ISSN 2078–5097. Вісн. Львів. ун-ту. Сер. прикл. матем. та інф. 2015. Вип. 23. С. 119–124
Visnyk of the Lviv University. Series Appl. Math. and Informatics. Issue 23. P. 119–124

УДК 004.02+004.021+519.17

ОБЧИСЛЮВАЛЬНА СКЛАДНІСТЬ ДЕЯКИХ АЛГОРИТМІВ НА ГРАФАХ

В. Черняхівський

Львівський національний університет імені Івана Франка,
вул. Університетська, 1, Львів, 79000, e-mail: v_chrn@franko.lviv.ua

Розглянуто задачу обчислення складності алгоритму будови максимального простого ланцюга графа. Викладено теоретичні міркування щодо обчислення складності для рекурсивного алгоритму будови максимального ланцюга. Опрацьовано метод обчислення кількості операцій, потрібних для будови ланцюгів. Зроблено висновки щодо складності алгоритму. Викладено результати практичного обчислення складності для тестових графів.

Ключові слова: граф, простий ланцюг, максимальний ланцюг, алгоритм, складність.

1. ВСТУП

Розглядаємо клас задач, модель яких можна подати у вигляді неповного графа. Потрібно відшукати максимальний простий ланцюг між двома довільними заданими вершинами V_a і V_b. Для практичних потреб часто виникає задача побудови максимального ланцюга графа, щоб отримати якнайбільше покриття, тобто критерієм максимізації є кількість включених до ланцюга вершин: $V_a - V_1 - V_2 - \ldots - V_k - V_b,\ k \to \max$. Розв'язання такої пошукової задачі виконує алгоритм, викладений в [1]. На підставі цього алгоритму розраховуємо обчислювальну складність без обмежень на включення вершин. Обчислювальну складність визначаємо відомим способом [2] як порядок функції $\lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = const \neq 0$.

2. ФОРМУЛЮВАННЯ ЗАДАЧІ

Заданий неорієнтований неповний граф $G = (V, E)$ з ребрами однакової ваги або незважений. Граф зв'язний, якщо незв'язний, то розглядаємо задачу лише для однієї окремої компоненти зв'язності. Побудувати простий ланцюг максимальної довжини, який сполучає дві задані вершини $V_a, V_b : (V_a, V_1, V_2, \ldots, V_k, V_b),\ k \to \max$, де $V_i \neq V_j$ при $i \neq j$, крім V_a і V_b. Ребра визначають всі пари вершин $(V_k, V_m),\ (k, m) \in \{1, 2, \ldots, n, n = |V|\}, k \neq m$, сполучених між собою безпосередньо. Рух між кожною визначеною парою сусідніх вершин двосторонній в обидвох напрямах: $(V_k, V_m) = (V_m, V_k),\ d(V_k, V_m) = d(V_m, V_k)$. Граф не має петель (V_p, V_p), якщо такі є, то їх викреслюємо і в розв'язуванні задачі вони участі не беруть.

В [1] запропоновано базовий алгоритм P_0^r відшукання максимального простого ланцюга неповного графа, який побудовано на підставі методу пошуку з поверненнями і як рекурсивний.

Для алгоритму $\mathrm{P}_0^{\mathrm{r}}$ треба отримати теоретичну та практичну оцінку обчислювальної складності [2] для визначення способів застосування.

3. АНАЛІЗ ОБЧИСЛЮВАЛЬНОЇ СКЛАДНОСТІ І ТЕОРЕТИЧНІ ОЦІНКИ

Задано деякий граф на рис.1. Позначимо S_b^a шуканий максимальний простий ланцюг між двома вершинами V_a і V_b. Будемо шукати S_{15}^1. Будуємо граф так, щоб всі вершини мали однаковий заданий степінь – у цьому випадку 4. Одна або декілька останніх в процесі будови графа вершин можуть мати менший степінь. У цьому разі для розрахунків можна прийняти, що всі вершини мають такий самий заданий степінь – це не буде зменшувати розрахункову складність алгоритму.

