедливі наступні оцінки.

На кроці *A1* кожен із вузлів, число яких *V*, формує дерево, що містить всього один цей вузол, і дереву присвоюється унікальний номер, наприклад, номер вузла. Тому в ході виконання такого кроку здійснюється всього *V* присвоювань, тобто $N_{A1} = V$. Якщо позначити масив номерів дерев через *nd*[*V+1*] (для кожного вузла вказується номер дерева, до якого він відноситься), то на мові *C* кроку А1 відповідатиме наступний програмний блок

```
for (m=1; m<=V; m++)  nd[m] = m; ;   //блок А1.
```

На кроці *A2* організується цикл по числу ребер, в ході реалізації якого формується остовне дерево. При цьому для кожного з ребер можуть виконуватися або операції блоків *A2.2*, *A3* та *A4* (всього *V*-1 раз), або операції блоку *A2.1*, число яких не перевищує *E-V+1*. Якщо позначити масив *MI*[*E*][3] – список інцидентності (*MI*[*i*][0] – вузол початку гілки, *MI*[*i*][1] – вузол кінця гілки, *MI*[*i*][2] – вага гілки), де дані про ребра записано в порядку зростання їх ваг, то на мові *C* цьому алгоритму відповідатиме наступний програмний блок:

```
k=0;
for (m=0; m<E; m++)  //блок А2
{ u=MI[m][0]; v=MI[m][1];
        if ( nd[u] == nd[v] ) //блок А2.1 та //блок А2.2
                        continue; //блок А2.1
                else
                {               k++; //блок А2.2
                                Об'єднання двох дерев; //блок А3
                        if ( k==V-1 ) break; //блок А4
        }
}
```

При аналізі наведеного програмного блоку легко замітити, що число операцій для блоків *A2.1*, *A2.2* та *A4* є постійним ($N_{A2.1}$ = O(1); $N_{A2.2}$ = O(1); N_{A4} = O(1)). Для найгіршого випадку, коли вихід із циклу (блок А4) реалізується тільки при m=E, матимемо N_{A2} =E, включаючи і визначення вузлів початку та кінця ребра. Тому для загального числа операцій N отримаємо:

$$N \le V + E + (V-1)\cdot(O(1) + N_{A3} + O(1)) + (E - V + 1)\cdot O(1) = (V-1)\cdot N_{A3} + E\cdot O(1) \quad (2)$$

Із співвідношення (2) випливає, що для оцінки складності алгоритму необхідно реалізувати процедуру злиття дерев та оцінити її складність. Тому розглянемо структуру інформації про дерева лісу та способи їх злиття.

2. Структура даних про ліс дерев і процедура злиття двох дерев

Для кожного з дерев лісу передбачимо формування наступних даних:

- масив *cv*[*V*] - число вузлів в дереві;
- масив *pv*[*V*] - послідовність вузлів дерева;
- масив *pvk*[*V*] - номер кінцевого вузла в списку вузлів дерева,

де для *i*-го дерева в масиві *pv*[*i*] задано номер першого вузла, добавленого в дерево до *i*-го вузла. Нехай це вузол *i*1 = *pv*[*i*]. Тоді *i*2 = *pv*[*i*1], *i*3 = *pv*[*i*2], *i*4 = *pv*[*i*3] і т.д. Для останнього ж з вузлів, добавлених в дерево (а саме вузла з номером *pvk*[*i*]]) *pv*[*pvk*[*i*]] = 0. Така організація списку вузлів відома як направлений список елементів множини.

Із врахуванням наведених описів масивів опишемо алгоритм блоку А3 об'єднання двох дерев (опис блоку та його програмна реалізація на мові *C*), для яких ребро *m*, що додається до остовного дерева описується вузлами u=MI[m][0]; v=MI[m][1].

А3.1. Визначаємо дерево с номером *d*1, що містить більше число вузлів та дерево з номером *d*2, що містить менше число вузлів:

d1=u; d2=v; if (cv[u] < cv[v]) { d1=v; d2=u; }

А3.2. Обчислюємо число вузлів в новому об'єднаному дереві з номером *d*1:

cv[d1] = cv[u]+cv[v];

А3.3. Формуємо дані про нове об'єднане дерево:

А3.3.1. Формуємо новий направлений список вузлів для об'єднаного дерева з номером *d*1, а саме: всі вузли дерева з номером *d*2 розміщуємо після вузлів з номером *d*1 в тому ж порядку, в якому вони були в дереві з номером *d*2:

pv[pvk[d1]] =d2;

А3.3.2. Всім вузлам дерева з номером *d*2 присвоюємо значення номера дерева *d*1:

j = d2; for (m=1; m<=cv[d2]; m++) { nd[j] = d1; j = pv[j]; }

Необхідно відмітити, що для формування направленого списку вузлів у блоці А3.3.1 достатньо всього одного оператора присвоювання, оскільки для цього слід тільки реалізувати посилання з останнього в списку вузла дерева з номером *d*1 на перший у списку вузол дерева *d*2, що і реалізується відповідним оператором присвоювання.

