4. kapitola. O efektivnosti dynamického programování

In: Jaroslav Morávek (author): O dynamickém programování. (Czech). Praha: Mladá fronta, 1973. pp. 34–37.

Persistent URL: http://dml.cz/dmlcz/403796

4. kapitola

O EFEKTIVNOSTI DYNAMICKÉHO PROGRAMOVÁNÍ

Pojednáme nyní o efektivnosti algoritmu dynamického programování z předešlé kapitoly. Efektivností algoritmu zde budeme rozumět asi tolik, jako rychlost, se kterou algoritmus řeší náš problém. Pro jednoduchost se budeme zabývat pouze problémem určení maximální hodnoty

$$f^* = \max \left\{ g_1(x_1) + \ldots + g_n(x_n) \,\middle|\, \begin{array}{l} x_1, \ldots, x_n \text{ celá a nezáp.} \\ x_1 + \ldots + x_n = a \end{array} \right\}$$

a otázku nalezení řešení ponecháme stranou.

Algoritmus dynamického programování vyžaduje určení hodnot $f_j(x)$ pro $j = 1, 2, \ldots, n$ a $x = 0, 1, \ldots, a$, tj. celkem $n(a + 1)$ hodnot. Abychom získali představu o značné efektivnosti tohoto algoritmu, srovnejme jej s tak zvaným *triviálním algoritmem* pro řešení stejného problému. Triviálním algoritmem pro nalezení extrému funkce, definované na konečné neprázdné množině, budeme rozumět, v souhlasu s běžně užívanou terminologií, metodu, záležející v postupném „vyčíslení" všech hodnot funkce (tj. pro všechny prvky jejího definičního oboru) a v určení extremální hodnoty z nich vzájemným srovnáním, tj. pomocí algoritmu popsaného v kapitole 2. Jinými slovy lze říci, že triviální algoritmus je zcela těžkopádný, mechanický postup určení extrému, nepoužívající žádné netriviální matematické myšlenky.

V našem případě bude pak triviální algoritmus záležet v tom, že se prozkoumají v nějakém libovolně zvoleném pořadí všechny n-tice $(x_1, x_2, \ldots, x_n)$ z množiny

$$\left\{(x_1, \ldots, x_n) \;\middle|\; \begin{array}{l} x_1, \ldots, x_n \text{ celá a nezáp.} \\ x_1 + \ldots + x_n = a \end{array}\right\} \qquad (2)$$

pro každou z nich se určí součet $g_1(x_1) + \ldots + g_n(x_n)$ a ze všech takto určených součtů se nalezne maximální. K posouzení efektivnosti triviálního algoritmu určíme počet prvků množiny (2).

Lemma 1:*) Počet všech prvků množiny (2) je $\binom{n+a-1}{n-1}$**).

Důkaz: Přiřaďme každé n-tici $(x_1, x_2, \ldots, x_n)$ z množiny (2) $(n-1)$-tici přirozených čísel $(y_1, y_2, \ldots, y_{n-1})$ takto:

$$y_1 = 1 + x_1,\ y_2 = 2 + x_1 + x_2,\ \ldots,$$
$$y_{n-1} = n - 1 + x_1 + \ldots + x_{n-1}$$

Zřejmě platí $1 \leqq y_1 < y_2 < \ldots < y_{n-1} \leqq n - 1 - a$.

Obráceně, každé $(n-1)$-tici přirozených čísel $(y_1, y_2, \ldots, y_{n-1})$ splňujících nerovnosti

$$1 \leqq y_1 < y_2 < \ldots < y_{n-1} \leqq n - 1 + a$$

---

*) Slovo „Lemma" je starořeckého původu a označuje pomocnou větu.

**) Připomeňme si definici symbolu $\binom{m}{p}$ (čti „m nad p"):

$\binom{m}{p} = \frac{m(m-1)\ldots(m-p+1)}{1.2.\ldots.p}$ pro všechny dvojice přirozených čísel m a p, kde $p \leqq m$

$\binom{m}{0} = 1$ pro všechna celá nezáporná m

odpovídá jediná n-tice $(x_1, x_2, \ldots, x_n)$ z množiny (2), splňující vztahy

$$y_1 = 1 + x_1, y_2 = 2 + x_1 + x_2, \ldots,$$
$$y_{n-1} = n - 1 + x_1 + \ldots + x_{n-1}$$

Odtud vyplývá, že počet prvků množiny (2) je roven počtu všech $(n-1)$-tic $(y_1, y_2, \ldots, y_{n-1})$ popsaného tvaru. Avšak těchto $(n-1)$-tic je právě tolik, kolik je všech $(n-1)$-prvkových podmnožin množiny $\{1, 2, \ldots, n-1+a\}$, tj. $\binom{n-1+a}{n-1}$, což dokončuje důkaz lemmatu.

Triviální algoritmus tedy vyžaduje k určení f^* vypočíst a srovnat $\binom{n-1+a}{n-1}$ součtů $g_1(x_1) + \ldots + g_n(x_n)$, zatímco netriviální algoritmus dynamického programování potřebuje k dosažení stejného cíle určit tabulku obsahující $n(a+1)$ čísel.

Ještě jasnější pohled na srovnání efektivnosti obou algoritmů získáme, určíme-li jimi požadovaný počet operací sčítání a srovnání. V případě triviálního algoritmu je to $(n-1)\binom{n-1+a}{n-1}$ sčítání a $\binom{n-1+a}{n-1} - 1$ srovnání. Při výpočtu

$$f_{j+1}(x) = \max \{f_j(x - x_{j+1}) + g_{j+1}(x_{j+1}) \mid x_{j+1} = 0, \ldots, x\}$$

v netriviálním algoritmu potřebujeme $x+1$ sčítání a x srovnání, takže výpočet všech hodnot $f_{j+1}(x)$ $(j = 1, \ldots, n-1; x = 0, 1, \ldots, a)$ vyžaduje celkem

$$(n-1)(1 + 2 + \ldots + (a+1)) = \frac{(n-1)(a+1)(a+2)}{2}$$

sčítání, a

$$(n-1)(0+1+\ldots+a) = \frac{(n-1)\,a\,(a+1)}{2} \text{ srovnání.}$$

Výsledek provedeného vyšetřování shrneme v tabulce 3

algoritmus	počet sčítání	počet srovnání
netriviální	$\frac{(n-1)\,(a+1)\,(a+2)}{2}$	$\frac{(n-1)\,a\,(a+1)}{2}$
triviální	$(n-1)\binom{n-1+a}{n-1}$	$\binom{n-1+a}{n-1}-1$

Tabulka 3

Ještě konkrétnější představu o srovnání efektivnosti obou algoritmů získá čtenář, dosadí-li si za n a a několik číselných hodnot, viz např. cvičení 1.

Cvičení

Cvičení 1: Porovnejte počet operací sčítání a srovnání v obou algoritmech pro tyto hodnoty a, n:

a) $a = 5$, $n = 5$ c) $a = 10$, $n = 10$
b) $a = 7$, $n = 7$ d) $a = 10$, $n = 15$

Cvičení 2: Určete počet všech uspořádaných n-tic $(x_1, \ldots, x_n)$ celých nezáporných čísel, splňujících vztah $x_1 + \ldots + x_n \leqq a$, kde a je dané celé číslo.

Cvičení 3: Pokuste se řešit cvičení 5a) a 5c) z kapitoly 3 triviálním algoritmem.