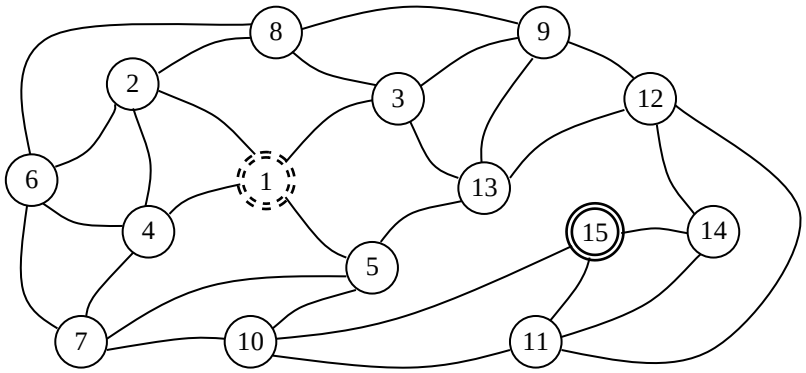

Рис.1. Приклад графа для пошуку S_{15}^1

Рекурсивний алгоритм $\mathrm{P}_0^{\mathrm{r}}$ виконує перегляд усіх можливих простих ланцюгів, які починаються у вершині V_a і за один крок рекурсії будує одну наступну ланку простого ланцюга, починаючи з вершини *i*: $\mathrm{P}_0^{\mathrm{r}}(i) \to j$, якщо це можливо, і сприяє рекурсивно $\mathrm{P}_0^{\mathrm{r}}(j)$. На кожному кроці рекурсії переглядають список усіх можливих продовжень ланцюга. Якщо ж нову ланку неможливо побудувати, то крок алгоритму не виконує ніяких операцій. Отже, $\mathrm{P}_0^{\mathrm{r}} = \mathrm{P}_0^{\mathrm{r}}(\mathrm{P}_0^{\mathrm{r}}(\mathrm{P}_0^{\mathrm{r}}(\ldots (\mathrm{P}_0^{\mathrm{r}}(a)) \ldots)))$. Побудова чергової ланки $\mathrm{P}_0^{\mathrm{r}}(i) \to j$ супроводжується викресленням вершини *i* та всіх інцидентних їй ребер з графа. Незалежно від можливості продовження ланцюга алгоритм повертається (кінець кроку рекурсії – пошук з поверненнями) до попередньої вершини, тому вершина *i* та інцидентні їй ребра знову стають доступними для включення до ланцюга.

Після переходу з вершини 1 до однієї з суміжних вершин граф розпадається на 4 підграфи: S_{15}^2, S_{15}^3, S_{15}^4, S_{15}^5. На рис.2 показано один з них – S_{15}^2. Решта мають таке саме топологічне зображення $G_1 = G(VG \setminus 1)$.

Будемо обчислювати кількість операцій як кількість переходів по графу за ребрами для будови всіх можливих простих ланцюгів при виконанні $\mathrm{P}_0^{\mathrm{r}}$.

Позначимо k_i кількість можливих операцій переходу до наступної вершини на кроці *i*. Кількість операцій дорівнює степені вершини. На початку $k_1 = 4$. Кількість операцій переходу до наступної вершини на другому кроці k_2, а загальна кількість переходів після двох кроків $F = k_1 \times k_2$.

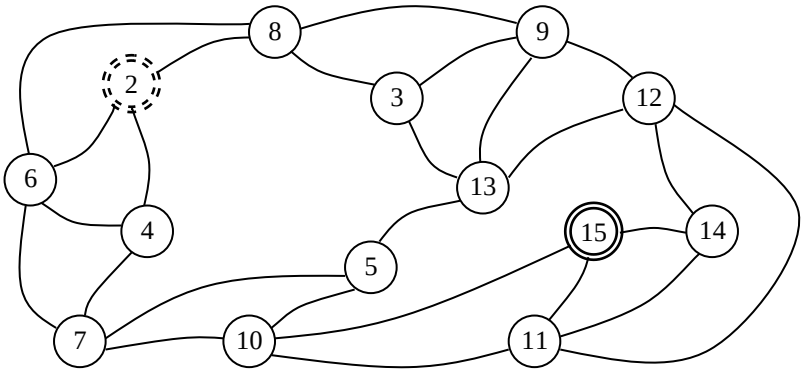

Рис. 2. Граф S_{15}^2

Приймемо такі допущення: 1) вдається побудувати максимальний ланцюг, який включає всі вершини графа (гамільтоновий шлях) – це максимальний випадок; 2) загальна кількість ребер графа, які залишаються по мірі просування побудованою частиною ланцюга, зменшується пропорційно до зменшення кількості ще не включених вершин і зменшення степенів не включених вершин; 3) у зв'язку з викресленням інцидентних ребер на кожному переході до суміжної вершини графа середня степінь вершин залишкового підграфа зменшується по мірі просування ланцюгом пропорційно до кількості залишених вершин (рис.2).

