INFORMATIKA

Optimální způsob placení
(Úlohy z MO kategorie P, 40. část)

PAVEL TÖPFER

Matematicko-fyzikální fakulta UK, Praha

V našem seriálu článků o úlohách z Matematické olympiády kategorie P (programování) se tentokrát seznámíme s řešením jedné poměrně snadné úlohy z domácího kola 69. ročníku MO (školní rok 2019/20). Úloha pojednává o známém problému, jak zaplatit danou částku pomocí nejmenšího možného počtu bankovek či mincí. Ukážeme si, kdy lze při řešení použít jednoduchý hladový algoritmus a co lze dělat, pokud tento postup použít nemůžeme. Jako obvykle se nejprve seznámíme s přesným zadáním soutěžní úlohy.

* * * * * * * * * * * *

Soutěžní úloha

V Kocourkově mají systém platidel velmi podobný našemu. Nejmenší nominální hodnotou je 1 tolar. Existují mince a bankovky s následujícími pěknými, systematicky zvolenými nominálními hodnotami: 1, 2, 5, 10, 20, 50, 100, 200, 500, 1 000, 2 000, 5 000, 10 000, 20 000 a 50 000 tolarů.

Kocourkovská centrální banka se rozhodla, že vydá ještě jednu novou bankovku v hodnotě x tolarů. Obyvatelé Kocourkova si nyní kladou řadu otázek tohoto typu: Když budu chtít zaplatit přesně t_i tolarů, kolik nejméně kusů platidel stačí použít?

Formát vstupu a výstupu

Na prvním řádku vstupu je číslo x: nominální hodnota nové bankovky. Na druhém řádku je číslo q: počet otázek, které mají obyvatelé Kocourkova.

Na třetím řádku vstupu jsou mezerami oddělená čísla $t_1, \ldots, t_q$: sumy pro jednotlivé otázky. Na výstup vypište q řádků: postupně pro každou otázku jeden řádek s příslušnou správnou odpovědí.

Omezení a hodnocení

Vaše řešení bude testováno na pěti sadách vstupních dat, za správné vyřešení každé z nich dostanete dva body. V každé sadě platí $q \leq 50$ a je zaručeno, že x bude různé od nominálních hodnot existujících platidel. V jednotlivých sadách platí následující další omezení:

vstup číslo 1	$x \leq 20$	všechna $t_i \leq 20$
vstup číslo 2	$x = 100\,000$	všechna $t_i \leq 105$
vstup číslo 3	$x \leq 105$	všechna $t_i \leq 105$
vstup číslo 4	$x \leq 107$	všechna $t_i \leq 107$
vstup číslo 5	$x \leq 2.109$	všechna $t_i \leq 2.109$

Příklad

Vstup:	Výstup:
4700	3
4	2
53 9400 9401 30000	3
	2

Nová bankovka má hodnotu 4700 tolarů.

- Nejlepším způsobem, jak zaplatit 53 tolarů, je použít platidla s hodnotami $50 + 2 + 1$.
- Nejlepším způsobem, jak zaplatit 9400 tolarů, je použít dvě nové bankovky.
- Nejlepším způsobem, jak zaplatit 9401 tolarů, je použít dvě nové bankovky a jednu jednotolarovou minci.
- Nejlepším způsobem, jak zaplatit 30000 tolarů, je $10000 + 20000$.

* * * * * * * * * * * *

Úlohu nejprve vyřešíme pro původní pravidelnou sadu platidel bez nově přidané bankovky s hodnotou x. Pro optimální zaplacení jakékoliv částky můžeme použít jednoduchý hladový algoritmus. Dokud není celá částka zaplacena, opakujeme tento postup: vždy použijeme největší platidlo, jehož hodnota nepřevýší tu částku, kterou je ještě třeba zaplatit. Jedná se o zcela

přirozený intuitivní postup, který funguje i pro české koruny nebo třeba pro eura. Jeho správnost ale musíme pořádně zdůvodnit.

K důkazu správnosti uvedeného postupu přistoupíme odzadu. Při placení částky menší než 10 tolarů jistě můžeme použít pouze mince s hodnotami 1, 2 a 5 tolarů. Snadno ověříme rozborem všech případů, že hladový algoritmus najde optimální způsob zaplacení každé takové částky. Chceme-li zaplatit částku 10 tolarů nebo vyšší, v jakékoliv optimální platbě budou mince 1, 2 a 5 představovat celkovou sumu nejvýše 9 tolarů. Jinak by totiž bylo možné nahradit část z nich desetitolarovou mincí a tím počet kusů platidel snížit (případně platbu typu $5+2+2+2$ nahradíme pomocí $10+1$, což je také snížení počtu kusů platidel). To znamená, že při optimálním placení jakékoliv částky použijeme mince 1, 2 a 5 pouze k zaplacení „poslední cifry“ této částky. Již víme, že to udělá hladový algoritmus správně. Nyní můžeme zrušit mince 1, 2 a 5 a také smazat poslední cifru placené částky a celou úvahu zopakovat pro další řád a platidla 10, 20 a 50. Situace je tam naprosto stejná. Postupně můžeme takto zpracovat odzadu všechny dekadické řády placené částky a dostáváme tak, že hladový algoritmus funguje i pro desítky, stovky, tisíce a desetitisíce tolarů. Zbývá doplnit, co se stane, když placená částka převýší sto tisíc tolarů. Nejvyšší bankovka má hodnotu 50 000, takže optimální zaplacení každých 100 000 tolarů obsažených ve výsledné sumě jistě získáme pomocí dvou takových nejvyšších bankovek. Přesně to dělá náš hladový algoritmus.

