УДК (004)07

А. І. Паўлоўскі, В. К. Панамарэнка

ВЫЛІЧАЛЬНАЯ СКЛАДАНАСЦЬ АЛГАРЫТМАЎ І РЭКУРЭНТНЫЯ СУАДНОСІНЫ

Як вядома, адну і тую ж задачу могуць правільна рашаць шмат алгарытмаў. Але які з іх найбольш эфектыўны? Для адказу на гэтае пытанне існуе паняцце «*вылічальная складанасць*».

Звычайна пад вылічальнай складанасцю разумеюць колькасць аперацый, якая патрэбна алгарытму для яго выканання. Больш дакладна, не саму колькасць аперацый для таго ці іншага набору ўваходных даных, а залежнасць колькасці аперацый алгарытму ад памеру ўваходнай інфармацыі. Такім чынам, вылічальная складанасць часцей за ўсё — гэта часавая складанасць, якая вымяраецца ў своеасаблівых адзінках — аперацыях. Пры гэтым па вылічальнай складанасці заўсёды параўноўваюцца алгарытмы для рашэння адной і той жа задачы і ніколі яны не параўноўваюцца для рашэння розных задач. Разам з тым, вызначаючы эфектыўнасць алгарытмаў, у першую чаргу, па затратах часу, у некаторых выпадках трэба ўлічваць і тое, колькі памяці патрэбна таму ці іншаму алгарытму для яго выканання, інакш кажучы, улічваць складанасць алгарытму па затратах памяці [1].

Дакладнае веданне колькасці аперацый, што выкананы алгарытмам, не мае істотнага значэння. Галоўнае тое, як хутка расце гэты лік пры павелічэнні памеру ўваходных даных. Так, калі вылічальная складанасць аднаго алгарытму рашэння некаторай задачы ёсць $T_1(n) = n+n*\log_2 n$, а другога — $T_2(n) = n^2$, відавочна, што першы алгарытм лепш за другі таму, што з ростам n $T_2(n)$ расце хутчэй, чым $T_1(n)$. Для кожнай функцыі f(n), што задае складанасць алгарытму, можна вызначыць тры класы функцый:

Ω(f) (Амега вялікае) — клас функцый, якія растуць не менш хутка, як f ;

O(f) (Омікрон вялікае) — клас функцый, якія растуць не больш хутка, як f;

Θ(f) (Тэта вялікае) — клас функцый, якія растуць з той жа хуткасцю, што і f.

Першы і трэці з гэтых класаў менш значныя ў тэорыі вылічальнай складанасці. Першы з іх таму, што алгарытмы, якім адпавядаюць функцыі гэтага класа, маюць большую ці, у лепшым выпадку, такую ж складанасць, як і алгарытм, якому належыць функцыя f. Трэці — таму, што згодна з вызначэннем класа Θ алгарытмы, якія з ім звязаны, не могуць быць лепш за алгарытм, якому належыць функцыя f. іншая справа клас O(f). Для функцый гэтага класа f утварае верхнюю мяжу і таму сярод алгарытмаў, адпаведных функцыям класа, можа існаваць алгарытм з функцыяй складанасці меншай за f.

Рэкурэнтныя суадносіны ў час аналізу алгарытму ўзнікаюць натуральным шляхам. Разгледзім вядомы прыклад задачы пра ханойскія вежы.

Ханойскія вежы. Ёсць тры стрыжні. На першым з іх у парадку памяншэння дыяметра (у выглядзе дзіцячай пірамідкі) змешчаны n дыскаў. Трэба перакласці ўсе гэтыя дыскі на трэці стрыжань, выкарыстоўваючы другі як прамежкавы, але ніколі дыск большага дыяметра нельга класці на дыск, дыяметр якога меншы.

Простыя разважанні прыводзяць да наступных рэкурэнтных суадносін:

$$T(n) = 2*T(n-1)+1, \qquad (1)$$

дзе T(n) — колькасць аперацый перакладання n дыскаў, T(n–1) — адпаведна, (n–1) дыску.

Сапраўды, каб перакласці самы вялікі дыск на трэці стрыжань, спачатку трэба перакласці n–1 дыск на другі і зрабіць пры гэтым T(n–1) аперацый перакладання, а пасля таго, як самы вялікі дыск перакладзены, зрабіць яшчэ T(n–1) такіх аперацый, перакладваючы дыскі з другога стрыжня на трэці [2].

Але калі існуюць, скажам, два алгарытмы рашэння адной і той жа задачы і адпавядаючыя гэтым алгарытмам рэкурэнтныя суадносіны, то зрабіць выснову, які з алгарытмаў па вылічальнай складанасці лепшы, вельмі цяжка. Каб зрабіць вывад, трэба мець рашэнне рэкурэнтных суадносін у замкнёнай форме, інакш кажучы, мець залежнасць ад n функцыі вылічальнай складанасці ў яўным выглядзе. У папярэднім артыкуле [3] быў разгледжаны адзін з метадаў атрымання такога рашэння — метад ітэрацый.