Нехай заданий граф має *n* вершин степеня *k*. Тоді загальна кількість ребер $R \le \frac{n \times k}{2}$, приймемо $R = \frac{n \times k}{2}$. Загальна кількість переходів F_n^k становитиме

$$F_n^k = k_1 \times k_2 \times k_3 \times \ldots \times k_{n-1}$$

де k_j – розраховані коефіцієнти середньої степені вершин на кроці *j*. Зауважимо, що завжди $k_j \ge 1$ для $k > 2$. Оцінку r_i кількості ребер графа, які залишаються на *i*-му кроці просування побудованою частиною ланцюга, можна визначити такими рекурентними формулами:

$$r_1 = R;\ r_2 = r_1 - 2 \times \frac{r_1}{n};\ r_3 = r_2 - 2 \times \frac{r_2}{n-2} + 1;$$

$$r_i = r_{i-1} - 2 \times \frac{r_{i-1}}{n-i+1} + 1,\ i = \overline{4, n-1}$$

Для прикладу графа S_{15}^1 на рис.1 ($n = 15,\ k = 4$) маємо такі оцінки r_i, $i = \overline{1,14}$ з точністю 0.01:

30.00, 26.00, 23.00, 20.17, 17.50, 15.00, 12.67, 10.50, 8.50, 6.67, 5.00, 3.50, 2.17, 1.00.

Оцінки є дробовими числами, у випадку прямого використання потрібно заокруглити отримані значення до більшого цілого.

Тепер коефіцієнти k_j визначити просто: $k_j = 2 \times \frac{r_j}{n-j+1}$, $j = \overline{1, n-1}$. Чисельна перевірка розрахункових коефіцієнтів k_j з точністю 0.01 дає такі результати:

1) при $n = 15,\ k = 4$:

4.00, 3.71, 3.54, 3.36, 3.18, 3.00, 2.81, 2.62, 2.43, 2.22, 2.00, 1.75, 1.44, 1.00

2) при $n = 24,\ k = 6$:

6.00, 5.74, 5.55, 5.35, 5.16, 4.96, 4.77, 4.57, 4.38, 4.18, 3.98, 3.78, 3.58, 3.38, 3.17, 2.96, 2.75, 2.53, 2.30, 2.06, 1.79, 1.46, 1.00

Знову ж таки, коефіцієнти є дробовими числами, а не цілими, бо ми їх використовуємо для оцінювання, а не прямого використання.

Приклади обчислень кількості переходів F_n^k для випадку гамільтонового шляху, тобто максимальні оцінки

$$F_{15}^4 = 340029,\ F_{20}^4 = 55374618,\ F_{30}^4 = 1770172014073 > 10^{12}$$

$$F_{15}^6 = 14341900,\ F_{20}^6 = 8654607161 > 10^9,\ F_{30}^6 = 3791637540449828 > 10^{15}.$$

Нехай $k = const$. Тоді $R = \frac{n \times const}{2} = c_1 n,\ c_1 = const$. Звідси випливає

$$r_1 = c_1 n$$

$$r_2 = r_1 - 2\frac{r_1}{n} = c_1 n - 2\frac{c_1 n}{n} = c_1 n - 2c_1 = c_1(n-2) = c_2 n$$

$$r_3 = r_2 - 2\frac{r_2}{n-2} + 1 = c_3 n$$

$$r_j = c_j n,\ c_j = const$$

$$k_j = 2\frac{r_j}{n-j+1} = 2c_j \frac{n}{n-j+1} = const.$$

Оскільки k_j є константами, то приймемо k_j такими, що дорівнюють середньому арифметичному $AG(k_j)$ обчислених попередньо значень. Очевидно, що $AG(k_j) < k$. Отже,

$$F_n^k = \prod_{j=1}^{n-1} k_j = (AG(k_j))^{n-1} < const^{n-1}.$$

Висновок 1. Обчислювальна складність алгоритму $\mathrm{P}_0^{\mathrm{r}}$ при фіксованих степенях вершин k є показниковою функцією від кількості вершин n – експоненціальна складність.

Якщо ж $n = const$, а змінюється k – степінь вершин $(k \le n-1)$, тоді $R = \frac{const \times k}{2}$, і маємо аналогічні формули для r_j, в яких замість n підставлено k

$$R = \frac{const \times k}{2} = c_1 k$$

$$r_j = c_j k, \quad c_j = const$$

$$k_j = 2\frac{r_j}{n-j+1} = 2k\frac{c_j}{n-j+1} = 2k\frac{const}{const-j+1} = k \times const_1$$

$$F_n^k = \prod_{j=1}^{n-1} k_j = (k \times const_1)^{const_2} < C_{\max} \times k^{const_2}.$$

Висновок 2. Обчислювальна складність алгоритму $\mathrm{P}_0^{\mathrm{r}}$ при фіксованій кількості вершин n є поліноміальною функцією від степенів вершин k (поліноміальна складність).