3. Оцінка складності процедури злиття дерев

Із наведеного вище опису блоків та їх програмної реалізації на мові С видно, що при програмній реалізації блоків А3.1, А3.2 и А3.3.1 використовується скінчене число операцій. Тому при кожному злитті двох дерев матимемо:

$$N_{A3.1} = N_{A3.2} = N_{A3.3.1} = O(1). \quad (3)$$

Тому для оцінки складності всього алгоритму слід оцінити складність блоку А3.3.2, а саме визначити число всіх присвоювань номера дерева вузлам графа при V-1 - ому об'єднанні дерев лісу. Розглянемо ряд варіантів об'єднань двох дерев.

1. Нехай на деякому з кроків об'єднання дерев обидва дерева містять число вузлів більше за 1, де виконується умова **cv[d1] ≥ cv[d2].**

Тоді у дереві, що буде об'єднанням дерев з номерами $d1$ та $d2$, загальне число вузлів стане рівним cv[u]+cv[v], серед яких для cv[d2] вузлів слід буде змінити номер дерева, якому вони належать. При цьому очевидно, що при одному і тому ж числі вузлів cv[u]+cv[v] у об'єднаному дереві максимальне число операцій переприсвоювання номера дерева буде у випадку cv[d1] = cv[d2].

2. Розглянемо тепер найтяжчий випадок, тобто випадок максимального числа переприсвоювань номера вузла в процесі V-1 - ого об'єднання дерев. Нехай число вузлів графа $V = 2^t$ і при кожному об'єднанні двох дерев виконується умова cv[$d1$] = cv[$d2$]. Тоді при останньому V-1 - ому об'єднанні обидва дерева містять 2^{t-1} вузлів і число переприсвоювань номера вузла в блоці А3.3.2 дорівнюватиме 2^{t-1}. Дерева лісу, що містять 2^{t-1} вузлів (їх два), отримані при об'єднанні дерев, які містять 2^{t-2} вузлів і число переприсвоювань номера вузла в блоці А3.3.2 дорівнюватиме $2 \cdot 2^{t-2} = 2^{t-1}$. Відповідно дерева лісу, що містять 2^{t-2} вузлів (їх чотири), отримані при об'єднанні дерев, які містять 2^{t-3} вузлів і число переприсвоювань номера вузла в блоці А3.3.2 дорівнюватиме $4 \cdot 2^{t-3} = 2^{t-1}$. І так далі. Тому загальне число переприсвоювань номера вузла в блоці А3.3.2 дорівнюватиме $(t\text{-}1) \cdot 2^{t-1} = V/2 \cdot \log_2 V = O(V \log V)$.

Отримана оцінка для максимального числа операцій в блоці А3.3.2 дозволяє визначити складність операцій V-1 - ого злиття дерев та оцінити складність наведеного алгоритму в цілому. Із врахуванням співвідношення (3) для V-1 - ого використання блоку А3 (V-1 - е злиття дерев) матимемо

$$\begin{aligned}(V-1)\cdot N_{A3} &= (V-1)\cdot (N_{A3.1} + N_{A3.2} + N_{A3.3.1}) + (V-1)\cdot N_{A3.3.2} = \\ &= (V-1)\cdot (O(1)+O(1)+O(1)) + O(V\cdot \log V) = O(V\cdot \log V)\end{aligned} \quad (4)$$

Відповідно з врахуванням співвідношення (2) оцінимо складність наведеного алгоритму в цілому.

$$N \le (V-1)\cdot N_{A3} + E\cdot O(1) = O(V\cdot \log V) + O(E) = O(E + V\log V)\,. \quad (5)$$

Висновки

Запропоновано простий для реалізації алгоритм методу Крускала побудови остовного дерева мінімальної ваги. Проведено детальний аналіз часової складності всіх кроків описаного алгоритму та дається загальна оцінка його складності. Показано, що для наведеного алгоритму часова складність оцінюється величиною $O(E+V\log V)$, що дещо краще за оцінку $O(E\log V)$, наведену в роботі [1], але для невідсортованих ваг ребер співпадає з нею.

Література:

1. *Кормен Т.* Алгоритмы: построение и анализ / [Кормен Т., Лейзерсон Ч., Риверст Р., Штайн К.]. – [2-е изд.]. – М. : Издательский дом "Вильямс", 2011. – 1296 с.