Скарыстаем гэты метад у задачы пра ханойскія вежы. Замест T(n–1) падставім у (1) 2*T(n–2)+1, потым анлагічна 2*T(n–3)+1 і г. д.

$$T(n) = 2*T(n-1)+1 = 2*(2*T(n-2)+1)+1 =$$
$$4*T(n-2)+3 = 4*(2*T(n-3)+1)+3 = 8*T(n-3)+7$$
$$=\ldots= 2^k*T(n-k)+2^k-1.$$

Такім чынам,

$$T(n) = 2^k* T(n-k)+2^k-1, \qquad (2)$$

дзе $k \le n-1$.

Улічым, што згодна з сэнсам задачы, T(1) = 1. Таму пры k = n–1 маем

$$T(n) = 2^{n-1}*T(1)+2^{n-1}-1 =2^{n-1}* 2-1 = 2^n-1. \qquad (3)$$

Другім метадам атрымання рашэння рэкурэнтных суадносін у замкнёнай форме з'яўляецца падстановачны метад [2; 4]. Сутнасць гэтага метаду ў тым, што ў рэкурэнтныя суадносіны, замест пакуль што невядомай функцыі, падстаўляецца такая фунцыя g(n), каб у выніку атрымалася сапраўдная няроўнасць. Функцыя g(n) павінна мець найменшы магчымы парадак. (Парадак функцыі вызначаецца хуткасцю ўзрастання самага хуткарастучага складаемага формулы). Пералічым у парадку ўзрастання складанасці некаторыя функцыі, якія ўжываюцца найбольш часта: $\log_2 n$, n, $n*\log_2 n$, n^2, n^3, ..., 2^n. Прыменім падстановачны метад да папярэдняга прыкладу. Відавочна, што трэба спрабаваць падставіць функцыю, у склад якой будзе ўваходзіць апошняя з пералічаных.

Возьмем $g(n) = 2^n+b$. Атрымаем наступную няроўнасць:

$2^n+b > 2*(2^{n-1}+b)+1$, адкуль вынікае, што $b \le -1$.

Узяўшы $b = -1$, маем (3).

У некаторых выпадках бывае мэтазгодным ужываць пэўную разнастайнасць падстановачнага метаду, якую мы называем метадам канечных рознасцей. Разгледзім наступны прыклад:

$$T(n+2) = 3*T(n+1) - 3*T(n)+T(n-1) \quad (4)$$

$T(1) = 2$

$T(2) = 8$

$T(3) = 18$.

Згодна з заданнем рэкурэнтных суадносін (4), вызначым некалькі першых членаў рэкурэнтнай паслядоўнасці T(n) і знойдзем рознасці паміж суседнімі членамі, потым — рознасці рознасцей (іншыя рознасці) і г. д. :

2 8 18 32 50 72 98 ...

6 10 14 18 22 26 ...

4 4 4 4 4 ...

Адзначым, што ўжо другая рознасць — канстанта. Таму, як вядома з тэорыі канечных рознасцей [5], паслядоўнасць T(n) можа быць задана паліномам другой ступені ад n.

$T(n) = a*n^2+b*n+c$,

дзе a, b, c — нявызначаныя каэфіцыенты. У гэты выраз для T(n) падставім паслядоўна n = 1, 2, 3. Як вынік маем:

$a+b+c = 2$

$4*a+2*b+c = 8$

$9*a+3*b+c = 18$

Калі разглядаць дадзеную сукупнасць роўнасцей як сістэму ўраўненняў адносна пераменных a, b, c, вызначаем каэфіцыенты a, b, c. Маем a = 2, b = c = 0. Такім чынам, рашэннем у замкнёнай форме рэкурэнтных суадносін (4) з'яўляецца

$$T(n) = 2*n^2. \quad (5)$$

Існуе метад ацэньвання складанасці алгарытмаў, звязаны з пабудовай рэкурсіўных дрэў [4; 6]. Па сутнасці ён з'яўляецца графічным прадстаўленнем метаду ітэрацый.

Рэкурэнтныя суадносіны ўзнікаюць не толькі ў тэорыі вылічальнай складанасці алгарытмаў. Так у курсе лікавых метадаў існуе рэкурэнтная формула трапецый:

$$T(j) = 0.5\ T(j-1)+h* \sum_{k=1}^{M} f(x_{2*k-1}) \text{ для } j = 1,2,\ldots,$$

дзе $h = (b-a)/2^j$ і $\{x_k = a+k*h\}$, $M=2^{j-1}$.

Выкарыстанне гэтай формулы істотна скарачае вылічальны працэс [7].

Літаратура

1. *Макконелл Дж.* Анализ алгоритмов. Вводный курс. М., 2002.
2. *Грэхем Р., Кнут Д., Паташник О.* Конкретная математика. Основание информатики. М., 1998.
3. *Паўлоўскі А. І., Панамарэнка В. К.* Рэкурэнтныя суадносіны ў курсах інфарматыкі педагагічнага універсітэта // Весці БДПУ. 2004. № 4. Сер. 3. С. 29—31.
4. *Котов В. М., Пилипчук Л. А., Соболевская Е. П.* Теория алгоритмов. Мн., 2001. Ч. 1.
5. *Гельфонд А. О.* Исчисление конечных разностей. М., 1967.
6. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М., 1999.
7. *Мэтьюз Д. Г., Финк К. Д.* Численные методы. Использование MATLAB. М.; СПб; Киев, 2001.

Summary

Connection between analysis of algorithms and recurrence relations is discussed in this article. Examples of solution in closed form are given.