4. РЕЗУЛЬТАТИ ОБЧИСЛЮВАЛЬНИХ ЕКСПЕРИМЕНТІВ

У табл. записані результати обчислень F_n^k для тестових графів для $n = 10, 15, 20, 25, 30, 35$ при $k = 3, 4, 5, 6, 7$. Тестові графи були побудовані так, щоб всі вершини мали однакову степінь k. Для кожного графа F_n^k проведена серія обчислень кількості переходів за ребрами будови ланцюга між різними парами вершин, після чого обрано максимальне значення. Переходи обчислювала програмна реалізація алгоритму $\mathrm{P}_0^{\mathrm{r}}$ шляхом включення окремих лічильників.

Результати обчислень кількості переходів за ребрами

n	k					$O(*)$
	3	4	5	6	7	
10	293	2245	10773	39985	125953	$O(k^8)$
15	1982	64794	892686	6736461	41087617	$O(k^{13})$
20	17836	1736621	54446374	958340635	10721147643	$O(k^{16})$
25	130407	41718515	3674429054	128251903770		$O(k^{21})$
30	921864	861151084	201235948570			$O(k^{25})$
35	3906270	18518539582				$O(k^{30})$
$O(*)$	$O(1{,}6^{n-1})$	$O(2^{n-1})$	$O(2{,}5^{n-1})$	$O(3^{n-1})$	$O(3{,}5^{n-1})$	

Останній рядок і останній стовпець таблиці є верхньою оцінкою фактичної обчислювальної складності як порядок функції кількості переходів за ребрами.

5. ВИСНОВКИ

1. Мета такої праці – визначити метод і загальні теоретичні оцінки складності алгоритму P_0^r та експериментальна перевірка. Наведені результати обчислювальної складності алгоритму P_0^r є верхньою оцінкою – для випадку повного пошуку за графом без обмежень. Теоретичні та експериментальні результати є відповідними. Обчислювальну складність можна зменшити завдяки оптимізаційним процедурам пошуку [2], коли повний пошук зайвий.

2. У базовому варіанті прямого розв'язку задача пошуку простого максимального ланцюга графа залишається NP-повною проблемою, яка сьогодні вирішена поліноміально лише для декількох класів графів [3].

3. Отримані результати є основою для порівняльної характеристики оптимізаційних і конструктивних вдосконалень алгоритму P_0^r .

4. Реальні графові моделі практичних задач мають степені вершин головно 3-4, тому алгоритм P_0^r можна застосувати безпосередньо до певних класів практичних задач.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. *Черняхівський В.В.* Рекурсивний алгоритм побудови максимального простого ланцюга неповного графа / В.В. Черняхівський // Вісн. Львів. ун-ту. Сер. прикл. матем. та інформ. – 2007. – Вип. 13. – С. 45-50.
2. *Goodman S.E.* Introduction to the design and analysis of algorithms / S.E. Goodman, S.T. Hedetniemi. – McGraw-Hill Book Company, 1977.
3. *Keshavarz-Kohjerdi F.* A linear-time algorithm for the longest path problem in rectangular grid graphs / F. Keshavarz-Kohjerdi, A. Bagheri, A. Asgharian-Sardroud // Discrete Applied Mathematics. – 2012. – Vol. 160, Is. 3. – P. 210-217.

Стаття: надійшла до редколегії 02.09.2015
доопрацьована 14.10.2015
прийнята до друку 28.10.2015

COMPUTATIONAL COMPLEXITY OF SOME ALGORITHMS ON GRAPHS

V. Chernyakhivskij

Ivan Franko National University of Lviv,
Universytetska Str., 1, Lviv, 79000, e-mail: v_chrn@franko.lviv.ua

The problem of computation complexity of the algorithm as simple chain structure graph is described. Theoretical considerations regarding the complexity of calculation algorithm for recursive structure maximum chain are presented. The method of calculating the number of operations required for the service of chains is shown. Conclusions are made on the complexity of the algorithm. The results of the practical difficulties for the calculation of test graphs are presented.

Key words: graph, simple chain, maximum chain, algorithm, complexity.