Všimněte si, že popsaný hladový algoritmus nemůžeme automaticky použít pro jakoukoliv sadu platidel. O tom se snadno přesvědčíme pomocí dvou konkrétních příkladů. Uvažujme nejprve jednoduchou sadu platidel tvořenou pouze mincemi 1, 3 a 4 tolary, chceme zaplatit částku 6 tolarů. Výše uvedený hladový algoritmus použije nejprve minci s hodnotou 4 tolary a pak dvě po 1 tolaru, celkem tedy tři kusy. Toto řešení zjevně není optimální, lepší je použít dvě mince po 3 tolarech. Druhý příklad vychází přímo z naší řešené úlohy. Předpokládejme situaci v Kocourkově popsanou v zadání úlohy, nová bankovka má hodnotu 444 tolarů, chceme zaplatit částku 888 tolarů. Hladový algoritmus najde řešení $500+200+100+50+20+10+5+2+1$ tvořené devíti platidly, zatímco optimální platbu dostaneme použitím pouze dvou nových 444tolarových bankovek.

Vraťme se nyní k naší úloze. Bylo poměrně snadné vyřešit první dvě testovací sady vstupních dat a získat tak v soutěži za tuto úlohu alespoň část bodů. První sada obsahuje pouze velmi malé hodnoty a můžeme ji

proto vyřešit hrubou silou tak, že vyzkoušíme všechny možnosti placení. Ve druhé sadě nová bankovka s hodnotou x pouze prodlužuje pravidelnou sadu platidel, pro kterou je možné použít jednoduchý hladový algoritmus, který jsme popsali výše.

Ukážeme si, jak vypadá obecné řešení úlohy po přidání nové bankovky s hodnotou x. Důležité je uvědomit si, že nepracujeme s nějakou zcela nepravidelnou sadou platidel, ale s naší pěknou pravidelnou sadou, do níž je pouze přidána jedna „cizí“ hodnota navíc. Kdybychom při placení částky t věděli, kolik kusů bankovky x máme použít, bylo by řešení snadné. Zbývající částku chceme totiž zaplatit optimálním způsobem pomocí pravidelné sady platidel, pro kterou již můžeme použít hladový algoritmus. My sice nevíme, kolik kusů bankovky x je třeba použít, ale můžeme snadno vyzkoušet všechny možnosti (tzn. 0, 1, 2, 3, ...) a vybrat z nich tu nejlepší.

Máme navíc jistotu, že počet zkoumaných možností nebude příliš velký. Bankovku s hodnotou x použijeme určitě méně než 50 000krát. Pokud $x < 50000$, nemůže se vyplatit použít 50 000 kusů bankovky x, neboť místo nich je výhodnější použít x kusů bankovky s hodnotou 50 000. Pokud $x > 50000$, určitě nepoužijeme 50 000 kusů bankovky x, neboť $50000^2 > 2 \cdot 10^9$ a ze zadání úlohy víme, že všechny zkoumané částky jsou shora omezeny číslem $2 \cdot 10^9$.

Implementace popsaného algoritmu je velmi jednoduchá – vyzkoušíme všechny možnosti pro počet použitých kusů bankovky s hodnotou x a pro každou z nich určíme hladovým algoritmem, jak lze optimálně zaplatit zbývající částku. Z takto získaných možností vybereme to nejlepší. Vhodně přitom omezíme počet zkoumaných možností: při placení částky t nemá smysl zkoušet větší počet bankovek s hodnotou x, než kolika kusy platidel zaplatíme částku t hladově bez použití této přidané bankovky. Bylo by pochopitelně také zbytečné zkoušet větší počet bankovek x, než kolik činí podíl t/x.

```
program Placeni;

var x: longint;  {hodnota nové bankovky}
    q: integer;  {počet dotazů}
    t: longint;  {placená částka}
    pocetx: longint;  {počet nových bankovek}
    r: integer;       {další možné řešení}
    reseni: integer;  {výsledek}
    i: integer;
```

```
function Hladove(t: longint): word;
const platidla: array[1..15] of word
                = (1, 2, 5, 10, 20, 50, 100, 200, 500,
                   1000, 2000, 5000, 10000, 20000, 50000);
var pocet, i: integer;
begin
pocet := 0;
for i:= 15 downto 1 do
  begin
  pocet := pocet + t div platidla[i];
  t := t mod platidla[i]
  end;
Hladove := pocet
end; {function Hladove}

begin
readln(x);
readln(q);
for i := 1 to q do
  begin
  read(t);
  reseni := Hladove(t);
  pocetx := 1;
  while (pocetx < reseni) and (pocetx * x <= t) do
    begin
    r := pocetx + Hladove(t - pocetx * x);
    if r < reseni then reseni := r;
    pocetx := pocetx + 1
    end;
  writeln(reseni)
  end
end.
```